aws

開発者ガイド

Amazon Personalize

Amazon Personalize: 開発者ガイド

Table of Contents

Amazon Personalize とは

Amazon Personalize は、データを使用してユーザーにアイテムのレコメンデーションを生成する、フルマネージド型の機械学習サービスです。また、特定のアイテムやアイテムメタデータに対するユーザーの親和性に基づいてユーザーセグメントを生成することもできます。

一般的なユースケースには以下が含まれます。

- 動画ストリーミングアプリのカスタマイズ — 事前設定またはカスタマイズ可能な Amazon Personalize リソースを使用して、ストリーミングアプリに複数の種類のパーソナライズされたビデオのレコメンデーションを追加できます。上位のおすすめ、こちらもおすすめ、最も人気のあるビデオのレコメンデーションのことです。
- eコマースアプリへの製品レコメンデーションの追加 — 事前設定またはカスタマイズ可能な Amazon Personalize リソースを使用して、複数の種類のパーソナライズされた商品レコメンデーションを小売アプリに追加できます。おすすめ、よく一緒に購入されています、X を閲覧したお客様はこちらも閲覧しましたなどです。
- アプリへのリアルタイムの次善アクションの追加 - カスタマイズ可能な Amazon Personalize リソースを使用して、ユーザーの行動に基づいてユーザーがとる可能性が高いアクションを推奨できます。例えば、ロイヤルティプログラムへの登録、モバイルアプリのダウンロード、プロモーションメールへの登録について、リアルタイムのレコメンデーションを追加できます。
- パーソナライズされたメールの作成 — カスタマイズ可能な Amazon Personalize リソースを使用して、メールリストのすべてのユーザー向けにバッチレコメンデーションを生成できます。その後、AWS サービスまたはサードパーティのサービスを使用して、カタログ内のアイテムを推奨するパーソナライズされたメールをユーザーに送信できます。
- ターゲットを絞ったマーケティングキャンペーンの作成 — Amazon Personalize を使用して、カタログ内のアイテムを操作する可能性が最も高いユーザーのセグメントを生成できます。次に、AWS サービスまたはサードパーティのサービスを使用して、さまざまなアイテムをさまざまなユーザーセグメントに宣伝するターゲットを絞ったマーケティングキャンペーンを作成できます。
- 検索結果のパーソナライズ — カスタマイズ可能な Amazon Personalize リソースを使用して、ユーザーの検索結果をパーソナライズできます。例えば、Amazon Personalize は、 で生成した検索結果を再ランク付けできますOpenSearch。

ほとんどのユースケースで、Amazon Personalize は、主にアイテムインタラクションデータに基づいてレコメンデーションを生成します。アイテムインタラクションデータとは、カタログ内の商品

を操作したユーザーからのデータです。例えば、ユーザーがさまざまなアイテムをクリックしたとします。アイテムインタラクションデータは、CSV ファイル内の履歴の一括インタラクションレコードと、ユーザーがカタログを操作した際のリアルタイムイベントの両方から取得できます。Amazon Personalize では、ジャンル、価格、性別など、アイテムやユーザーからのデータも使用する場合があります。また、次善のアクションシナリオでは、アクションとアクションインタラクションデータが使用されます。

バルクデータをインポートする場合、Amazon SageMaker Data Wrangler を使用して 40 以上のソースからデータをインポートし、Amazon Personalize 用に準備できます。詳細については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。

Amazon Personalize には、リアルタイムのパーソナライゼーションのための API オペレーション、およびバルクレコメンデーションとユーザーセグメントのバッチオペレーションが含まれています。ビジネスドメインのために、ユースケース向けに最適化されたレコメンダーですぐに開始することも、独自の設定可能なカスタムリソースを作成することもできます。

トピック

- Amazon Personalize の料金
- Amazon Personalize の初回利用ユーザーに推奨されるガイダンス
- 関連 AWS サービスとソリューション
- サードパーティのサービス
- 詳細はこちら

Amazon Personalize の料金

Amazon Personalize では、最低料金や前払いの義務は発生しません。AWS 無料利用枠には、利用可能な AWS リージョンごとに最大 20 GB のデータ処理の月間クォータ、対象 AWS リージョンごとに最大 100 時間のトレーニング時間、最大 180,000 件のレコメンデーションリクエストがあります。無料利用枠は、使用を開始してから最初の 2 か月間有効です。

課金および料金の詳細な一覧については、「Amazon Personalize の料金表」を参照してください。

Amazon Personalize の初回利用ユーザーに推奨されるガイダンス

Amazon Personalize を初めて使用する場合は、開始にあたって次のリソースが役立ちます。

トピック

マジック・ムービー・マシンで Amazon Personalize の魅力を発見しよう

マジック・ムービー・マシーンは、インタラクティブな学習体験です。Amazon Personalize の機能を理解し、レコメンデーションの生成について詳しく知るのに役立ちます。簡単な紹介については、以下のビデオをご覧ください。そして、マジック・ムービー・マシンを試してみてください

Amazon Personalize の使用開始

このガイドをナビゲートする

以下のセクションを順番に読むことをお勧めします。

1. 仕組み – このセクションでは、Amazon Personalize のワークフローを概説し、ユーザー向けにパーソナライズされたエクスペリエンスを作成する手順を説明します。このセクションには、一般的な Amazon Personalize の用語とその定義も含まれています。レコメンデーションの取得を開始する前に、このセクションを完了して、Amazon Personalize のワークフローと用語を十分に理解していることを確認してください。
2. Amazon Personalize の設定 – このセクションでは、 を設定し AWS アカウント、Amazon Personalize を使用するために必要なアクセス許可を設定し、Amazon Personalize を使用および管理するために AWS CLI と AWS SDKs を設定します。
3. 開始 – このセクションでは、映画のシンプルなデータセットで Amazon Personalize の使用を開始します。これらのチュートリアルを完了して、Amazon Personalize を実際に体験しましょう。ドメインデータセットグループまたはカスタムデータセットグループのいずれかを開始することを選択できます。
   - ドメインデータセットグループの作成を開始するには、開始方法の前提条件 を完了してから、ドメインデータセットグループの開始方法 のチュートリアルを開始します。
   - カスタムデータセットグループを使用して開始するには、開始方法の前提条件 を完了してから、ドメインデータセットグループの開始方法 のチュートリアルを開始します。
4. ドメインのユースケースとカスタムレシピ — Amazon Personalize でモデルをトレーニングするために使用できるドメインのユースケースとカスタムレシピについて説明します。この情報を使用して、ユースケースを Amazon Personalize のリソースと一致させるのに役立てましょう。

5. 準備チェックリスト — 準備チェックリストを確認して、自身のデータで Amazon Personalize を使用する準備を始めましょう。このチェックリストには、Amazon Personalize の機能、要件、およびデータガイダンスのリストが記載されています。計画を立てるのに役立ちますし、Amazon Personalize でリソースを作成する際の参照として使用することもできます。
6. Amazon Personalize のワークフロー — このセクションでは、Amazon Personalize のワークフロー全体について説明します。ドメインデータセットグループまたはカスタムデータセットグループの作成、データの準備とインポート、レコメンダーまたはカスタムリソースの作成、レコメンデーションの取得の手順を説明します step-by-step。
7. イベントの記録 - このセクションでは、アイテムインタラクションイベントとアクションインタラクションイベントをリアルタイムで記録する方法について説明します。Amazon Personalize のリソースを設定したら、このセクションを完了して、データセットをユーザーの行動に合わせて最新の状態に保つ方法を学びましょう。
8. レコメンデーションとユーザーセグメントのフィルタリング – このセクションでは、レコメンデーションをフィルタリングする方法について説明します。カスタム基準に基づいてレコメンデーションをフィルタリングするフィルター式を作成する方法を学習するためにこのセクションを完了します。例えば、ユーザーがすでに購入した製品や、ユーザーがすでに視聴した映画を推奨したくない場合があります。

関連 AWS サービスとソリューション

Amazon Personalize は、他の AWS のサービスやソリューションとシームレスに統合します。例えば、以下のことが可能です。

- Amazon SageMaker Data Wrangler (Data Wrangler) を使用して、40 以上のソースから Amazon Personalize データセットにデータをインポートします。Data Wrangler は、データをインポート、準備、変換、分析するための end-to-end ソリューションを提供する Amazon SageMaker Studio の機能です。詳細については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。
- を使用して AWS Amplify 、アイテムインタラクションイベントを記録します。Amplify には、JavaScriptウェブクライアントアプリケーションからのイベントを記録するためのライブラリが含まれています。サーバーコードのイベントを記録するためのライブラリも含まれています。詳細については、「Amplify ドキュメント」を参照してください。
- 「機械学習を利用してパーソナライズされたエクスペリエンスの維持」により、Amazon Personalize タスクを自動化およびスケジュールします。この AWS ソリューションの実装

は、データのインポート、ソリューションバージョンのトレーニング、バッチワークフローなど、Amazon Personalize ワークフローを自動化します。

- Amazon CloudWatch Evidently を使用して、Amazon Personalize のレコメンデーションで A/B テストを実行します。詳細については、「CloudWatch Evidently を使用した A/B テスト」を参照してください。
- Amazon Pinpoint を使用して、ターゲットを絞ったマーケティングキャンペーンを作成します。Amazon Pinpoint と Amplify を使用して Amazon Personalize レコメンデーションをマーケティングメールキャンペーンとウェブアプリケーションに追加する方法を示す例については、「Amplify によるウェブ分析」を参照してください。

サードパーティのサービス

Amazon Personalize は、さまざまなサードパーティーサービスとうまく連携します。

- Amplitude — Amplitude を使用してユーザーアクションを追跡し、ユーザーの行動を理解するのに役立ちます。Amplitude と Amazon Personalize の使用方法については、 AWS パートナーネットワーク (APN) のブログ記事「Amplitude と Amazon Personalize によるパーソナライゼーションの効果の測定」を参照してください。
- Braze — Braze を使用して、カタログ内のアイテムを推奨するパーソナライズされたメールをユーザーに送信できます。Braze は市場をリードするメッセージングプラットフォーム (メール、プッシュ、SMS) です。Amazon Personalize と Braze を統合する方法を示すワークショップについては、「Amazon Personalize ワークショップ」を参照してください。
- mParticle — mParticle を使用してアプリからイベントデータを収集できます。mParticle と Amazon Personalize を使用してパーソナライズされた製品レコメンデーションを実装する方法を示す例については、「CDP の力を機械学習に活用する方法: パート 2」を参照してください。
- 最適化 — Optimizely を使用すると、Amazon Personalize のレコメンデーションによる A/B テストを実行できます。Optimizely と Amazon Personalize の使用方法については、「Optimizely と Amazon Personalize を統合して、強力な機械学習と実験を組み合わせる」を参照してください。
- セグメント — セグメントを使用してデータを Amazon Personalize に送信できます。セグメントと Amazon パーソナライズの統合の詳細については、「Amazon Personalize デスティネーション」を参照してください。

パートナーの全リストについては、「Amazon Personalize パートナー」を参照してください。

詳細はこちら

以下のリソースは、Amazon Personalize に関する追加情報を提供します。

- Amazon Personalize がお客様のユースケースに適合するかどうかを判断するのに役立つクイックリファレンスについては、Amazon Personalize サンプルリポジトリにある Amazon Personalize チートシート。を参照してください
- Amazon Personalize の使用方法に関する一連の動画については、「」にある「Amazon Personalize Deep Dive Video シリーズ」を参照してください YouTube。
- 詳細なチュートリアルとコードサンプルについては、amazon-personalize-samples GitHub リポジトリ「」を参照してください。

Amazon Personalize と生成 AI

Amazon Personalize は生成人工知能 (生成 AI) と高い親和性があります。Amazon Personalize Content Generator では、生成 AI を活用して、関連アイテムのバッチレコメンデーションに魅力的なテーマを追加できます。Content Generator は Amazon Personalize が管理する生成 AI 機能です。

また、Amazon Personalize レコメンデーションを使用して Amazon Personalize を生成 AI ワークフローと統合し、ユーザーエクスペリエンスを向上させることもできます。例えば、生成 AI プロンプトにレコメンデーションを追加して、各ユーザーの興味に合わせたマーケティングコンテンツを作成できます。また、推薦コンテンツの簡潔な概要を作成したり、チャットボットを通じて商品やコンテンツを推奨したりすることもできます。

次の動画は、Amazon Personalize と生成 AI を使用してレコメンデーションを強化する方法を示しています。

Amazon Personalize と生成 AI によるレコメンデーションの強化

以下の Amazon Personalize 機能は、生成 AI を使用するか、パーソナライズされたコンテンツを作成する生成 AI ソリューションの構築に役立ちます。生成 AI で Amazon Personalize を使用する方法を示すサンプル Jupyter Notebook については、Amazon Personalize サンプルリポジトリの「Generative AI with Amazon Personalize」を参照してください。

トピック

- Content Generator のテーマ付きレコメンデーション
- レコメンデーションメタデータ
- パーソナライゼーション用の事前設定済み LangChain コード

Content Generator のテーマ付きレコメンデーション

Amazon Personalize Content Generator では、わかりやすいテーマをバッチレコメンデーションに追加できます。Content Generator は Amazon Personalize が管理する生成 AI 機能です。

テーマ付きバッチレコメンデーションを受け取ると、Amazon Personalize Content Generator は類似アイテムのセットごとにわかりやすいテーマを追加します。例えば、朝食用フードの関連商品のアイテムレコメンデーションが表示される場合、Amazon Personalize は元気な朝や朝の必需品などのテーマを生成する場合があります。このテーマを、よく一緒に買われるものなどの一般的なカルー

セルタイトルの代わりに使用できます。また、このテーマをプロモーションメールやマーケティングキャンペーンに組み込んで、新しいメニューオプションを用意することもできます。

テーマを生成するには、アイテムインタラクションとアイテムデータセットにデータをインポートし、Similar-Items レシピを使用してカスタムソリューションを作成し、バッチレコメンデーションを生成します。アイテムデータにはアイテムの説明とタイトル情報が含まれている必要があります。アイテムの説明とタイトルを詳細なものにすると、Content Generator がより正確で魅力的なテーマを作成するのに役立ちます。

- Amazon Personalize のワークフローの詳細については、「Amazon Personalize のワークフロー」を参照してください。
- バッチレコメンダーの詳細については、「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。
- テーマ付きレコメンデーションの生成については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

レコメンデーションメタデータ

レコメンデーションを受け取ると、Amazon Personalize がアイテムデータセットから各推奨アイテムに関するメタデータを返すように設定できます。このメタデータを Amazon Personalize のレコメンデーションと共に、生成 AI プロンプトに追加して、より説得力のあるコンテンツを生成できます。

例えば、生成 AI を使用してマーケティングメールを作成できます。Amazon Personalize のレコメンデーションとそのメタデータ (映画のジャンルなど) は、生成 AI のプロンプトエンジニアリングの一部として使用できます。パーソナライズされたプロンプトにより、生成 AI を使用して各顧客の興味に合わせた魅力的なマーケティング E メールを作成できます。

レコメンデーションメタデータを取得するには、まず Amazon Personalize ワークフローを完了してデータをインポートし、ドメインリソースまたはカスタムリソースを作成します。Amazon Personalize レコメンダーまたはキャンペーンを作成するときは、レコメンデーションにメタデータを含めるオプションを有効にします。レコメンデーションを受け取ったら、どのアイテムデータの列を含めるかを指定できます。

- Amazon Personalize のワークフローの詳細については、「Amazon Personalize のワークフロー」を参照してください。
- レコメンダーのメタデータを有効にする方法については、「レコメンデーションのメタデータを有効にする (ドメインリソース)」を参照してください。

- キャンペーンのメタデータを有効にする方法については、「レコメンデーションのメタデータを有効にする (カスタムリソース)」を参照してください。
- Amazon Personalize と生成 AI を組み合わせてマーケティングキャンペーンを作成する方法の詳細については、「Amazon Personalize と生成系 AI でマーケティングソリューションを高度化する」を参照してください。

パーソナライゼーション用の事前設定済み LangChain コード

LangChain は、言語モデルを活用したアプリケーションを開発するためのフレームワークです。Amazon Personalize 用に構築されたコードを備えています。このコードを使用して、Amazon Personalize のレコメンデーションを生成 AI ソリューションと統合できます。

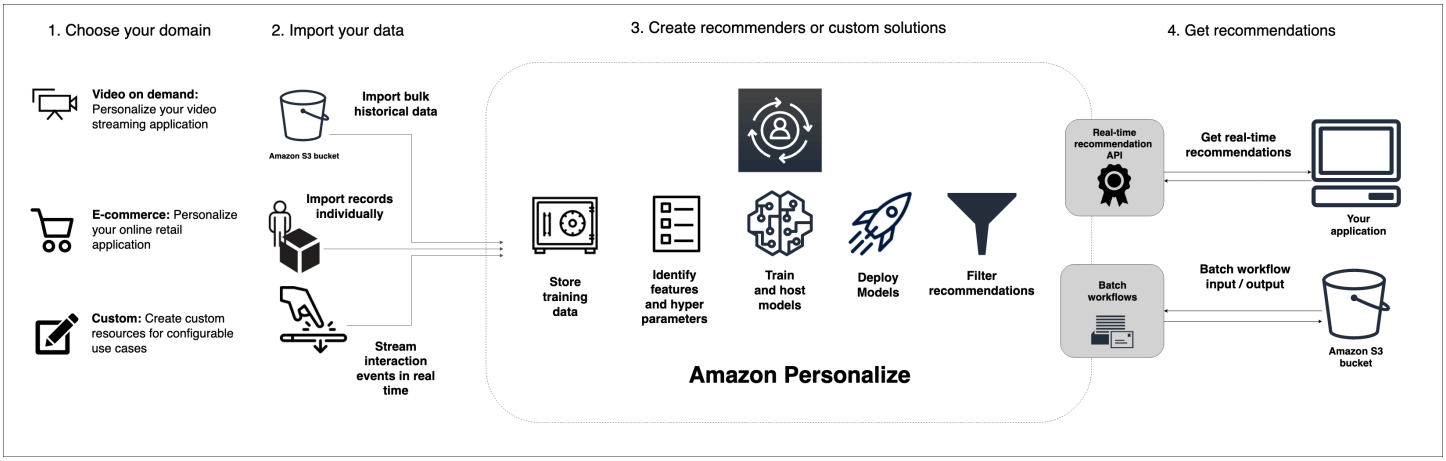
例えば、次のコードを使用して、ユーザー向けの Amazon Personalize レコメンデーションをチェーンに追加できます。

```
from aws_langchain import AmazonPersonalize
from aws_langchain import AmazonPersonalizeChain
from langchain.llms.bedrock import Bedrock

recommender_arn="RECOMMENDER ARN"

bedrock_llm = Bedrock(model_id="anthropic.claude-v2", region_name="us-west-2")
client=AmazonPersonalize(credentials_profile_name="default",region_name="us-
west-2",recommender_arn=recommender_arn)
# Create personalize chain
# Use return_direct=True if you do not want summary
chain = AmazonPersonalizeChain.from_llm(
    llm=bedrock_llm,
    client=client,
    return_direct=False
)
response = chain({'user_id': '1'})
print(response)
```

- の開始方法については LangChain、 LangChain ドキュメントの「Introduction」を参照してください。
- より高度な LangChain コードサンプルなど、Amazon Personalize 用に構築されたコードの使用については、AWS サンプルリポジトリの「Amazon Personalize LangChain 拡張機能」を参照してください。

仕組み

Amazon Personalize は、データを使用して、ドメインベースまたはカスタマイズ可能なレコメンデーションモデルをトレーニングします。アプリケーションでプライベートレコメンデーション API を使用して、リアルタイムのレコメンデーションをリクエストします。Amazon Personalize は、アイテムのレコメンデーションとユーザーセグメントを取得するバッチワークフローもサポートしています。



トピック

- Amazon Personalize ワークフローの概要
- Amazon Personalize の用語
- Amazon Personalize が使用できるデータの種類

Amazon Personalize ワークフローの概要

Amazon Personalize ワークフローは次のとおりです。

1. データセットグループ

   データセットグループは、Amazon Personalize のリソースのコンテナです。作成するデータセットグループのタイプによって、Amazon Personalize ワークフローのステップ 3 で作成できるリソースが決まります。

   - ドメインデータセットグループを使用すると、VIDEO_ON_DEMAND ドメインまたは eコマース ドメインのユースケースのレコメンデーションを作成できます。Amazon Personalize は、これらのレコメンダーの設定、トレーニング、更新を管理します。ドメインデータセットグループから始める場合でも、カスタムリソースを追加できます。ドメインデータセットグループに

は、アクションデータセットやアクションインタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。

- または、カスタムリソースを使用してカスタムデータセットグループを作成できます。これらには、ソリューション、ソリューションバージョン、キャンペーンが含まれます。これらのリソースでは、構成、更新、再トレーニングをより細かく制御できます。

2. データを準備してインポートする

   インタラクション、アイテム、ユーザー、およびアクションインタラクションのレコードをデータセットにインポートします (データ用の Amazon Personalize のコンテナ)。レコードは一括でインポートすることも、個別にインポートすることもできます。一括データをインポートする場合、Amazon SageMaker Data Wrangler を使用して 40 以上のソースからデータをインポートし、Amazon Personalize 用に準備できます。詳細については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。

   Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

3. ドメインレコメンダーまたはカスタムリソースを作成する

   データをインポートしたら、ドメインレコメンダー (ドメインデータセットグループ用) またはカスタムリソース (カスタムデータセットグループ用) を作成して、データに基づいてモデルをトレーニングします。これらのリソースを使用して、レコメンデーションを生成します。

4. レコメンデーションの取得

   レコメンダーやカスタムキャンペーンを使ってレコメンデーションを取得しましょう。カスタムデータセットグループでは、バッチレコメンデーションやユーザーセグメントも取得できます。

Amazon Personalize ワークフローを初めて完了した後は、データを最新の状態に保ち、カスタムソリューションを定期的に再トレーニングしてください。これにより、モデルはユーザーの最新のアクティビティから学習し、レコメンデーションの関連性を維持および改善できます。詳細については、「レコメンデーションの関連性の維持」を参照してください。

Amazon Personalize の用語

このセクションでは、Amazon Personalize で使用される用語をご紹介します。

トピック

データのインポートと管理

次の用語は、Amazon Personalize でのデータのインポート、エクスポート、およびフォーマットに関連するものです。

アクションデータセット

アクションに関するメタデータのコンテナ。アクションとは、モバイルアプリのインストールやロイヤルティプログラムへの参加など、ユーザーに推奨するエンゲージメントや収益を生み出すアクティビティです。アクションのメタデータには、アクションの有効期限のタイムスタンプ、値、繰り返し頻度データ、カテゴリメタデータが含まれる場合があります。このタイプのデータは、 でのみ使用されますNext-Best-Action レシピ。

アクションインタラクションデータセット

ユーザーとアクション間のやり取りから収集する履歴データとリアルタイムデータのコンテナ。各アクションインタラクションは、userID、actionID、タイムスタンプ、イベントタイプ、およびインタラクションに関するその他のデータ (カテゴリ別メタデータなど) で構成されます。このタイプのデータは、 でのみ使用されますNext-Best-Action レシピ。

コンテキストメタデータ

イベント (クリックなど) が発生したときに、ユーザーの閲覧コンテキスト (使用デバイスや場所など) について収集するインタラクションデータ。コンテキストメタデータは、新規ユーザーおよび既存のユーザーに対する推奨事項の関連性を向上させることができます。

データセット

Amazon Personalize にアップロードするデータのコンテナ。Amazon Personalize データセットには、ユーザー、アイテム、アイテムインタラクションデータセット、アクションの 5 種類があります。

データセットグループ

データセット、ドメインレコメンダー、カスタムリソースなどの Amazon Personalize リソースのコンテナです。データセットグループは、リソースを独立したコレクションにまとめるため、あるデータセットグループのリソースが他のデータセットグループのリソースに影響を与えるこ

とはできません。データセットグループは、ドメインデータセットグループまたはカスタムデータセットグループのいずれかになります。

ドメインデータセットグループ

さまざまなビジネスドメインおよびユースケース用に事前設定されたリソースを含むデータセットグループ。Amazon Personalize は、トレーニングモデルとデプロイのライフサイクルを管理します。ドメインデータセットグループを作成するときは、ビジネスドメインを選択し、データをインポートして、各ユースケースのレコメンダーを作成します。アプリケーションでレコメンダーを使用して、 オペレーションで GetRecommendationsレコメンデーションを取得します。

ドメインデータセットグループで始める場合でも、カスタムユースケースのレシピでトレーニングされたソリューションやソリューションバージョンなどのカスタムリソースを追加できます。

カスタムデータセットグループ

ソリューション、ソリューションバージョン、フィルター、キャンペーン、バッチ推論ジョブなどのカスタムリソースを含むデータセットグループ。キャンペーンを使用して、 オペレーションで GetRecommendationsレコメンデーションを取得します。トレーニングモデルとデプロイのライフサイクルを管理します。カスタムデータセットグループから始める場合、後でそれをドメインに関連付けることはできません。代わりに、新しいドメインデータセットグループを作成します。

データセットのエクスポートジョブ

データセット内のレコードを Amazon S3 バケット内の 1 つ以上の CSV ファイルに出力するレコードエクスポートツール。出力 CSV ファイルには、データセットのスキーマのフィールドと一致する列名を持つヘッダー行が含まれています。

データセットのインポートジョブ

Amazon S3 バケット内の CSV ファイルからのデータを、Amazon Personalize のデータセットに取り込む一括インポートツール。

イベント

クリック、購入、動画視聴など、ユーザーが記録し、Amazon Personalize のアイテムインタラクションデータセットにアップロードするユーザーアクション。CSV ファイルから一括で、Amazon Personalize コンソールを使用して増分的に、およびリアルタイムで、イベントをインポートします。

明示的なインプレッション

Amazon Personalize のアイテムインタラクションデータセットに手動で追加するアイテムのリスト。暗黙的なインプレッション (Amazon Personalize がレコメンデーションデータから自動的に派生させるもの) とは異なり、明示的なインプレッションに含めるものを選択します。

暗黙的なインプレッション

アプリケーションがユーザーに表示するレコメンデーションです。アイテムインタラクションデータセットに手動で追加する明示的なインプレッションとは異なり、Amazon Personalize は、レコメンデーションデータから暗黙的なインプレッションを自動的に派生させます。

インプレッションデータ

クリック、表示、購入などによって、ユーザーが特定のアイテムを操作したときに、ユーザーに提示したアイテムのリスト。Amazon Personalize では、インプレッションデータを使用して、ユーザーが同じ製品を選択または無視した頻度に基づいて、ユーザーの新製品の関連性を計算します。

インタラクションデータセット

ユーザーと製品間のやり取り (イベントといいます) から収集された履歴データとリアルタイムデータのコンテナ。インタラクションデータには、イベントタイプデータとコンテキストメタデータ を含めることができます。

アイテムデータセット

料金、ジャンル、在庫状況など、製品に関するメタデータのコンテナ。

繰り返し頻度

Actions データセットにインポートできるアクションメタデータのタイプ。繰り返し頻度データは、Action インタラクションデータセット内のユーザーの履歴に基づいて、ユーザーが特定のアクションとやり取りした後、Amazon Personalize がその特定のアクションを推奨するまで待機する日数を指定します。

スキーマ

データの構造を Amazon Personalize に知らせる Apache Avro 形式の JSON オブジェクト。Amazon Personalize は、スキーマを使用してデータを解析します。

ユーザーデータセット

年齢、性別、ロイヤルティメンバーシップなど、ユーザーに関するメタデータのコンテナ。

トレーニング

次の用語は、Amazon Personalize でのモデルのトレーニングに関連するものです。

item-to-item 類似度 (SIMS) レシピ

RELATED_ITEMS レシピ。インタラクションデータセットのデータを使用して、指定された製品に類似する製品のレコメンデーションを作成します。SIMS レシピは、料金や色などの製品メタデータを照合するのではなく、ユーザーがアイテムを操作する方法に基づいて類似度を計算します。

item-affinity

アイテムインタラクションデータセットとアイテムデータセットのデータを使用して、ユーザーがアイテムを操作する可能性に基づいて指定した、各アイテムのユーザーセグメントを作成する USER_SEGMENTATION レシピ。

item-attribute-affinity

アイテムインタラクションデータセットとアイテムデータセットのデータを使用して、ユーザーが属性を使用してアイテムを操作する可能性に基づいて指定した、各アイテム属性のユーザーセグメントを作成する USER_SEGMENTATION レシピ。

Next-Best-Action レシピ

このレシピは、ユーザーにとって次に最適なアクションに関するリアルタイムのレコメンデーションを生成します。ユーザーにとって次善のアクションは、ユーザーが実行する可能性が最も高いアクションです。例えば、ロイヤルティプログラムへの登録、アプリのダウンロード、クレジットカードの申請などです。詳細については、Next-Best-Action レシピを参照してください。

Personalized-Ranking-v2 レシピ

PERSONALIZED_RANKING のレシピ。特定のユーザーについて予測される関心レベルに基づいて、提供する製品のコレクションをランク付けします。このレシピは、トランスフォーマーベースのアーキテクチャを使用して、アイテムインタラクションデータ、アイテムメタデータ、ユーザーメタデータから学習するモデルをトレーニングします。Personalized-Ranking-v2 レシピを使用して、特定のユーザー向けにパーソナライズされたアイテムまたは検索結果の厳選されたリストの順序をパーソナライズします。最大 500 万項目でトレーニングし、以前のバージョンよりもレイテンシーが低く、より関連性の高いレコメンデーションを生成できます。

personalized-ranking の recipe

PERSONALIZED_RANKING のレシピ。特定のユーザーについて予測される関心レベルに基づいて、提供する製品のコレクションをランク付けします。パーソナライズされたランキングレシ

ピを使用して、特定のユーザー向けにパーソナライズされたアイテムまたは検索結果のキュレーションされたリストの順序をパーソナライズします。

popularity-count の recipe

USER_PERSONALIZATION レシピ。一意のユーザーとのやり取りが最も多いアイテムを推奨します。

レコメンダー

レコメンデーションを生成するドメインデータセットグループのツール。ドメインデータセットグループのレコメンダーを作成し、アプリケーションで を使用して GetRecommendations API でリアルタイムのレコメンデーションを取得します。レコメンダーを作成する場合、ユースケースを指定します。そして、Amazon Personalize は、ユースケースに最適な設定でレコメンダーをサポートするモデルをトレーニングします。

recipe

ユーザーが操作する製品を予測する (USER_PERSONALIZATION レシピの場合)、またはユーザーが関心を示した特定の製品に類似する製品を計算する (RELATED_ITEMS レシピの場合)、または予測された関心に基づいて提供した製品のコレクションをランク付けするように事前構成された Amazon Personalize アルゴリズムの特定のユーザー。

solution

Amazon Personalize がレコメンデーションを生成するために使用するレシピ、カスタマイズされたパラメータ、およびトレーニング済みモデル (ソリューションバージョン)。

ソリューションバージョン

Amazon Personalize でソリューションの一部として作成するトレーニング済みのモデル。キャンペーンでソリューションバージョンをデプロイして、レコメンデーションのリクエストに使用するパーソナライゼーション API をアクティブ化します。

トレーニングモード

ソリューションバージョンを作成するときに実行するトレーニングの範囲。FULL と UPDATE の 2 つの異なるモードがあります。FULL モードでは、データセットグループ内のデータセットからのトレーニングデータ全体に基づいて、完全に新しいソリューションバージョンが作成されます。UPDATE は、既存のソリューションバージョンを増分的に更新して、前回のトレーニング以降に追加した新しいアイテムを推奨します。

> **Note**
>
> User-Personalization-v2、User-Personalization、または Next-Best-Action を使用すると、Amazon Personalize は FULL トレーニングモードでトレーニングされた最新のソリューションバージョンを自動的に更新します。自動更新 を参照してください。

User-Personalization-v2 レシピ

USER_PERSONALIZATION レシピは、ユーザーが好みに基づいて操作するアイテムを推奨します。このレシピは、トランスフォーマーベースのアーキテクチャを使用して、アイテムインタラクションデータ、アイテムメタデータ、ユーザーメタデータから学習するモデルをトレーニングします。最大 500 万項目でトレーニングし、以前のバージョンよりもレイテンシーが低く、より関連性の高いレコメンデーションを生成できます。

User-Personalization レシピ

ユーザーがインタラクションするアイテムを予測する、Hierarchical Recurrent Neural Network (HRNN) ベースの USER_PERSONALIZATION レシピ。user-personalization の recipe では、製品の探索とインプレッションのデータを使用して、新製品の推奨事項を生成できます。

モデルのデプロイとレコメンデーション

以下の用語は、モデルのデプロイと使用に関するレコメンデーションの生成に関連しています。

アクション最適化期間

Amazon Personalize が、ユーザーが実行する可能性が最も高いアクションを予測するときに使用する期間。例えば、アクションの最適化期間が 14 日の場合、Amazon Personalize はユーザーが今後 14 日間に実行する可能性が最も高いアクションを予測します。を使用してソリューションを作成するときに、アクション最適化期間を設定しますNext-Best-Action レシピ。

バッチ推論ジョブ

Amazon S3 バケットからバッチ入力データをインポートし、ソリューションバージョンを使用してレコメンデーションを生成し、レコメンデーションを Amazon S3 バケットにエクスポートするツール。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。バッチ推論ジョブを使用して、リアルタイムの更新を必要としない大規模なデータセットのレコメンデーションを取得します。

バッチセグメントジョブ

Amazon S3 バケットからバッチ入力データをインポートし、ソリューションバージョンを使用してユーザーセグメントを作成し、ユーザーセグメントを Amazon S3 バケットにエクスポートするツール。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。USER_SEGMENTATION レシピにサポートされたソリューションでバッチセグメントジョブを使用して、ユーザーがさまざまなアイテムまたはさまざまなアイテム属性を持つアイテムを操作する可能性に基づいてユーザーのセグメントを作成します。

キャンペーン

専用のプロビジョンドトランザクション容量を搭載した、デプロイ済みのソリューションバージョン (トレーニング済みモデル)。アプリケーションユーザー向けのリアルタイムレコメンデーションを作成します。キャンペーンを作成したら、`getRecommendations` または `getPersonalizedRanking` の API 操作を使用してレコメンデーションを取得します。

製品調査

探索では、新しいアイテムまたはアクション、インタラクションがほとんどないアイテムまたはアクション、以前の行動に基づいてユーザーに関連性が低いアイテムまたはアクションなど、通常はユーザーにレコメンデーションされる可能性が低いアイテムまたはアクションがレコメンデーションに含まれます。

メトリクス属性

アイテムレコメンデーションの影響を測定するために使用するツール。メトリクス属性は、インポートしたアイテムインタラクションとアイテムのデータ、および指定したメトリクスに基づいてレポートを作成します。例えば、ユーザーが視聴した映画の合計時間やクリックイベントの総数などです。

レコメンデーション

Amazon Personalize が、ユーザーが操作すると予測する商品のリスト。レコメンデーションは、使用している Amazon Personalize レシピに応じて、製品のリスト (USER_PERSONALIZATION レシピと RELATED_ITEMS レシピ) か、指定した製品コレクションのランキング (PERSONALIZED_RANKING レシピ) のいずれかになります。

ユーザーセグメント

Amazon Personalize が、カタログを操作すると予測するユーザーのリスト。使用した USER_SEGMENTATION レシピに応じて、アイテム (Item-Affinity レシピ) アイテムメタデータ (Item-Attribute-Affinity レシピ) に基づいてユーザーセグメントを作成します。バッチセグメントジョブを使用してユーザーセグメントを作成します。

Amazon Personalize が使用できるデータの種類

次のトピックでは、Amazon Personalize にインポートできるさまざまなタイプのデータをご紹介します。

トピック

インタラクションデータ

インタラクションは、記録してからトレーニングデータとしてインポートするイベントです。Amazon Personalize は、主にインタラクションデータに基づいてレコメンデーションを生成します。インタラクションデータには以下が含まれます。

- イベントタイプとイベント値のデータ
- コンテキストメタデータ
- インプレッションデータ

インタラクションデータをアイテムインタラクションデータセットにインポートします。インタラクションデータセットの詳細については、「アイテムインタラクションデータセット」を参照してください。

アイテムデータ

Amazon Personalize が使用できるアイテムメタデータには以下が含まれます。

- 価格など、各商品に関する数値データ。
- 商品のジャンルや色など、各商品に関するカテゴリメタデータ。
- 各アイテムの作成タイムスタンプデータ。
- 商品説明や映画のあらすじなど、構造化されていないテキストメタデータ。

アイテムに関するメタデータを Items データセットにインポートします。データセットの詳細については、「製品データセット」を参照してください。

ユーザーデータ

Amazon Personalize が使用できるユーザーメタデータには以下が含まれます。

- 各ユーザーに関する数値データ (年齢など)。
- 性別やロイヤルティメンバーシップのステータスなど、各ユーザーに関するカテゴリ別メタデータ。

ユーザーに関するメタデータを Amazon Personalize の Users データセットにインポートします。データセットの詳細については、「ユーザーデータセット」を参照してください。

アクションデータ

Amazon Personalize が使用できるアクションデータには以下が含まれます。

- 各アクションのビジネス価値または重要性。
- 季節性や独占権など、各アクションのカテゴリメタデータ。
- Amazon Personalize が各アクションの推奨を停止するタイミングを指定するアクション有効期限タイムスタンプデータ。
- ユーザーが操作を行った後、Amazon Personalize が各アクションを推奨するまで待機する時間を指定する繰り返し頻度データ。

アクションに関するデータをアクションデータセットにインポートします。ドメインデータセットグループには、アクションデータセットやアクションインタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。アクションデータセットの詳細については、「Actions データセット」を参照してください。

アクションインタラクションデータ

Amazon Personalize がアクションを伴うユーザーインタラクションから使用できるデータには以下が含まれます。

- イベントタイプデータ
- カテゴリ別メタデータ

インタラクションデータをアクションインタラクションデータセットにインポートします。ドメインデータセットグループには、アクションデータセットやアクションインタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。アクションインタラクションデータセットの詳細については、「Action インタラクションデータセット」を参照してください。

Amazon Personalize の設定

Amazon Personalize を使用する前に、管理者ユーザーの存在する Amazon Web Services (AWS) アカウントが必要です。必要なアクセス許可を設定したら、Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsにアクセスできます。

トピック

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、 を保護し AWS アカウントのルートユーザー、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「IAM ユーザーガイド」の AWS アカウント 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、「 AWS サインイン ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

リージョンとエンドポイント

エンドポイントは、ウェブサービスのエントリポイントとなるURLです。各エンドポイントは特定の AWS リージョンに関連付けられています。特定のキャンペーンのすべての Amazon Personalize コンポーネント (データセット、ソリューション、キャンペーン AWS CLI、イベントトラッカー) を同じリージョンに作成する必要があるため、Amazon Personalize コンソール、、および Amazon Personalize SDKs のデフォルトリージョンに注意してください。Amazon Personalize によってサポートされているリージョンとエンドポイントについては、「リージョンとエンドポイント」を参照してください。

アクセス許可のセットアップ

ユーザー、グループ、またはロールに Amazon Personalize のリソースを操作するためのアクセス許可を付与する必要があります。また、Amazon Personalize で作成したリソースにアクセスし、ユーザーに代わってタスクを実行するためのアクセス許可を Amazon Personalize に付与する必要があります。

アクセス許可を設定するには

1. ユーザー、グループ、またはロールに Amazon Personalize リソースを操作するアクセス許可を付与し、ロールを Amazon Personalize に渡します。Amazon Personalize にアクセスする許可をユーザーに付与する を参照してください。
2. Amazon Personalize に Amazon Personalize 内のお客様のリソースへのアクセス許可と、お客様に代わってタスクを実行する許可を付与してください。Amazon Personalize にリソースへのアクセス許可を付与する を参照してください。

3. Amazon Personalize サービスロールの信頼ポリシーを変更して、混乱した代理問題を防ぎましょう。信頼関係ポリシーの例については、「サービス間での不分別な代理処理の防止」を参照してください。ロールの信頼ポリシーを変更する方法については、「ロールの変更」を参照してください。
4. 暗号化に AWS Key Management Service（ AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize の IAM サービスロールにキーを使用するアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。
5. Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与 のステップを完了し、IAM と Amazon S3 バケットポリシーを使用して Amazon S3 リソースへのアクセス権を Amazon Personalize に付与します。

トピック

- Amazon Personalize にアクセスする許可をユーザーに付与する
- Amazon Personalize にリソースへのアクセス許可を付与する
- Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与
- AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する

Amazon Personalize にアクセスする許可をユーザーに付与する

ユーザーに Amazon Personalize へのアクセスを提供するには、Amazon Personalize リソースへのアクセス許可を付与する IAM ポリシーを作成し、Amazon Personalize にロールを渡します。次に、ユーザー、グループ、ロールに許可を追加する際にそのポリシーを使用します。

ユーザー用に新規 IAM ポリシーを作成する

Amazon Personalize に Amazon Personalize のリソースへのフルアクセスを提供する IAM ポリシーを作成します。

JSON ポリシーエディタを使用してポリシーを作成するには

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。

初めて [ポリシー] を選択する場合には、[管理ポリシーにようこそ] ページが表示されます。[今すぐ始める] を選択します。

3. ページの上部で、[ポリシーの作成] を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. 次の JSON ポリシードキュメントを入力します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "personalize.amazonaws.com"
                }
            }
        }
    ]
}
```

6. [次へ] をクリックします。

> **Note**
>
> いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、IAM ユーザーガイドの「ポリシーの再構成」を参照してください。

7. [確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。[このポリシーで定義されているアクセス許可] を確認して、ポリシーによって付与されたアクセス許可を確認します。
8. [ポリシーの作成] をクリックして、新しいポリシーを保存します。

Amazon Personalize のタスクの実行に必要なアクセス許可のみを付与するには、前述のポリシーを変更して、ユーザーに必要なアクションのみを付与するようにします。Amazon Personalize のアクションの一覧については、「Amazon Personalize のアクション、リソース、および条件キー」を参照します。

Amazon Personalize へのアクセスの提供

ユーザーにアクセス許可を与える際には、新しい IAM ポリシーをアタッチします。

アクセスを提供するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- AWS IAM Identity Center のユーザーとグループ:

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「シークレットの作成と管理」の手順に従ってください。
- ID プロバイダーを通じて IAM で管理されているユーザー:

  ID フェデレーションのロールを作成する。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ユーザー:
  - ユーザーが設定できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (非推奨) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加します。「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス許可の追加」の指示に従います。

Amazon Personalize にリソースへのアクセス許可を付与する

リソースへのアクセス許可を Amazon Personalize に付与する IAM ポリシーを作成します。
または、 AWS 管理`AmazonPersonalizeFullAccess`ポリシーを使用することもできます。`AmazonPersonalizeFullAccess`は、必要以上のアクセス許可を提供します。必要なアクセス許

可のみを付与する新しい IAM ポリシーを作成することをお勧めします。管理ポリシーの詳細については、「AWS マネージドポリシー」を参照してください。

ポリシーを作成後、Amazon Personalize 向けの IAM ロールを作成して新しいポリシーをアタッチします。

トピック

- 新しい Amazon Personalize の IAM ポリシーの作成
- Amazon Personalize 向けの IAM ロールの作成

新しい Amazon Personalize の IAM ポリシーの作成

Amazon Personalize に Amazon Personalize のリソースへのフルアクセスを提供する IAM ポリシーを作成します。

JSON ポリシーエディタでポリシーを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。

   初めて [ポリシー] を選択する場合には、[管理ポリシーにようこそ] ページが表示されます。[今すぐ始める] を選択します。
3. ページの上部で、[ポリシーを作成] を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. 次の JSON ポリシードキュメントを入力します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:*"
            ],
            "Resource": "*"
        }
    ]
}
```

6. [次へ] をクリックします。

> **Note**
>
> いつでも [Visual] と [JSON] エディタオプションを切り替えることができます。ただし、[Visual] エディタで [次] に変更または選択した場合、IAM はポリシーを再構成して visual エディタに合わせて最適化することがあります。詳細については、「IAM ユーザーガイド」の「ポリシーの再構成」を参照してください。

7. [確認と作成] ページで、作成するポリシーの [ポリシー名] と [説明] (オプション) を入力します。[このポリシーで定義されているアクセス許可] を確認して、ポリシーによって付与されたアクセス許可を確認します。
8. [ポリシーの作成] をクリックして、新しいポリシーを保存します。

Amazon Personalize 向けの IAM ロールの作成

Amazon Personalize を使用するには、Amazon Personalize のサービス AWS Identity and Access Management ロールを作成する必要があります。サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。Amazon Personalize のサービスロールを作成したら、必要に応じてそのロールに その他のサービスロールのアクセス許可 に記載されている追加のアクセス権限を付与します。

Amazon Personalize 向けのサービスロールを作成するには (IAM コンソール)

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。
3. 信頼できるエンティティタイプ で、AWS のサービス を選択します。
4. サービスまたはユースケース で、Amazon Personalize を選択し、次に Personalize ユースケースを選択します。
5. [次へ] をクリックします。
6. 前の手順で作成したポリシーを使用します。
7. (オプション) アクセス許可の境界を設定します。このアドバンスド機能は、サービスロールで使用できますが、サービスにリンクされたロールではありません。

a. [アクセス許可の境界の設定] セクションを開き、[アクセス許可の境界を使用してロールのアクセス許可の上限を設定する] を選択します。

   IAM には、アカウント内の AWS 管理ポリシーとカスタマー管理ポリシーのリストが含まれます。

b. アクセス許可の境界として使用するポリシーを選択します。

8. [次へ] をクリックします。
9. このロールの目的を識別しやすいロール名またはロール名サフィックスを入力します。

> **Important**
>
> ロールに名前を付けるときは、次のことに注意してください。
>
> - ロール名は 内で一意である必要があり AWS アカウント、大文字と小文字を区別することはできません。
>
>   例えば、**PRODROLE** と **prodrole** の両方の名前でロールを作成することはできません。ロール名がポリシーまたは ARN の一部として使用される場合、ロール名は大文字と小文字が区別されます。ただし、サインインプロセスなど、コンソールにロール名がユーザーに表示される場合、ロール名は大文字と小文字が区別されません。
> - 他のエンティティがロールを参照する可能性があるため、ロールを作成した後にロール名を編集することはできません。

10. (オプション) [説明] にロールの説明を入力します。
11. (オプション) ロールのユースケースとアクセス許可を編集するには、[ステップ 1: 信頼されたエンティティを選択] または [ステップ 2: アクセス権限を追加] のセクションで [編集] を選択します。
12. (オプション) ロールの識別、整理、検索を簡単にするには、キーと値のペアとしてタグを追加します。IAM でのタグの使用に関する詳細については、『IAM ユーザーガイド』の「IAM リソースにタグを付ける」を参照してください。
13. ロールを確認したら、[Create role] (ロールを作成) を選択します。

Amazon Personalize のロールを作成したら、Amazon S3 バケットと任意のキー へのアクセスを許可する準備が整います。 AWS KMS

その他のサービスロールのアクセス許可

ロールを作成し、このロールに Amazon Personalize のリソースへのアクセス権を与えたら、以下を行います。

1. Amazon Personalize サービスロールの信頼ポリシーを変更して、混乱した代理問題を防ぎましょう。信頼関係ポリシーの例については、「サービス間での不分別な代理処理の防止」を参照してください。ロールの信頼ポリシーを変更する方法については、「ロールの変更」を参照してください。
2. 暗号化に AWS Key Management Service（ AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize の IAM サービスロールにキーを使用するアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与

Amazon Personalize に Amazon S3 バケットへのアクセス権を付与するには、次の手順を実行します。

1. まだ行っていない場合は、アクセス許可のセットアップ のステップに従って許可を設定して、あなたに代わって Amazon Personalize が ¥Amazon Personalize 内のリソースにアクセスできるようにします。
2. Amazon S3 バケットへのアクセスを許可するポリシーを Amazon Personalize のサービスロール (「Amazon Personalize 向けの IAM ロールの作成」を参照) にアタッチします。詳細については、「Amazon Personalize サービスロールに対する、Amazon S3 ポリシーのアタッチ」を参照してください。
3. データファイルを含む Amazon S3 バケットにバケットポリシーをアタッチして、Amazon Personalize がそれらにアクセスできるようにします。詳細については、「Amazon S3 バケットに対する、Amazon Personalize のアクセスポリシーのアタッチ」を参照してください。
4. 暗号化に AWS Key Management Service（ AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize の IAM サービスロールに、 キーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

> **Note**
>
> Amazon Personalize は AWS VPCs、Amazon Personalize は VPC アクセスのみを許可する Amazon S3 バケットとやり取りできません。

トピック

- Amazon Personalize サービスロールに対する、Amazon S3 ポリシーのアタッチ
- Amazon S3 バケットに対する、Amazon Personalize のアクセスポリシーのアタッチ

Amazon Personalize サービスロールに対する、Amazon S3 ポリシーのアタッチ

Amazon S3 ポリシーを Amazon Personalize のロールにアタッチするには、次の手順を実行します。

1. IAM コンソール (https://console.aws.amazon.com/iam/) にサインインします。
2. ナビゲーションペインで、[ポリシー]、[ポリシーの作成] の順に選択します。
3. [JSON] タブを選択し、以下のようにポリシーを更新します。bucket-name をバケットの名前に置き換えます。データセットのインポートジョブまたはデータ削除ジョブには、次のポリシーを使用できます。バッチワークフローを使用している場合、またはデータセットのエクスポートジョブを作成している場合、Amazon Personalize には追加のアクセス許可が必要です。「バッチワークフロー用のサービスロールのポリシー」または「データセットをエクスポートするための Amazon S3 バケットポリシー」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
```

```
        }
    ]
}
```

4. [次へ : タグ] を選択します。オプションでタグを追加し、[Review] (レビュー) を選択します。
5. ポリシーに名前を付けます。
6. (オプション) [Description] (説明) で、このポリシーを説明する短い文 (「**Allow Amazon Personalize to access its Amazon S3 bucket.**」など) を入力します
7. [ポリシーの作成] を選択します。
8. ナビゲーションペインで、[Roles] (ロール) を選択し、Amazon Personalize 用に作成したロールを選択します。Amazon Personalize 向けの IAM ロールの作成 を参照してください。
9. [Permissions] (許可) で、[Attach policies] (ポリシーをアタッチ) を選択します。
10. リストのポリシーを表示するには、[Filter policies (ポリシーのフィルタ)] フィルタボックスにポリシー名の一部を入力します。
11. この手順の前半で作成したポリシーの横にあるチェックボックスをオンにします。
12. Attach policy] (ポリシーのアタッチ) を選択します。

    ロールを Amazon Personalize で使用できるようにする前に、データを含む Amazon S3 バケットにバケットポリシーをアタッチする必要もあります。Amazon S3 バケットに対する、Amazon Personalize のアクセスポリシーのアタッチ を参照してください。

バッチワークフロー用のサービスロールのポリシー

バッチワークフローを完了するには、Amazon S3 バケットにアクセスしてファイルを追加するための許可を Amazon Personalize に付与する必要があります。上記の手順に従って、次のポリシーを Amazon Personalize のロールにアタッチします。bucket-name をバケットの名前に置き換えます。バッチワークフローの詳細については、「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

データセットをエクスポートするためのサービスロールのポリシー

データセットをエクスポートするため、Amazon Personalize のサービスロールでは、Amazon S3 バケットで PutObject および ListBucket アクションを使用するための許可が必要です。次のポリシー例は、PutObject と ListBucket の許可を Amazon Personalize に付与します。bucket-name をバケットの名前に置き換えて、Amazon Personalizeのサービスロールに対してポリシーをアタッチします。サービスロールへのポリシーのアタッチについては、「Amazon Personalize サービスロールに対する、Amazon S3 ポリシーのアタッチ」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

Amazon S3 バケットに対する、Amazon Personalize のアクセスポリシーのアタッチ

Amazon Personalize には、S3 バケットにアクセスするための許可が必要です。データセットのインポートジョブまたはデータ削除ジョブには、次のポリシーを使用できます。bucket-name をバケットの名前に置き換えます。バッチワークフローについては、「バッチワークフロー向けの Amazon S3 バケットポリシー」を参照してください。

Amazon S3 バケットポリシーの詳細については、「S3 バケットポリシーを追加するにはどうすればよいですか?」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "personalize.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

バッチワークフロー向けの Amazon S3 バケットポリシー

バッチワークフローの場合、Amazon Personalize には、Amazon S3 バケットにアクセスしてファイルを追加するための許可が必要です。次のポリシーをバケットにアタッチします: bucket-name をバケットの名前に置き換えます。

Amazon S3 バケットポリシーをバケットに追加する方法の詳細については、「S3 バケットポリシーを追加するにはどうすればよいですか?」を参照してください。バッチワークフローの詳細については、「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "personalize.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

データセットをエクスポートするための Amazon S3 バケットポリシー

データセットをエクスポートするには、Amazon S3 バケットで PutObject および ListBucket アクションを使用するための許可が Amazon Personalize に付与されている必要があります。次のポリシー例は、PutObject と ListBucket の許可を Amazon Personalize のプリンシプルに付与します。bucket-name をバケットの名前に置き換えて、ポリシーをバケットにアタッチします。Amazon S3 バケットポリシーをバケットに追加する方法については、Amazon Simple Storage Service ユーザーガイドの「S3 バケットポリシーを追加するにはどうすればよいですか?」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "personalize.amazonaws.com"
```

```
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する

Amazon Personalize コンソールまたは API を使用するときに AWS Key Management Service (AWS KMS) キーを指定する場合、またはキーを使用して Amazon S3 バケットを暗号化する場合は、AWS KMS キーを使用するアクセス権限を Amazon Personalize に付与する必要があります。アクセス権限を付与するには、AWS KMSキーポリシーとサービスロールにアタッチされている IAM ポリシーで、キーを使用するアクセス権限を Amazon Personalize に付与する必要があります。これは Amazon Personalize で以下を作成する場合に適用されます。

- データセットグループ
- データセットのインポートジョブ (権限を付与する必要があるのは AWS KMS キーポリシーのみ)
- データセットのエクスポートジョブ
- バッチ推論ジョブ
- バッチセグメントジョブ
- メトリクス属性

AWS KMS キーポリシーと IAM ポリシーは、以下のアクションに対するアクセス許可を付与する必要があります。

- Decrypt
- GenerateDataKey
- DescribeKey
- CreateGrant (キーポリシーでのみ必要)
- ListGrants

リソースの作成後に AWS KMS キー権限を取り消すと、フィルターの作成時や推奨情報の取得時に問題が発生する可能性があります。AWS KMS ポリシーの詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS でキーポリシーを使用する」を参照してください。IAM ポリシーの作成については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。IAM アイデンティティへのポリシーのアタッチに関する詳細については、「IAM ユーザーガイド」の「IAM ID のアクセス許可の追加および削除」を参照してください。

トピック

キーポリシーの例

以下のキーポリシーの例では、Amazon Personalize とお客様のロールに、上記の Amazon Personalize オペレーションに必要な最低限のアクセス権限を付与しています。データセットグループの作成時にキーを指定し、データセットからデータをエクスポートする場合は、キーポリシーに `GenerateDataKeyWithoutPlaintext` アクションを含める必要があります。

```
{
  "Version": "2012-10-17",
  "Id": "key-policy-123",
  "Statement": [
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<account-id>:role/<personalize-role-name>",
        "Service": "personalize.amazonaws.com"
      },
      "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:ListGrants"
            ],
      "Resource": "*"
    }
  ]
}
```

IAM ポリシーの例

次の IAM ポリシーの例では、前述の Amazon Personalize オペレーションに必要な最小限の AWS KMS アクセス権限をロールに付与します。データセットのインポートジョブの場合、アクセス権限を付与する必要があるのは AWS KMS キーポリシーだけです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:DescribeKey",
                "kms:ListGrants"
            ],
            "Resource": "*"
        }
    ]
}
```

のセットアップ AWS CLI

AWS Command Line Interface （ AWS CLI) は、Amazon Personalize を含む AWS のサービスを管理するための統合デベロッパーツールです。このツールをインストールすることをお勧めします。

1. をインストールするには AWS CLI、AWS Command Line Interface 「インターフェイスユーザーガイド」の「 AWS Command Line Interfaceのインストール」の手順に従います。
2. を設定し AWS CLI 、 を呼び出すプロファイルを設定するには AWS CLI、 ユーザーガイドの「 AWS CLIの設定AWS Command Line Interface 」の手順に従います。
3. AWS CLI プロファイルが正しく設定されていることを確認するには、次のコマンドを実行します。

   ```
   aws configure --profile default
   ```

   プロファイルが正しく設定されている場合は、次のような出力が表示されます。

   ```
   AWS Access Key ID [****************52FQ]:
   ```

```
AWS Secret Access Key [****************xgyZ]:
Default region name [us-west-2]:
Default output format [json]:
```

4. AWS CLI が Amazon Personalize で使用するように設定されていることを確認するには、次のコマンドを実行します。

   ```
   aws personalize help
   ```

   また、

   ```
   aws personalize-runtime help
   ```

   また、

   ```
   aws personalize-events help
   ```

   が正しく設定されている場合 AWS CLI 、Amazon Personalize、Amazon Personalize ランタイム、および Amazon Personalize イベントでサポートされている AWS CLI コマンドのリストが表示されます。

   をセットアップ AWS CLI しても Amazon Personalize のコマンドが認識されない場合は、 を更新します AWS CLI。を更新するには AWS CLI、次のコマンドを実行します。

   ```
   pip3 install awscli --upgrade --user
   ```

   詳細については、「pip を使用した AWS CLI のインストール」を参照してください。

AWS SDKsのセットアップ

使用する AWS SDKsをダウンロードしてインストールします。このガイドでは、SDK for Python (Boto3)、SDK for Java 2.x、および SDK for JavaScript v3 の例を示します。他の AWS SDKs「Amazon Web Services のツール」を参照してください。Amplify の設定について詳しくは、「AWS Amplify」を参照してください。

- AWS SDK for Python (Boto3)

SDK for Python (Boto3) をインストールするには、Boto3 ドキュメントの「クイックスタート」の手順に従います。

- SDK for Java 2.x

  SDK for Java 2.x の設定については、「AWS SDK for Java 2.x デベロッパーガイド」の「SDK for Java 2.x の開始方法」のトピックを参照してください。

  Amazon Personalize のコード例については、AWS SDK サンプルリポジトリにある「Amazon Personalize Java コードサンプル」を参照してください。

- AWS SDK for JavaScript v3

  SDK for JavaScript v3 の設定については、「 デベロッパーガイド」の AWS SDK for JavaScript「 の開始方法」トピックを参照してください。 AWS SDK for JavaScript

  Amazon Personalize のコード例については、 SDK サンプルリポジトリの「 SDK for JavaScript v3 の Amazon Personalize コードAWS 例」を参照してください。

開始

以下のセクションは、Amazon Personalize コンソール、 AWS CLI、および AWS SDKsの使用を開始するのに役立ちます。このチュートリアルでは、600 名のユーザーからの 9,700 本の映画に対する 100,000 件のレーティングで構成される履歴データを使用します。

チュートリアルを簡略化するには:

- ここでは小さなデータセットを使います。これにより、リソースが生成するあらゆる指標に悪影響が及ぶ可能性があります。チュートリアルは Amazon Personalize ワークフローの紹介を目的としており、必ずしも最高のパフォーマンスを発揮するモデルを生成できるわけではありません。
- アイテムインタラクションデータセットのみを作成します。ユーザーが映画を見たという事実に基づいており、映画の評価内容は問いません。これにより、トレーニングデータを簡単に作成することができます。
- イベントレコードのライブユーザーのインタラクションイベントは記録されません。ユーザーイベントをキャプチャする方法については、「イベントの記録」を参照してください。

ドメインデータセットグループまたはカスタムデータセットグループを開始することを選択できます。

- ドメインデータセットグループは、ドメインに基づいてさまざまなユースケース向けに最適化されたリソースを提供します。ドメインデータセットグループの作成を開始するには、開始方法の前提条件 を完了してから、ドメインデータセットグループの開始方法 のチュートリアルを完了します。
- カスタムデータセットグループを使用すると、カスタムリソースのみを作成および設定できます。カスタムリソースとUser-Personalization-v2 レシピレシピを使用してパーソナライズされた映画のレコメンデーションをユーザーに提供し始めるには、 を完了開始方法の前提条件してから、 でチュートリアルを開始しますカスタムデータセットグループの開始方法。

開始方法の演習を完了したら、不要な料金が発生しないように、作成したリソースを、「リソースのクリーンアップ」の手順に従って削除します。

トピック

- 開始方法の前提条件
- ドメインデータセットグループの開始方法

- カスタムデータセットグループの開始方法
- リソースのクリーンアップ

開始方法の前提条件

以下のステップは、入門ガイド演習の前提条件です。

1. Amazon Personalize がユーザーに代わってリソースにアクセスできるように許可を設定します。これには、Amazon Personalize のサービスロールを作成し、IAM ポリシーを使用して Amazon Personalize リソースへのアクセスを許可することが含まれます。詳細については、Amazon Personalize にリソースへのアクセス許可を付与するを参照してください。
2. トレーニングデータを準備し、このデータを Amazon S3 バケットにアップロードします。
   - ドメインデータセットグループのチュートリアルについては、「トレーニングデータの作成 (ドメインデータセットグループ)」を参照してください。
   - カスタムデータセットグループのチュートリアルについては、「トレーニングデータの作成 (カスタムデータセットグループ)」を参照してください。
3. Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与 で指定されているように、Amazon S3 リソースにアクセスするための許可を Amazon Personalize のサービスロールに付与します。

トレーニングデータの作成 (ドメインデータセットグループ)

トレーニングデータを作成するには、映画のレーティングデータをダウンロードして変更し、Amazon Simple Storage Service (Amazon S3) バケットに保存します。その後、バケットから読み取るための許可を Amazon Personalize に付与します。

トレーニングデータを作成するには

1. 教育と開発にMovieLens推奨される (F. Maxwell Harper と Joseph A. Konstan) から、映画評価 zip ファイル ml-latest-small.zip をダウンロードして解凍します。 2015 年。 MovieLens データセット: 履歴とコンテキスト。 インタラクティブインテリジェントシステム (TiiS) の ACM トランザクション 5、4: 19:1 ~ 19:19. https://doi.org/10.1145/2827872)。
2. `ratings.csv` ファイルを開きます。このファイルには、このチュートリアルのインタラクションデータが含まれています。

   a. [評価] 列を削除します。

b. `userId` と `movieId` の列の名前をそれぞれ `USER_ID` と `ITEM_ID` に変更します。

c. EVENT_TYPE 列を追加して、すべてのレコードの値を `watch` に設定します。Microsoft Excel を使用している場合は、列の最初のセルに `watch` を入力し、セルの右下をダブルクリックすることで、すべてのレコードに EVENT_TYPE を設定できます。ヘッダーは次のようになります。

`USER_ID,ITEM_ID,TIMESTAMP,EVENT_TYPE`

Amazon Personalize でデータを認識するには、これらの列を正確に表示する必要があります。データの最初の数行は次のようになります。

```
USER_ID,ITEM_ID,TIMESTAMP,EVENT_TYPE
1,1,964982703,watch
1,3,964981247,watch
1,6,964982224,watch
1,47,964983815,watch
1,50,964982931,watch
....
....
```

`ratings.csv` ファイルを保存します。

3. Amazon S3 バケットに `ratings.csv` をアップロードします。詳細については、Amazon Simple Storage Service ユーザーガイドの「ドラッグアンドドロップを使用したファイルとフォルダのアップロード」を参照してください。

4. バケット内のデータを読み取るための許可を Amazon Personalize に付与します。詳細については、「Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与」を参照してください。

トレーニングデータの作成 (カスタムデータセットグループ)

トレーニングデータを作成するには、映画のレーティングデータをダウンロードして変更し、Amazon Simple Storage Service (Amazon S3) バケットに保存します。その後、バケットから読み取るための許可を Amazon Personalize に付与します。

1. 教育と開発にMovieLens推奨される (F. Maxwell Harper と Joseph A. Konstan) から、映画評価 zip ファイル ml-latest-small.zip をダウンロードして解凍します。 2015 年。 MovieLens データ

セット: 履歴とコンテキスト。 インタラクティブインテリジェントシステム (TiiS) の ACM トランザクション 5、4: 19:1 ~ 19:19. https://doi.org/10.1145/2827872)。

2. `ratings.csv` ファイルを開きます。このファイルには、このチュートリアルのインタラクションデータが含まれています。

   a. [評価] 列を削除します。

   b. ヘッダー行を以下のように置き換えます。

      `USER_ID,ITEM_ID,TIMESTAMP`

      Amazon Personalize でデータを認識するには、これらのヘッダーを正確に表示する必要があります。

   `ratings.csv` ファイルを保存します。

3. Amazon S3 バケットに `ratings.csv` をアップロードします。詳細については、Amazon Simple Storage Service ユーザーガイドの「ドラッグアンドドロップを使用したファイルとフォルダのアップロード」を参照してください。

4. バケット内のデータを読み取るための許可を Amazon Personalize に付与します。詳細については、「Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与」を参照してください。

ドメインデータセットグループの開始方法

この入門ガイドチュートリアルでは、VIDEO_ON_DEMAND ドメインのドメインデータセットグループを作成し、CSV ファイルからインタラクションデータをインポートします。その後、上位のおすすめユースケースを含むレコメンダーを作成します。次に、レコメンダーを使用して、ユーザー向けの映画のレコメンデーションを取得します。このチュートリアルでは、600 名のユーザーからの 9,700 本の映画に対する 100,000 件のレーティングで構成される履歴データを使用します。

はじめに、開始方法の前提条件 を完了してから、Amazon Personalize リソースの作成方法に応じて、ドメインデータセットグループの開始方法 (コンソール)、ドメインデータセットグループの開始方法、ドメインデータセットグループの開始方法 (SDK for Java 2.x)、または ドメインデータセットグループの開始方法 (SDK for JavaScript v3) のいずれかに進みます。

開始方法の演習を完了したら、不要な料金が発生しないように、作成したリソースを、「リソースのクリーンアップ」の手順に従って削除します。

トピック

- ドメインデータセットグループの開始方法 (コンソール)
- ドメインデータセットグループの開始方法 (SDK for Java 2.x)
- ドメインデータセットグループの開始方法
- ドメインデータセットグループの開始方法 (SDK for JavaScript v3)

ドメインデータセットグループの開始方法 (コンソール)

この演習では、Amazon Personalize コンソールを使用して、ドメインデータセットグループと、特定のユーザー向けの映画のレコメンデーションを返すレコメンダーを作成します。

この演習を開始する前に、開始方法の前提条件 を確認してください。

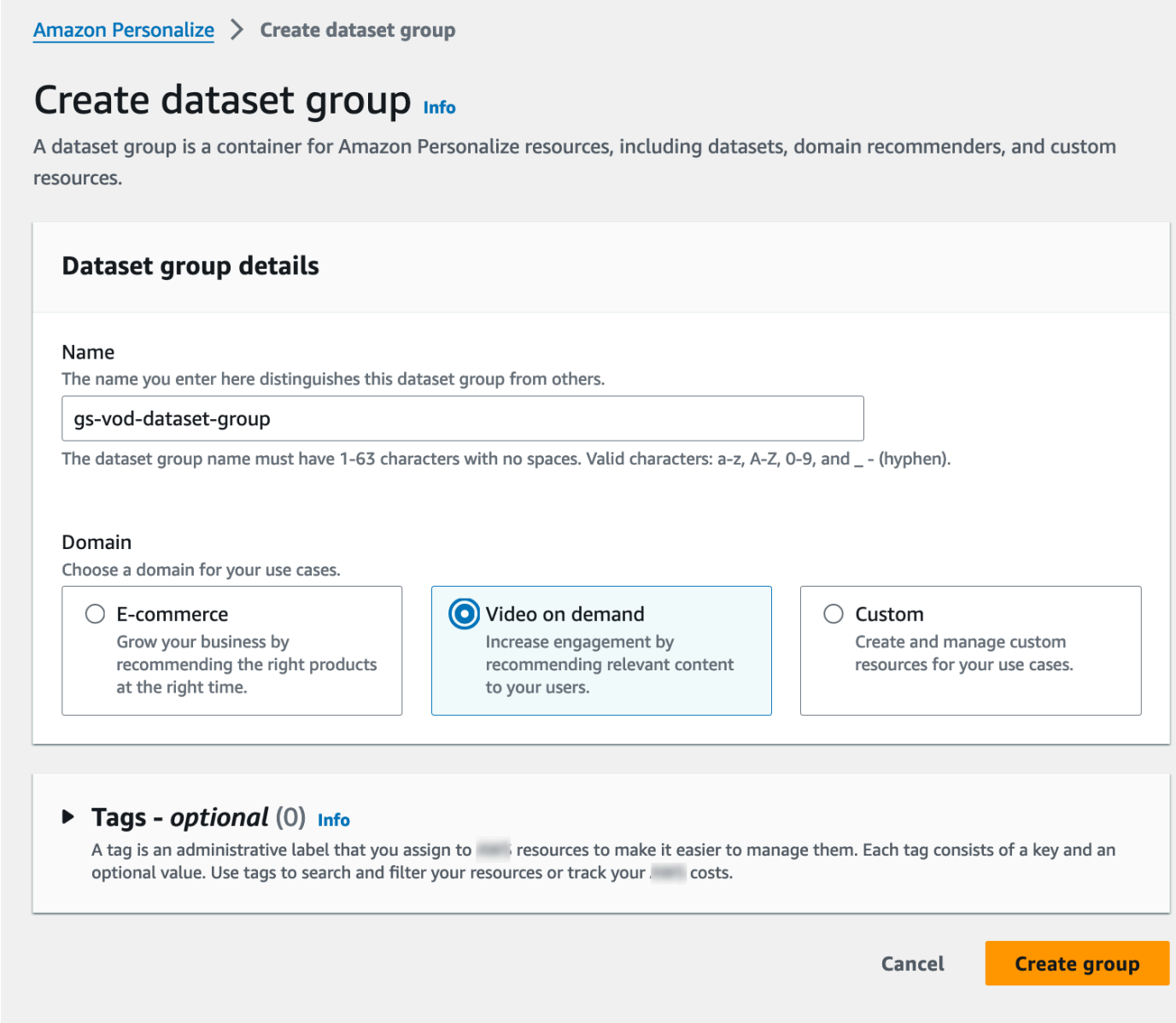
開始方法の演習を完了したら、不要な料金が発生しないように、作成したリソースを、「リソースのクリーンアップ」の手順に従って削除します。

ステップ 1: ドメインデータセットグループを作成する

この手順では、VIDEO_ON_DEMAND ドメインのドメインデータセットグループを作成し、デフォルトの VIDEO_ON_DEMAND ドメイン向けスキーマを使用してアイテムインタラクションデータセットを作成して、トレーニングデータの作成 (ドメインデータセットグループ) で作成したアイテムインタラクションデータをインポートします。

ドメインデータセットグループを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. ナビゲーションペインで、[Create dataset group] (データセットグループを作成) を選択します。
3. データセットグループの詳細でデータセットグループの名前を指定します。
4. [Domain] (ドメイン) で、[Video on demand] (ビデオオンデマンド) を選択します。選択したドメインによって、データのインポート時に使用するデフォルトスキーマが決まります。また、レコメンダーが利用できるユースケースが決まります。画面の表示は次のようになります。



5. [データセットグループの作成] を選択します。概要が表示されます。ステップ 2: データをインポートする に進みます。

ステップ 2: データをインポートする

この手順では、デフォルトの VIDEO_ON_DEMAND ドメインスキーマを使用してアイテムインタラクションデータセットを作成します。次に、トレーニングデータの作成 (ドメインデータセットグループ) で作成したアイテムインタラクションデータをインポートします。

データをインポートするには

1. 概要ページのステップ 1。データセットを作成してデータをインポートし、データセットの作成を選択し、アイテムインタラクションデータセット を選択します。
2. [Amazon Personalize データセットにデータを直接インポートする] を選択し、[次へ] を選択します。
3. [アイテムインタラクションスキーマを設定] ページで、[データセット名] にアイテムインタラクションデータセットの名前を入力します。
4. [Dataset schema] (データセットのスキーマ) で、[Create a new domain schema by modifying the existing default schema for your domain] (ドメインの既存のデフォルトスキーマを変更して新しいドメインスキーマを作成) を選択し、スキーマの名前を入力します。[スキーマの定義] によって VIDEO_ON_DEMANDドメインのデフォルトスキーマの表示が更新されます。スキーマは変更しないでおきます。画面の表示は次のようになります。

Configure item interactions schema Info

Dataset details

Dataset name

The name you enter here can help you distinguish this dataset import job from others.

gs-interactions-ds

The dataset name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Dataset schema

The schema you provide allows Amazon Personalize to understand and import your data.

- (●) Create a new domain schema by modifying the existing default schema for your domain
- ( ) Use an existing domain related schema

Schema name

The name you enter here can help you distinguish this schema from others.

gs-interactions-domain-schema

The schema name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Schema definition

Verify your data structure matches the following schema.

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "USER_ID",
      "type": "string"
    },
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "TIMESTAMP",
      "type": "long"
    },
```

5. [次へ] をクリックします。[アイテムインタラクションデータセットのインポートジョブの設定] ページが表示されます。
6. [アイテムインタラクションデータセットのインポートジョブの設定] ページで、[データのインポートソース] を [S3 からデータをインポート] のままにしておきます。
7. [Dataset import job name] (データセットのインポートジョブ名) で、インポートジョブに名前を付けます。
8. 「データインポートソース」で、Amazon Simple Storage Service (S3) にデータを保存する場所を指定します。次の構文を使用します。

   ```
   s3://<name of your S3 bucket>/<folder path>/<CSV filename>
   ```

9. [IAM ロール] の [IAM サービスロール] で [カスタム IAM ロールの ARN の入力] を選択し、「Amazon Personalize 向けの IAM ロールの作成」で作成したロールの Amazon リソースネーム (ARN) を入力します。画面の表示は次のようになります。

Configure item interactions dataset import job Info

Dataset import job details

Data import source

- (●) Import data from S3
  Specify the location where your data is stored in S3.
- ( ) Incrementally import data with APIs
  Incrementally import item interactions data with the event ingestion SDK.

Dataset import job name
The name you enter here can help you distinguish this dataset import job from others.

my-dataset-import-job-name

The dataset import job name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Data import source

Additional S3 bucket policy required
In addition to the IAM service role defined above, Amazon Personalize also requires you to add a bucket policy to the S3 bucket containing your data files so that it can process them. Follow the instructions described here to add the required bucket policy to your S3 bucket.

Data location Info
Choose the S3 location of your data.

s3://bucket/path-to-your-data/

Your file needs to be in a CSV format and reflect the schema.

IAM Role

IAM service role
Amazon Personalize requires permissions to access your S3 bucket. Choose an existing role with access or create a role in the IAM console with the AmazonPersonalizeFullAccess IAM policy attached.

Enter a custom IAM role ARN ▼

Custom IAM role ARN

arn:aws:iam::YourAccountID:role/YourRole

After you import data from S3, you can still incrementally import data with the Amazon Personalize console, the [redacted] Command Line Interface ([redacted] CLI), or the [redacted] SDKs.

10. インポートの開始を選択してデータをインポートします。ドメインデータセットグループの [Overview] (概要) のページが表示されます。[Set up datasets] (データセットを設定) のセクションのインポートのステータスに注意してください。ステータスが `Interaction data active` になったら、ステップ 3: レコメンダーを作成する に進みます。

ステップ 3: レコメンダーを作成する

この手順では、VIDEO_ON_DEMAND ドメインの [Top picks for you] (上位のおすすめ) のユースケース向けにレコメンダーを作成します。

レコメンダーを作成するには

1. ドメインデータセットグループの [概要] ページの [ステップ 3] で [ビデオオンデマンドのレコメンデーションを使用] タブを選択して、[レコメンダーを作成] を選択します。
2. ユースケースの選択ページで、上位のおすすめを選択し、レコメンダー名 を指定します。画面の表示は次のようになります。

Choose use case Info

You use recommenders to get recommendations for specific e-commerce use cases. Amazon Personalize trains the models backing each recommender with the optimal configurations for these use cases.

Video on demand recommenders

Video on demand use cases to create recommenders for
Amazon Personalize will create a recommender for each selected use case.

- [ ] Because you watched X
  Get recommendations for videos that other users also watched based on a video you specify.
  Recommender name
  my-recommender-name
  The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

- [ ] More like X
  Get recommendations for videos that are similar to a video you specify.
  Recommender name
  my-recommender-name
  The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

- [ ] Most popular
  Get recommendations for videos that have been watched by the most users.
  Recommender name
  my-recommender-name
  The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

- [x] Top picks for you
  Get personalized content recommendations for a user you specify.
  Recommender name
  my-recommender-name
  The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

- [ ] Trending now - *new*
  Get recommendations for videos that are trending now.
  Recommender name
  my-recommender-name
  The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Cancel Next

3. [次へ] をクリックします。
4. 詳細設定ページのフィールドは変更せずに、次へ を選択します。
5. レコメンダーの詳細を確認し、レコメンダーの作成を選択してレコメンダーを作成します。

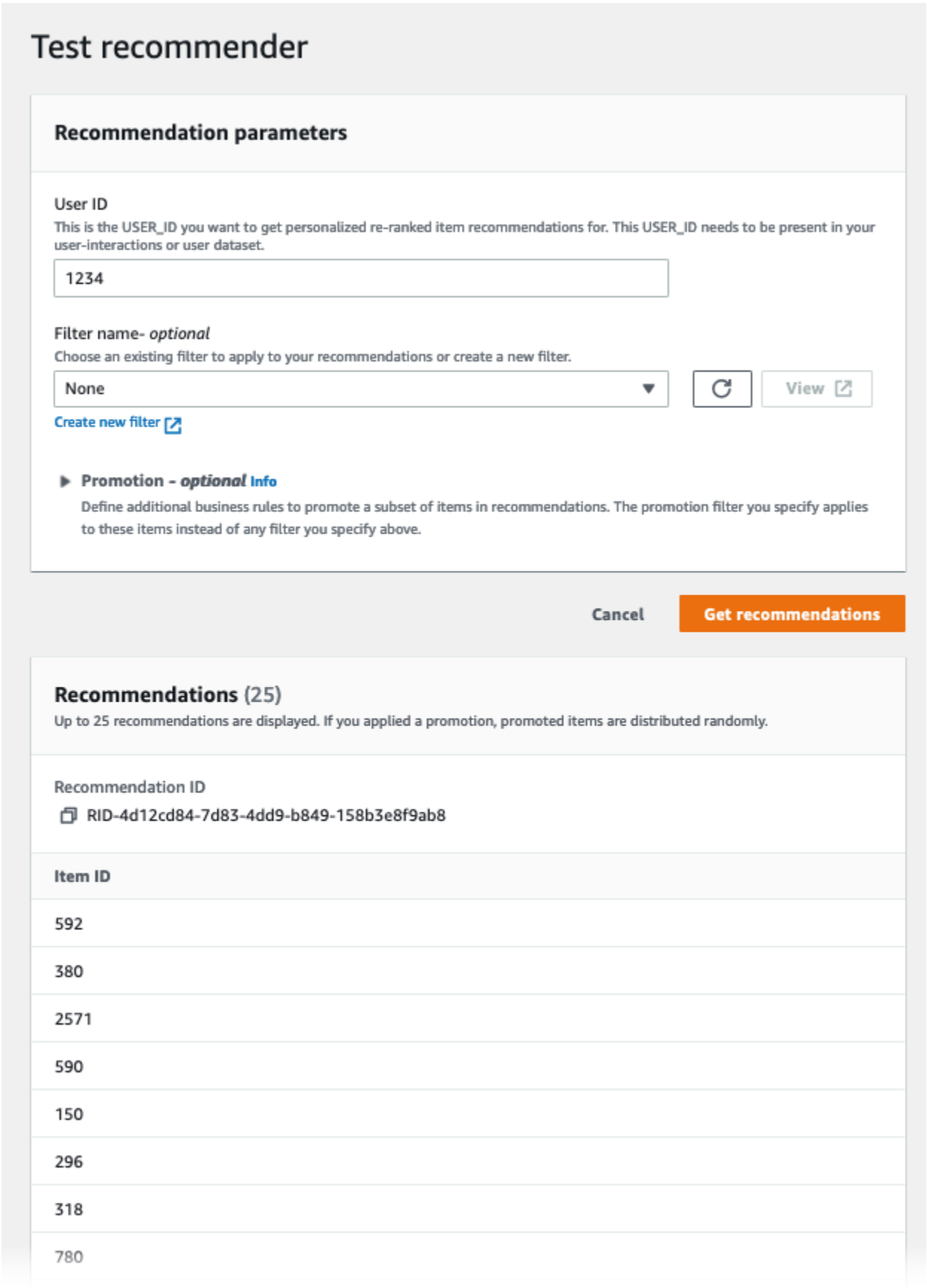
   各レコメンダーのステータスは、[Recommenders] (レコメンダー) のページでモニタリングできます。レコメンダーステータスがアクティブ の場合、それを使用して でレコメンデーションを取得できますステップ 4: レコメンデーションを取得する。

ステップ 4: レコメンデーションを取得する

この手順では、前のステップで作成したレコメンダーを使用して、レコメンデーションを取得します。

推奨事項を取得するには

1. ドメインデータセットグループの [概要] ページの [ナビゲーション] ペインで、[レコメンダー] を選択します。
2. [レコメンダー] ページで、レコメンダーを選択します。
3. 右上で、テスト を選択します。
4. [レコメンデーションパラメータ] にユーザー ID を入力します。他のフィールドはそのままにしておきます。
5. [レコメンデーションの取得] を選択します。ユーザーの上位 25 個の推奨アイテムを含むテーブルが表示されます。画面の表示は次のようになります。


Test recommender
Recommendation parameters
User ID
This is the USER_ID you want to get personalized re-ranked item recommendations for. This USER_ID needs to be present in your user-interactions or user dataset.
1234
Filter name- optional
Choose an existing filter to apply to your recommendations or create a new filter.
None
View
Create new filter
Promotion - optional Info
Define additional business rules to promote a subset of items in recommendations. The promotion filter you specify applies to these items instead of any filter you specify above.
Cancel
Get recommendations
Recommendations (25)
Up to 25 recommendations are displayed. If you applied a promotion, promoted items are distributed randomly.
Recommendation ID
RID-4d12cd84-7d83-4dd9-b849-158b3e8f9ab8
Item ID
592
380
2571
590
150
296
318
780

ドメインデータセットグループの開始方法 (SDK for Java 2.x)

このチュートリアルでは、SDK for Java 2.x を使用して VIDEO_ON_DEMAND ドメインのドメインデータセットグループを作成する方法を示します。このチュートリアルでは、上位のおすすめユースケースのレコメンダーを作成します。

不要な料金が発生しないように、開始方法の演習が終了したら、リソースのクリーンアップ の手順に従って、チュートリアルで作成したリソースを削除してください。

前提条件

このチュートリアルを完了するための前提条件となる手順は次のとおりです。

- 開始方法の前提条件 を完了して必要な権限を設定し、トレーニングデータを作成します。ドメインデータセットグループの開始方法 (コンソール) も完了していれば、同じソースデータを再利用できます。独自のソースデータを使用する場合は、前提条件に示しているようにそのデータがフォーマットされていることを確認します。
- SDK for Java 2.x 環境と AWS 認証情報を、「 AWS SDK for Java 2.x デベロッパーガイド」の「セットアップ AWS SDK for Java 2.x」の手順で指定されているとおりに設定します。

チュートリアル

次のステップでは、Amazon Personalize パッケージを使用するようにプロジェクトを設定し、Amazon Personalize の SDK for Java 2.x クライアントを作成します。次に、データをインポートし、上位のおすすめユースケースのレコメンダーを作成して、レコメンデーションを取得します。

ステップ 1: Amazon Personalize のパッケージを使用するようにプロジェクトを設定する

前提条件を完了したら、Amazon Personalize の依存関係を pom.xml ファイルに追加し、Amazon Personalize のパッケージをインポートします。

1. 次の依存関係を pom.xml ファイルに追加します。最新のバージョン番号は、サンプルコードとは異なる場合があります。

```
<dependency>
 <groupId>software.amazon.awssdk</groupId>
 <artifactId>personalize</artifactId>
 <version>2.16.83</version>
</dependency>
<dependency>
```

```
 <groupId>software.amazon.awssdk</groupId>
 <artifactId>personalizeruntime</artifactId>
 <version>2.16.83</version>
</dependency>
<dependency>
 <groupId>software.amazon.awssdk</groupId>
 <artifactId>personalizeevents</artifactId>
 <version>2.16.83</version>
</dependency>
```

2. プロジェクトに次のインポートステートメントを追加します。

```
// import client packages
import software.amazon.awssdk.services.personalize.PersonalizeClient;
import software.amazon.awssdk.services.personalizeruntime.PersonalizeRuntimeClient;
// Amazon Personalize exception package
import software.amazon.awssdk.services.personalize.model.PersonalizeException;
// schema packages
import software.amazon.awssdk.services.personalize.model.CreateSchemaRequest;
// dataset group packages
import software.amazon.awssdk.services.personalize.model.CreateDatasetGroupRequest;
import software.amazon.awssdk.services.personalize.model.DescribeDatasetGroupRequest;
// dataset packages
import software.amazon.awssdk.services.personalize.model.CreateDatasetRequest;
// dataset import job packages
import
 software.amazon.awssdk.services.personalize.model.CreateDatasetImportJobRequest;
import software.amazon.awssdk.services.personalize.model.DataSource;
import software.amazon.awssdk.services.personalize.model.DatasetImportJob;
import
 software.amazon.awssdk.services.personalize.model.DescribeDatasetImportJobRequest;
// recommender packages
import software.amazon.awssdk.services.personalize.model.CreateRecommenderRequest;
import software.amazon.awssdk.services.personalize.model.CreateRecommenderResponse;
import software.amazon.awssdk.services.personalize.model.DescribeRecommenderRequest;
// get recommendations packages
import
 software.amazon.awssdk.services.personalizeruntime.model.GetRecommendationsRequest;
import
 software.amazon.awssdk.services.personalizeruntime.model.GetRecommendationsResponse;
import software.amazon.awssdk.services.personalizeruntime.model.PredictedItem;
// Java time utility package
import java.time.Instant;
```

ステップ 2: Amazon Personalize のクライアントを作成する

Amazon Personalize の依存関係を pom.xml ファイルに追加し、必要なパッケージをインポートして、次の Amazon Personalize のクライアントをインスタンス化します。

```
PersonalizeClient personalizeClient = PersonalizeClient.builder()
  .region(region)
  .build();

PersonalizeRuntimeClient personalizeRuntimeClient = PersonalizeRuntimeClient.builder()
  .region(region)
  .build();
```

ステップ 3: データをインポートする

Amazon Personalize のクライアントを初期化したら、開始方法の前提条件 の完了時に作成した履歴データをインポートします。過去のデータを Amazon Personalize にインポートするには、次の操作を実行します。

1. 次の Avro スキーマを JSON ファイルとして作業ディレクトリに保存します。このスキーマは、トレーニングデータの作成 (ドメインデータセットグループ) を完了したときに作成した CSV ファイルの列と一致します。

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
```

```
            "type": "long"
        }
    ],
    "version": "1.0"
}
```

2. 次の createDomainSchema メソッドを使用して、Amazon Personalize でドメインスキーマを作成します。パラメータとして、Amazon Personalize のサービスクライアント、スキーマの名前、ドメイン用のVIDEO_ON_DEMANDおよび前のステップで作成したスキーマ JSON ファイルのファイルパスを渡します。このメソッドは、新しいスキーマの Amazon リソースネーム (ARN) を返します。後で使用するために、これを保存します。

```
    public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
            String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .domain(domain)
                    .schema(schema)
                    .build();

            String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
```

```
    }
```

3. データセットグループを作成します。次の createDomainDatasetGroup メソッドを使用して、ドメインデータセットグループを作成します。パラメータとして、Amazon Personalize のサービスクライアントとデータセットグループの名前を渡し、ドメイン用の VIDEO_ON_DEMAND を渡します。このメソッドは、新しいデータセットグループの ARN を返します。後で使用するために、これを保存します。

```
   public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
           String datasetGroupName,
           String domain) {

       try {
           CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                   .name(datasetGroupName)
                   .domain(domain)
                   .build();
           return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
       } catch (PersonalizeException e) {
           System.out.println(e.awsErrorDetails().errorMessage());
       }
       return "";
   }
```

4. アイテムインタラクションデータセットを作成します。次の createDataset メソッドを使用して、アイテムインタラクションデータセットを作成します。パラメータとして、Amazon Personalize のサービスクライアント、データセットの名前、スキーマの ARN、データセットグループの ARN、およびデータセットタイプの Interactions を渡します。このメソッドは、新しいデータセットの ARN を返します。後で使用するために、これを保存します。

```
   public static String createDataset(PersonalizeClient personalizeClient,
           String datasetName,
           String datasetGroupArn,
           String datasetType,
           String schemaArn) {
       try {
           CreateDatasetRequest request = CreateDatasetRequest.builder()
                   .name(datasetName)
                   .datasetGroupArn(datasetGroupArn)
```

```
                    .datasetType(datasetType)
                    .schemaArn(schemaArn)
                    .build();

            String datasetArn = personalizeClient.createDataset(request)
                    .datasetArn();
            System.out.println("Dataset " + datasetName + " created.");
            return datasetArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

5. データセットのインポートジョブを使用してデータをインポートします。次の `createPersonalizeDatasetImportJob` メソッドを使用して、データセットのインポートジョブを作成します。

   パラメータとして以下を渡します: Amazon Personalize のサービスクライアント、ジョブ名、およびインタラクションデータセット ARN。トレーニングデータを保存した Amazon S3 バケットパス (s3://*bucket name*/*folder name*/ratings.csv) とサービスロールの ARN を渡します。このロールは 開始方法の前提条件 の一部として作成しました。このメソッドは、データセットのインポートジョブの ARN を返します。オプションで、後で使用するために保存します。

```
    public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
            String jobName,
            String datasetArn,
            String s3BucketPath,
            String roleArn) {

        long waitInMilliseconds = 60 * 1000;
        String status;
        String datasetImportJobArn;

        try {
            DataSource importDataSource = DataSource.builder()
                    .dataLocation(s3BucketPath)
                    .build();
```

```
            CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                    .datasetArn(datasetArn)
                    .dataSource(importDataSource)
                    .jobName(jobName)
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient
                        .describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

ステップ 4: レコメンダーを作成する

データセットのインポートジョブが完了すると、レコメンダーを作成する準備が整います。次の `createRecommender` メソッドを使用して、レコメンダーを作成します。パラメータとして、Amazon Personalize のサービスクライアント、データセットグループの Amazon リソースネーム (ARN)、`arn:aws:personalize:::recipe/aws-vod-top-picks` 用のレシピ ARN を渡します。このメソッドは、新しいレコメンダーの ARN を返します。後で使用するために、これを保存します。

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
```

```
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

ステップ 5: レコメンデーションを取得する

レコメンダーを作成したら、それを使用してレコメンデーションを取得します。次の getRecs メソッドを使用して、ユーザー向けのレコメンデーションを取得します。パラメータとして、Amazon Personalize のランタイムクライアント、前のステップで作成したレコメンダーの Amazon リソースネーム (ARN)、およびユーザー ID (例: 123) を渡します。このメソッドは、推奨アイテムのリストを画面に出力します。

```
    public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
            String userId) {

        try {
            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                    .recommenderArn(recommenderArn)
                    .numResults(20)
                    .userId(userId)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
```

```
                .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

ドメインデータセットグループの開始方法

このチュートリアルでは、SDK for Python (Boto3) を使用して VIDEO_ON_DEMAND ドメイン用のドメインデータセットグループを作成する方法を示します。このチュートリアルでは、「上位のおすすめ」ユースケースのレコメンダーを作成します。

不要な料金が発生しないように、開始方法の演習が終了したら、チュートリアルで作成したリソースを削除してください。詳細については、「リソースのクリーンアップ」を参照してください。

トピック

- 前提条件
- チュートリアル
- Jupyter (iPython) ノートブックでの Amazon Personalize API の開始方法

前提条件

このガイドで Python の例を使用するための前提条件のステップを以下に示します。

- 開始方法の前提条件 を完了して必要な権限を設定し、トレーニングデータを作成します。独自のソースデータを使用する場合は、前提条件に示しているようにそのデータがフォーマットされていることを確認します。
- 「AWS SDKsのセットアップ」に指定されているように、AWS SDK for Python (Boto3) 環境を設定します。

チュートリアル

次のステップでは、環境を確認し、Amazon Personalize の Python (Boto3) クライアント用 Python (Boto3) クライアントを作成します。次に、データをインポートし、「上位のおすすめ」ユースケースのレコメンダーを作成して、レコメンデーションを取得します。

ステップ 1: Python 環境を確認し、boto3 クライアントを作成する

前提条件のステップを完了したら、以下の Python の例を実行して、環境が適切に設定されていることを確認します。このコードでは、このチュートリアルで使用する Amazon Personalize boto3 クライアントも作成されます。環境が正しく設定されている場合、使用可能なレシピのリストが表示され、このガイドの他の Python の例を実行できます。

```
import boto3

personalizeRt = boto3.client('personalize-runtime')
personalize = boto3.client('personalize')

response = personalize.list_recipes()

for recipe in response['recipes']:
    print (recipe)
```

ステップ 2: データをインポートする

Amazon Personalize boto3 クライアントを作成して環境を検証したら、開始方法の前提条件 の完了時に作成した履歴データをインポートします。過去のデータを Amazon Personalize にインポートするには、次の操作を実行します。

1. 以下のコードを使用して、Amazon Personalize でスキーマを作成します。`gs-domain-interactions-schema` を、スキーマの名前に置き換えます。

```
import json
schema = {
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
```

```
        },
        {
            "name": "ITEM_ID",
            "type": "string"
        },
        {
            "name": "EVENT_TYPE",
            "type": "string"
        },
        {
            "name": "TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}

create_interactions_schema_response = personalize.create_schema(
    name='gs-domain-interactions-schema',
    schema=json.dumps(schema),
    domain='VIDEO_ON_DEMAND'
)

interactions_schema_arn = create_interactions_schema_response['schemaArn']
print(json.dumps(create_interactions_schema_response, indent=2))
```

2. 次のコードを使用してデータセットグループを作成します。dataset group name をデータセットグループの名前に置き換えます。

```
response = personalize.create_dataset_group(
    name = 'dataset group name',
    domain = 'VIDEO_ON_DEMAND'
)
dsg_arn = response['datasetGroupArn']

description = personalize.describe_dataset_group(datasetGroupArn = dsg_arn)
['datasetGroup']

print('Name: ' + description['name'])
print('ARN: ' + description['datasetGroupArn'])
print('Status: ' + description['status'])
```

3. 次のコードを使用して、新しいデータセットグループにアイテムインタラクションデータセットを作成します。データセットに名前を付け、前のステップの schema_arn および dataset_group_arn を指定します。

```
response = personalize.create_dataset(
    name = 'interactions-dataset-name',
    schemaArn = interactions_schema_arn,
    datasetGroupArn = dsg_arn,
    datasetType = 'INTERACTIONS'
)

dataset_arn = response['datasetArn']
```

4. 以下のコードとともにデータセットのインポートジョブを使用してデータをインポートします。このコードでは describe_dataset_import_job メソッドを使用してジョブのステータスを追跡します。

   ジョブの名前、前のステップの dataset_arn、トレーニングデータおよび IAM サービスロール ARN を格納した Amazon S3 バケットパス (s3://*bucket name*/*folder name*/ratings.csv) をパラメータに渡します。このロールは 開始方法の前提条件 の一部として作成しました。Amazon Personalize には、バケットにアクセスするための許可が必要です。アクセス権限の付与については、「Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与」を参照してください。

```
import time
response = personalize.create_dataset_import_job(
    jobName = 'JobName',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn'
)

dataset_interactions_import_job_arn = response['datasetImportJobArn']

description = personalize.describe_dataset_import_job(
    datasetImportJobArn = dataset_interactions_import_job_arn)['datasetImportJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['datasetImportJobArn'])
print('Status: ' + description['status'])
```

```
max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:
    describe_dataset_import_job_response = personalize.describe_dataset_import_job(
        datasetImportJobArn = dataset_interactions_import_job_arn
    )
    status = describe_dataset_import_job_response["datasetImportJob"]['status']
    print("Interactions DatasetImportJob: {}".format(status))

    if status == "ACTIVE" or status == "CREATE FAILED":
        break

    time.sleep(60)
```

ステップ 4: レコメンダーを作成する

データセットのインポートジョブが完了すると、レコメンダーを作成する準備が整います。次のコードを使用してレコメンダーを取得します。以下をパラメータとして渡します: レコメンダーの名前、データセットグループの Amazon リソースネーム (ARN)、レシピ ARN の`arn:aws:personalize:::recipe/aws-vod-top-picks` このコードは、describe_recommendations der メソッドを使用してレコメンダーのステータスを追跡します。

```
import time
create_recommender_response = personalize.create_recommender(
  name = 'gs-python-top-picks',
  recipeArn = 'arn:aws:personalize:::recipe/aws-vod-top-picks',
  datasetGroupArn = dsg_arn
)
recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:

    version_response = personalize.describe_recommender(
        recommenderArn = recommender_arn
    )
    status = version_response["recommender"]["status"]

    if status == "ACTIVE":
        print("Creation succeeded for {}".format(recommender_arn))

    elif status == "CREATE FAILED":
```

```
        print("Creation failed for {}".format(recommender_arn))

    if status == "ACTIVE":
        break
    else:
        print("Recommender creation is still in progress")

    time.sleep(60)
```

ステップ 5: レコメンデーションを取得する

レコメンダーを作成したら、それを使用して次のコードでレコメンデーションを取得します。パラメータとして、前のステップで作成したレコメンダーの Amazon リソースネーム (ARN) とユーザー ID (例: 123) を渡します。このメソッドは、推奨事項のリストを画面に出力します。

```
response = personalizeRt.get_recommendations(
    recommenderArn = "arn:aws:personalize:us-west-2:014025156336:recommender/gs-python-
top-picks-89",
    userId = '123'
)
print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
```

Jupyter (iPython) ノートブックでの Amazon Personalize API の開始方法

Jupyter Notebook を使用してドメインデータセットグループの作成を開始するには、Amazon Personalize サンプルリポジトリの notebooks_managed_domains フォルダにある一連のノートブックのクローンを作成するかダウンロードします。ノートブックでは、トレーニングデータのインポート、レコメンダーの作成、Amazon Personalize でのレコメンデーションの取得について説明します。

> **Note**
>
> ノートブックを使用する前に、README.md の手順に従って環境を構築してください。

ドメインデータセットグループの開始方法 (SDK for JavaScript v3)

このチュートリアルでは、AWS SDK for JavaScript v3 を使用して VIDEO_ON_DEMAND ドメインのドメインデータセットグループを作成する方法を示します。このチュートリアルでは、Top Picks for You ユースケースのレコメンダーを作成します。

このチュートリアルで使用されているコードを GitHub で確認するには、AWS SDK コードサンプルリポジトリの「JavaScript v3 用 SDK 用の Amazon Personalize コードサンプル」を参照してください。

不要な料金が発生しないように、開始方法の演習が終了したら、チュートリアルで作成したリソースを削除してください。詳細については、「リソースのクリーンアップ」を参照してください。

トピック

前提条件

このチュートリアルを完了するための前提条件となる手順は次のとおりです。

- 開始方法の前提条件 を完了して必要な権限を設定し、トレーニングデータを作成します。ドメインデータセットグループの開始方法 (コンソール) も完了していれば、同じソースデータを再利用できます。独自のソースデータを使用する場合は、前提条件に示しているようにそのデータがフォーマットされていることを確認します。
- AWS SDK for JavaScript デベロッパーガイドの「SDK for JavaScript の設定」の手順で指定されているように、AWS 認証情報を設定します。

チュートリアル

次の手順では、必要な依存関係をインストールします。次に、データセットグループを作成し、データをインポートし、ユースケースの上位のおすすめ向けのレコメンダーを作成して、レコメンデーションを取得します。

Node.js を使用する場合は、サンプルを JavaScript ファイルとして保存してから実行することで、`node <fileName.js>` 実行できます。

ステップ 1: Amazon Personalize の依存関係をインストールする

前提条件を満たしたら、Amazon Personalize の以下の依存関係をインストールします。

- @aws-sdk/client-personalize
- @aws-sdk/client-personalize-runtime
- @aws-sdk/client-personalize-events (このチュートリアルではオプションですが、レコメンダーを作成した後にイベントを記録する場合は必須)

次は、使用できる `package.json` ファイルの例です。Node.js を使用して依存関係をインストールするには、`package.json` ファイルを保存した場所に移動して `npm install` を実行します。

```
{
  "name": "personalize-js-project",
  "version": "1.0.0",
  "description": "personalize operations",
  "type": "module",
  "author": "Author Name <email@address.com>",
  "license": "ISC",
  "dependencies": {
    "@aws-sdk/client-personalize": "^3.350.0",
    "@aws-sdk/client-personalize-events": "^3.350.0",
    "@aws-sdk/client-personalize-runtime": "^3.350.0",
    "fs": "^0.0.1-security"
  },
  "compilerOptions": {
    "resolveJsonModule": true,
    "esModuleInterop": true
  }
}
```

ステップ 2: Amazon Personalize のクライアントを作成する

依存関係をインストールしたら、Amazon Personalize クライアントを作成します。このチュートリアルでは、`libs` というディレクトリに保存された `personalizeClients.js` という名前のファイルにクライアントを作成することを前提としています。

次は、`personalizeClient.js` ファイルの例です。

```
import { PersonalizeClient } from "@aws-sdk/client-personalize";
import { PersonalizeRuntimeClient } from "@aws-sdk/client-personalize-runtime";
```

```
import { PersonalizeEventsClient } from "@aws-sdk/client-personalize-events";
// Set your AWS region.
const REGION = "region"; //e.g. "us-east-1"

const personalizeClient = new PersonalizeClient({ region: REGION});
const personalizeEventsClient = new PersonalizeEventsClient({ region: REGION});
const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: REGION});

export { personalizeClient, personalizeEventsClient, personalizeRuntimeClient };
```

ステップ 3: データをインポートする

Amazon Personalize のクライアントを作成したら、開始方法の前提条件 の完了時に作成した履歴データをインポートします。過去のデータを Amazon Personalize にインポートするには、次の操作を実行します。

1. 次の Avro スキーマを JSON ファイルとして作業ディレクトリに保存します。このスキーマは、トレーニングデータの作成 (ドメインデータセットグループ) を完了時に作成した CSV ファイルの列と一致します。

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
```

```
}
```

2. 以下の createDomainSchema.js コードを使用して Amazon Personalize でドメインスキーマを作成します。SCHEMA_PATH を、先ほど作成した schema.json へのパスに置き換えます。createSchemaParam を更新してスキーマの名前を指定し、domain を VIDEO_ON_DEMAND と指定してください。

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = 'TEST' // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema, /* required */
  domain: 'DOMAIN'   /* required for a domain dataset group, specify ECOMMERCE or
 VIDEO_ON_DEMAND */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSchemaCommand(createDomainSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

3. 次のcreateDomainDatasetGroup.jsコードを使用して、Amazon Personalize コンソールにドメインデータセットグループを作成します。domainDatasetGroupParams を更新してデータセットグループの名前を指定し、domain を VIDEO_ON_DEMAND と指定してください。

```
// Get service clients module and commands using ES6 syntax.
import {  CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: 'NAME',  /* required */
  domain: 'DOMAIN'   /* required for a domain dsg, specify ECOMMERCE or
 VIDEO_ON_DEMAND */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetGroupCommand(domainDatasetGroupParams));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

4. 次の createDataset.js コードを使用して Amazon Personalize でアイテムインタラクションデータセットを作成します。createDatasetParam を更新して、先ほど作成したデータセットグループとスキーマの Amazon リソースネーム (ARN) を指定し、データセットに名前を付け、datasetType を Interactions と指定してください。

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from
  "@aws-sdk/client-personalize";
```

```
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  datasetType: 'DATASET_TYPE', /* required */
  name: 'NAME', /* required */
  schemaArn: 'SCHEMA_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetCommand(createDatasetParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

5. 次の createDatasetImportJob.js コードを使用してデータをインポートします。datasetImportJobParam を更新して次のように指定します。
   - ジョブの名前を指定し、インタラクションデータセットの ARN を指定します。
   - dataLocation を指定し、トレーニングデータを保存した Amazon S3 バケットへのパスを指定します。
   - roleArn には、Amazon Personalize サービスロールの Amazon リソースネームを指定してください。このロールは 開始方法の前提条件 の一部として作成しました。

```
// Get service clients module and commands using ES6 syntax.
import {CreateDatasetImportJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
```

```
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: {  /* required */
    dataLocation: 'S3_PATH'
  },
  jobName: 'NAME',/* required */
  roleArn: 'ROLE_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

ステップ 4: レコメンダーを作成する

データセットのインポートジョブが完了すると、レコメンダーを作成する準備が整います。レコメンダーを作成するには、次の createRecommender.js コードを使用します。createRecommenderParam を次のように更新します: レコメンダーの名前を指定し、データセットグループの ARN を指定して、recipeArn に対して arn:aws:personalize:::recipe/aws-vod-top-picks を指定してください。

```
// Get service clients module and commands using ES6 syntax.
import { CreateRecommenderCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: 'NAME', /* required */
  recipeArn: 'RECIPE_ARN', /* required */
  datasetGroupArn: 'DATASET_GROUP_ARN'  /* required */
```

```
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

ステップ 5: レコメンデーションを取得する

キャンペーンを作成したら、そのキャンペーンを使用してレコメンデーションを取得します。次の getRecommendations.js コードを使用して、ユーザー向けのレコメンデーションを取得します。getRecommendationsParam を更新して前のステップで作成したレコメンダーの ARN を指定し、ユーザ ID (例:123) を指定します。

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: 'RECOMMENDER_ARN', /* required */
  userId: 'USER_ID',       /* required */
  numResults: 15    /* optional */
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
```

```
  }
};
run();
```

カスタムデータセットグループの開始方法

> **Important**
>
> このチュートリアルでは、自動トレーニングを使用するソリューションを作成します。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブになっている間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。詳細については、「リソースのクリーンアップ」を参照してください。

この入門ガイドでは、カスタムデータセットグループとUser-Personalization-v2 レシピレシピを使用して、パーソナライズされた映画のレコメンデーションをユーザーに提供する方法を説明します。このチュートリアルでは、600 名のユーザーからの 9,700 本の映画に対する 100,000 件のレーティングで構成される履歴データを使用します。

開始するには、「開始方法の前提条件」を完了してから、「開始方法 (コンソール)」、「開始方法 (AWS CLI)」、「開始方法 (SDK for Python (Boto3))」、または「開始方法 (SDK for Java 2.x)」のいずれかに進みます。

開始方法の演習を完了したら、不要な料金が発生しないように、作成したリソースを、「リソースのクリーンアップ」の手順に従って削除します。

トピック

- 開始方法 (コンソール)
- 開始方法 (AWS CLI)
- 開始方法 (SDK for Python (Boto3))
- 開始方法 (SDK for Java 2.x)

開始方法 (コンソール)

この演習では、Amazon Personalize コンソールを使用して、特定のユーザー向けに映画のレコメンデーションを返すソリューションを含むカスタムデータセットグループを作成します。この演習を開始する前に、「開始方法の前提条件」を確認してください。

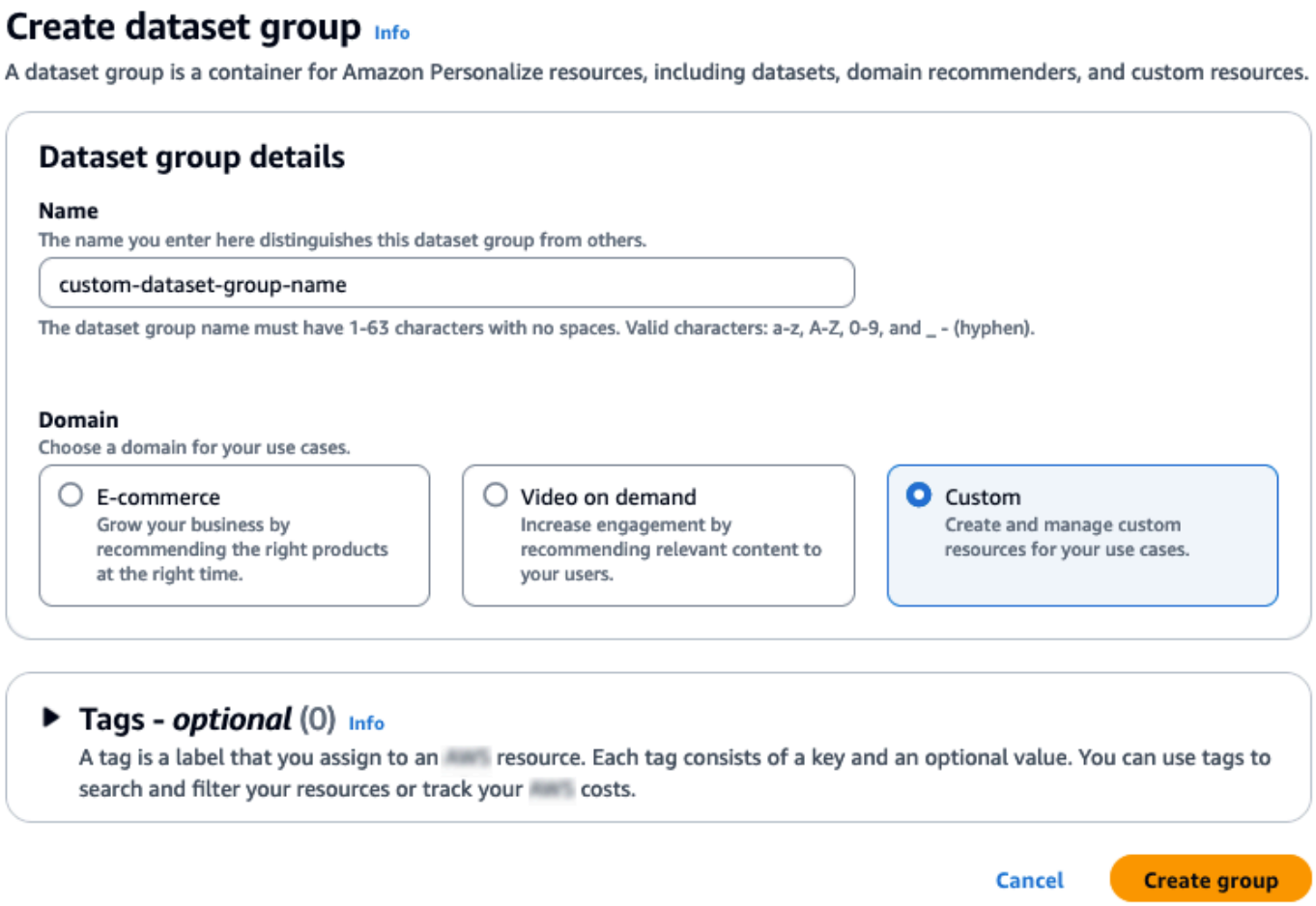
開始方法の演習を完了したら、不要な料金が発生しないように、作成したリソースを、「リソースのクリーンアップ」の手順に従って削除します。

ステップ 1: データセットグループとデータセットを作成する

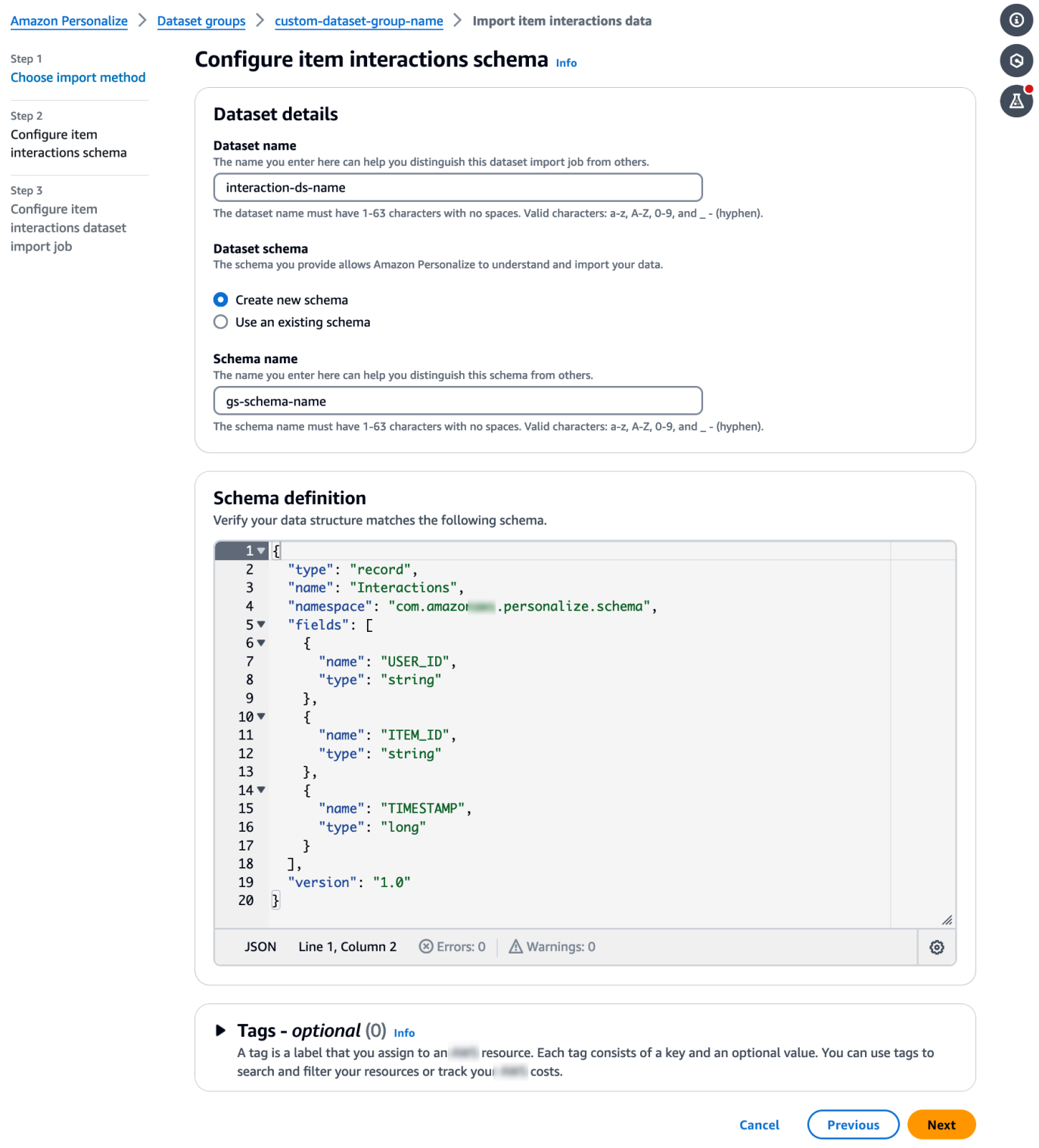
この手順ではまず、データセットグループを作成します。次に、Amazon Personalize のアイテムインタラクションデータセットをデータセットグループに作成します。

データセットグループとデータセットを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループの作成] を選択します。
3. [データセットグループの詳細] で [Dataset group name (データセットグループ名)] にデータセットグループの名前を指定します。
4. [Domain] (ドメイン) で、[Custom] (カスタム) を選択します。画面の表示は次のようになります。



5. [Create group] (グループの作成) を選択します。概要ページが表示されます。
6. ステップ 1。データセットを作成してデータをインポートし、データセットの作成 を選択し、アイテムインタラクションデータセット を選択します。
7. [Amazon Personalize データセットにデータを直接インポートする] を選択し、[次へ] を選択します。
8. [アイテムインタラクションデータセットを設定] ページで、[データセット名] にデータセットの名前を指定します。
9. [スキーマの選択] で、[新しいスキーマの作成] を選択します。スキーマ定義セクションには、最小限のアイテムインタラクションスキーマが表示されます。スキーマは以前に `ratings.csv` ファイルに追加したヘッダーと一致するため、変更を加える必要はありません。トレーニングデータをまだ作成していない場合は、「開始方法の前提条件」を参照してください。
10. [新しいスキーマ名] で、新しいスキーマの名前を指定します。画面の表示は次のようになります。



11. [次へ] をクリックします。[アイテムインタラクションデータセットのインポートジョブの設定] ページが表示されます。次に、「ステップ 2: アイテムインタラクションデータをインポートする」を完了してインタラクションデータのインポートします。

ステップ 2: アイテムインタラクションデータをインポートする

データセットを作成したので、今度はアイテムインタラクションデータをデータセットにインポートします。

アイテムインタラクションデータをインポートするには

1. [アイテムインタラクションデータセットのインポートジョブを設定] ページで、[データのインポートソース] を [S3 からデータをインポート] のままにしておきます。
2. [データセットのインポートジョブ名] で、インポートジョブの名前を指定します。
3. [追加の S3 バケットポリシーが必要です] という名前の情報ダイアログボックスで、Amazon Personalize にアクセス許可を付与していない場合は、指示に従って必要な Amazon S3 バケットポリシーを追加します。
4. [Data location] (データの場所) で、映画のデータファイルが Amazon Simple Storage Service (S3) のどこに保存されるかを指定します。次の構文を使用します。

   ```
   s3://<name of your S3 bucket>/<folder path>/filename.csv
   ```

5. IAM ロール セクションの IAM サービスロール で、カスタム IAM ロール ARN を入力 を選択します。
6. [カスタム IAM ロールの ARN] で、Amazon Personalize 向けの IAM ロールの作成 で作成したロールを指定します。

   [データセットのインポートジョブの詳細] と [IAM ロール]のセクションは、次のようになっているはずです。

Configure item interactions dataset import job Info

Dataset import job details

Data import source

- (●) Import data from S3
  Specify the location where your data is stored in S3.
- ( ) Incrementally import data with APIs
  Incrementally import item interactions data with the event ingestion SDK.

Dataset import job name
The name you enter here can help you distinguish this dataset import job from others.

gs-dataset-import-job

The dataset import job name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Data import source

Additional S3 bucket policy required
In addition to the IAM service role defined above, Amazon Personalize also requires you to add a bucket policy to the S3 bucket containing your data files so that it can process them. Follow the instructions described here to add the required bucket policy to your S3 bucket.

Data location Info
Choose the S3 location of your data.

s3://bucket-name/ratings.csv

Your file needs to be in a CSV format and reflect the schema.

IAM Role

IAM service role
Amazon Personalize requires permissions to access your S3 bucket. Choose an existing role with access or create a role in the IAM console with the AmazonPersonalizeFullAccess IAM policy attached.

Enter a custom IAM role ARN ▼

Custom IAM role ARN

arn:[redacted]:iam::accountNumber:role/roleName

After you import data from S3, you can still incrementally import data with the Amazon Personalize console, the AWS Command Line Interface [redacted] CLI), or the [redacted] SDKs.

Publish event metrics to S3 - *optional*

When you create a metric attribution, reports related to this import job can be published to S3 for analysis with your tool of choice.

To track and publish metrics for events, you must create a metric attribution and define event metrics. Create metric attribution

☐ Publish metrics from this import job (you have not created a metric attribution)

7. イベントメトリクスの発行セクションを S3 およびタグセクションに変更せずに、インポートの開始 を選択します。データインポートジョブが開始され、[ダッシュボード概要] ページが表示されます。初期状態では、状態は作成待機 (続いて作成進行中) であり、リューションの作成は無効になっています。

   データインポートジョブが完了すると、ステータスが [アクティブ] になり [ソリューション作成] ボタンが有効になります。

8. データをインポートし、「ステップ 3: ソリューションを作成する」でソリューションを作成する準備が整いました。

ステップ 3: ソリューションを作成する

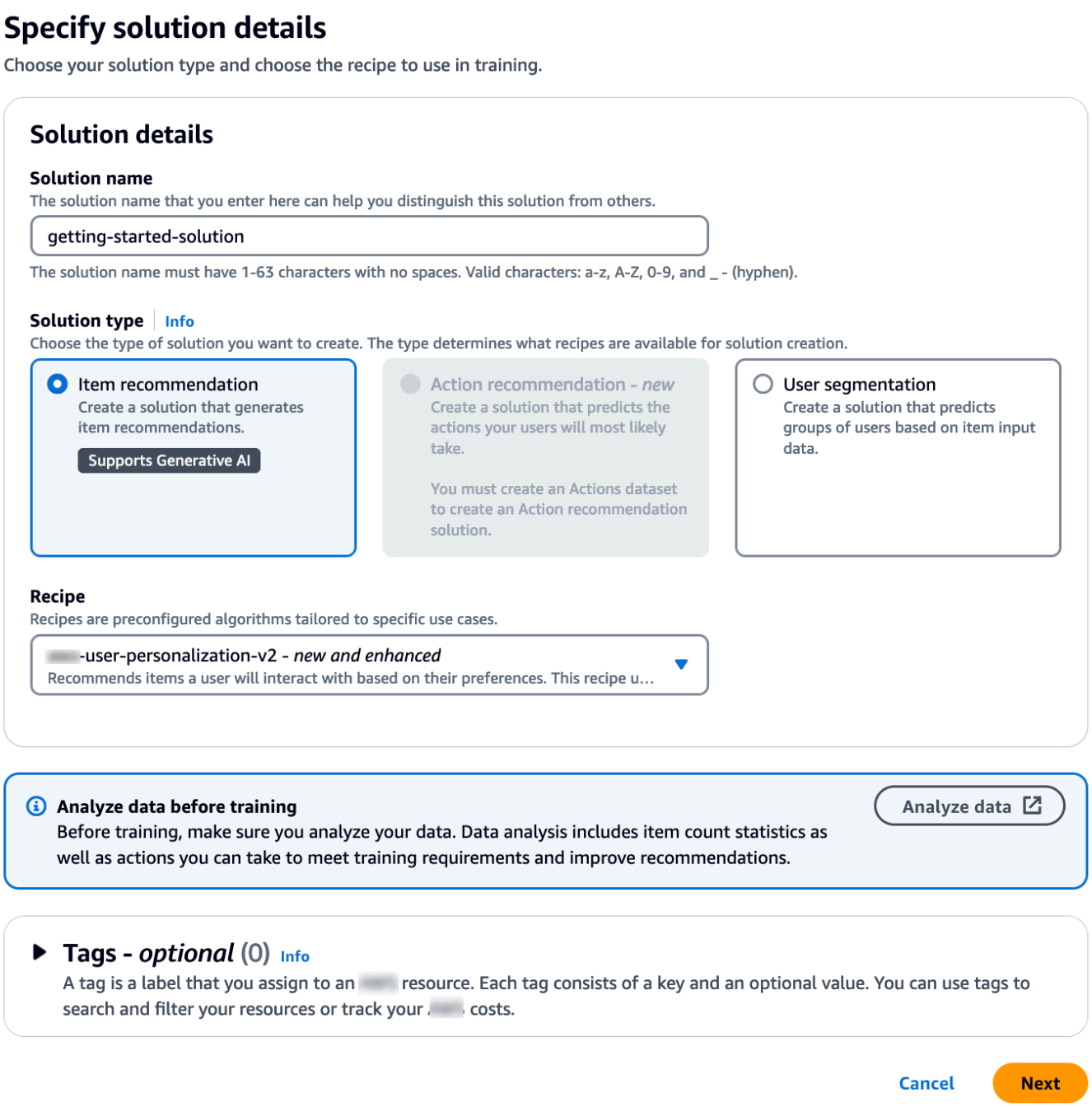
このチュートリアルでは、インポートしたデータセットを使用してモデルをステップ 2: アイテムインタラクションデータをインポートするトレーニングします。トレーニングしたモデルはソリューションバージョンと呼ばれます。

> **Important**
>
> このチュートリアルでは、自動トレーニングを使用するソリューションを作成します。自動トレーニングでは、ソリューションがアクティブになっている間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。詳細については、「リソースのクリーンアップ」を参照してください。

ソリューションを作成するには

1. データセットグループの概要ページのステップ 3。トレーニングリソースとレコメンデーションリソースを設定する ソリューションの作成 を選択します。
2. [Solution name (ソリューション名)] で、ソリューションの名前を指定します。
3. [ソリューションタイプ] には [おすすめアイテム] を選択します。
4. レシピ で、 を選択しますaws-user-personalization-v2。

   画面の表示は次のようになります。



5. [次へ] をクリックします。トレーニング設定フィールドは変更しないでください。作成したソリューションは 7 日ごとに新しいモデルを自動的にトレーニングし、最新のアイテムインタラクションデータに重みを付けます。
6. [次へ] を選択し、ソリューションの詳細を確認します。
7. ソリューションの作成を選択し、ソリューションの詳細ページが表示されます。ソリューションを作成すると、Amazon Personalize は 1 時間以内に最初のソリューションバージョンの作成を

開始します。トレーニングが開始されると、詳細ページのソリューションバージョンセクションに表示され、そのステータスをモニタリングできます。

ソリューションバージョンのステータスが[アクティブ] になったら、ステップ 4: キャンペーンの作成 に移行できます。

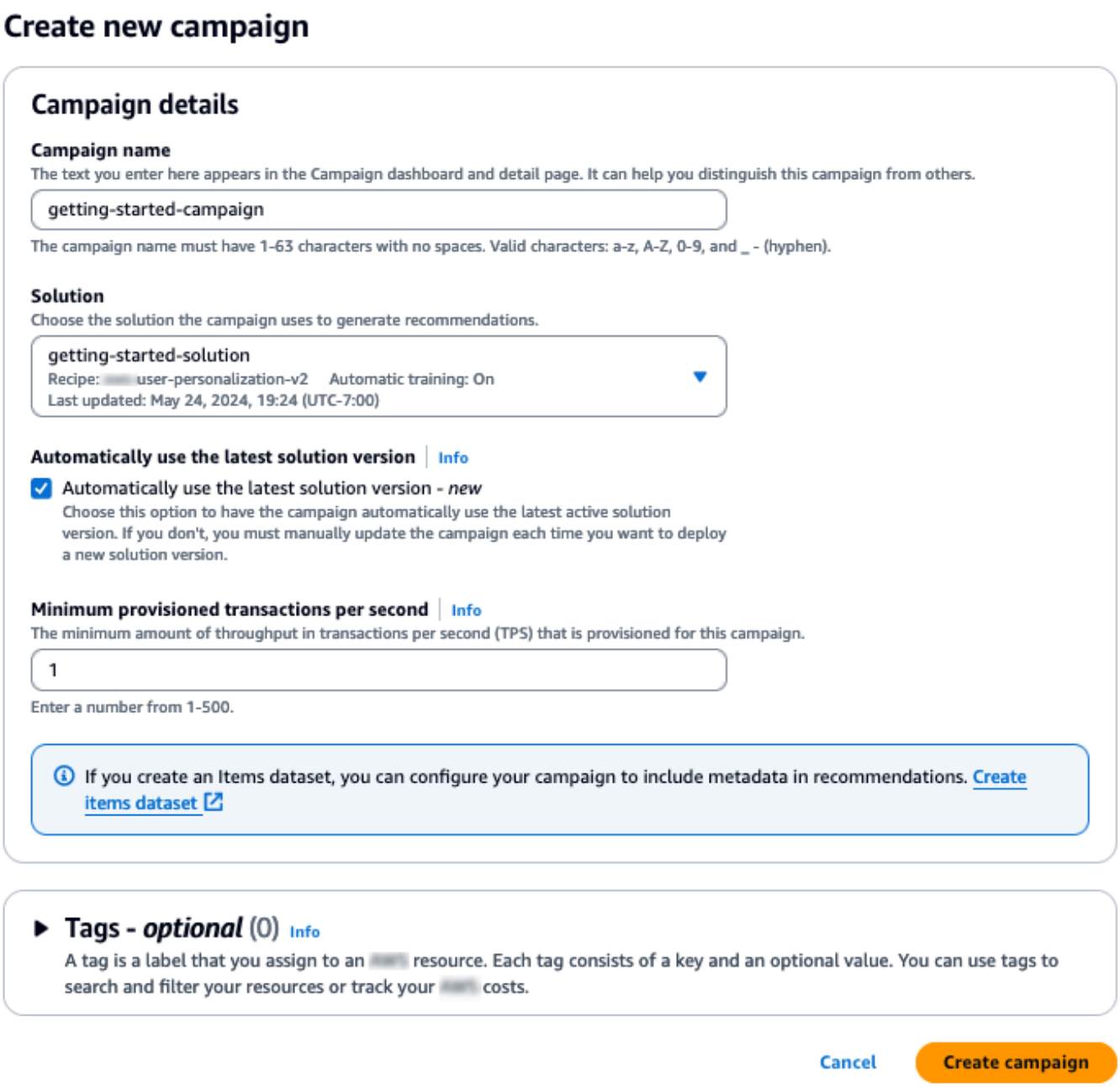
ステップ 4: キャンペーンの作成

この手順では、キャンペーンを作成します。このキャンペーンは、前のステップで作成したソリューションバージョンをデプロイします。

キャンペーンを作成するには

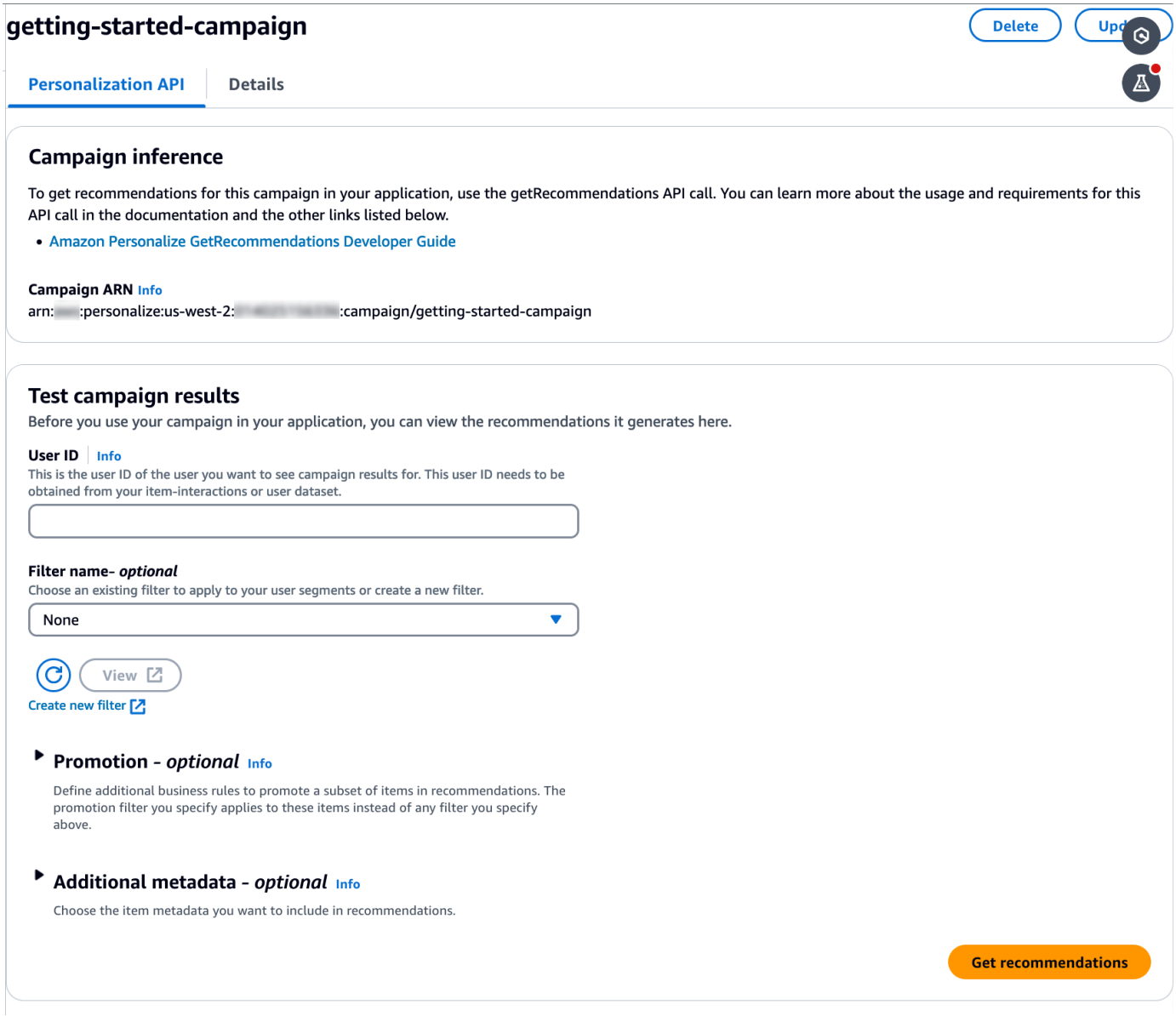
1. ナビゲーションペインで [カスタムリソース] を展開し、[キャンペーン] を選択します。
2. [キャンペーンの作成 を選択します。[新しいキャンペーンを作成] ページが表示されます。
3. [Campaign details (キャンペーンの詳細)] で [Campaign name (キャンペーン名)] にキャンペーンの名前を指定します。
4. ソリューション で、前のステップで作成したソリューションを選択します。
5. 最新のソリューションバージョン を自動的に使用するを選択します。他のすべてのフィールドは変更しないでください。

   画面の表示は次のようになります。



6. [キャンペーンの作成] を選択します。キャンペーンの作成が開始され、キャンペーンの詳細ページに [パーソナライゼーション API] セクションが表示されます。

   キャンペーンの作成には数分かかることがあります。Amazon Personalize がキャンペーンの作成を完了すると、ページが更新され、[Test campaign results] (キャンペーン結果をテスト) のセクションが表示されます。画面の表示は次のようになります。



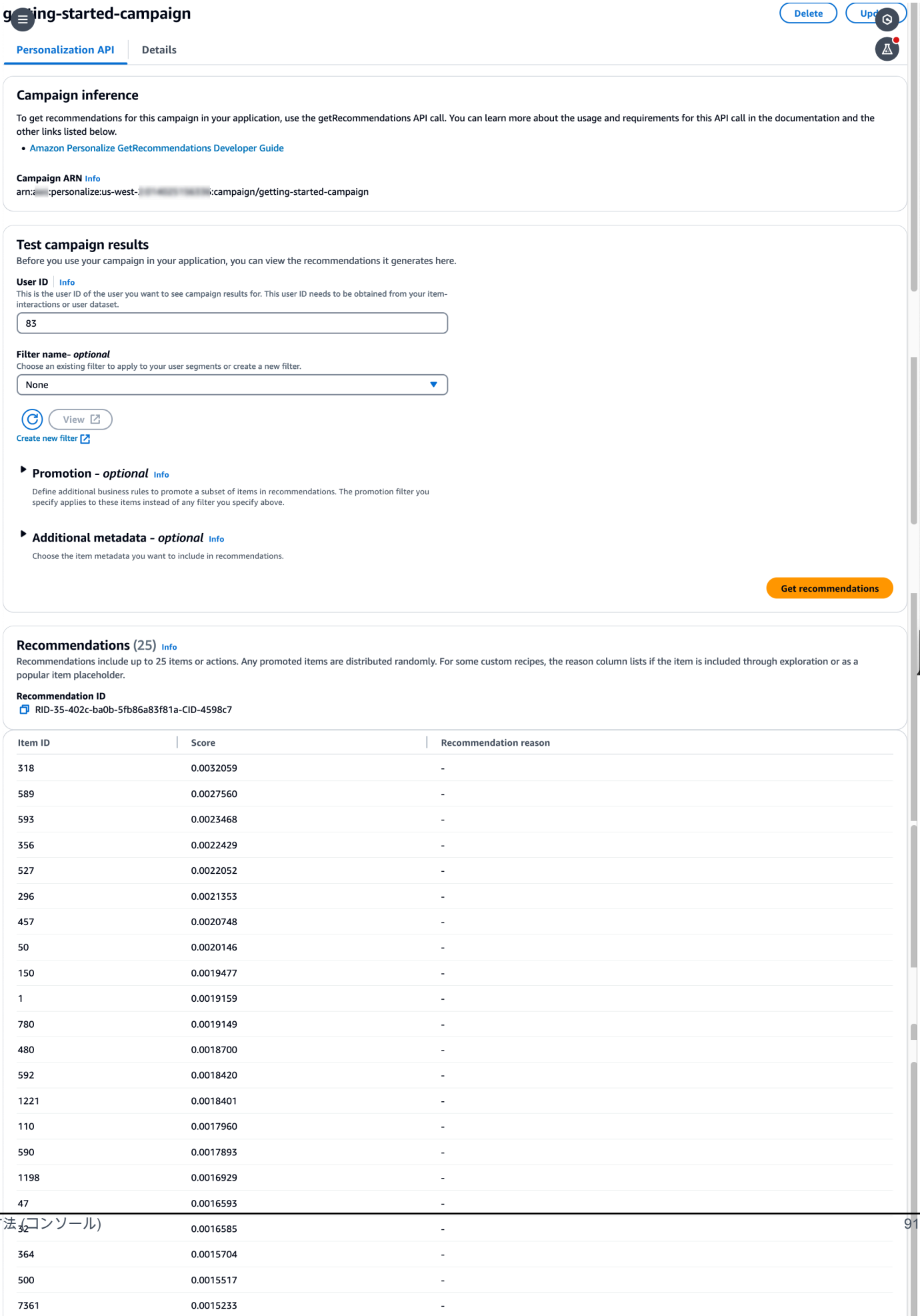
ステップ 5: レコメンデーションを取得する

この手順では、前のステップで作成したキャンペーンを使用してレコメンデーションを取得します。

推奨事項を取得するには

1. [Test campaign result (キャンペーン結果のテスト)] の [ユーザー ID] で、レーティングの値 (例: **83**) を指定します。他のすべてのフィールドは変更しないでください。
2. [レコメンデーションの取得] を選択します。[Recommendations] (レコメンデーション) のパネルには、推奨アイテムのアイテム ID とスコアが一覧表示されます。

画面の表示は次のようになります。



Item ID	Score	Recommendation reason
318	0.0032059	-
589	0.0027560	-
593	0.0023468	-
356	0.0022429	-
527	0.0022052	-
296	0.0021353	-
457	0.0020748	-
50	0.0020146	-
150	0.0019477	-
1	0.0019159	-
780	0.0019149	-
480	0.0018700	-
592	0.0018420	-
1221	0.0018401	-
110	0.0017960	-
590	0.0017893	-
1198	0.0016929	-
47	0.0016593	-
32	0.0016585	-
364	0.0015704	-
500	0.0015517	-
7361	0.0015233	-

開始方法 (AWS CLI)

この演習では、 AWS Command Line Interface （ AWS CLI) を使用して Amazon Personalize を試します。映画のレコメンデーションを特定のユーザー ID に返すキャンペーンを作成します。

この演習を開始する前に、以下を実行します。

- 開始方法「開始方法の前提条件」を見る
- 「」で指定されているように AWS CLI、 をセットアップしますのセットアップ AWS CLI。

開始方法の演習を完了したら、不要な料金が発生しないように、作成したリソースを、「リソースのクリーンアップ」の手順に従って削除します。

> **Note**
>
> この演習の AWS CLI コマンドは Linux でテスト済みです。Windows での AWS CLI コマンドの使用については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interfaceのパラメータ値の指定」を参照してください。

ステップ 1: トレーニングデータをインポートする

次の手順に従って、データセットグループを作成し、このグループにデータセットを追加します。次に、映画のレーティングデータを使用してデータセットを事前設定します。

1. 次のコマンドを実行してデータセットグループを作成します。データセットグループを暗号化するには、AWS Key Management Service キー ARN と、そのキーへの許可を持つ IAM ロールの ARN を入力パラメータとして渡します。API の詳細については、「CreateDatasetGroup」を参照してください。

   ```
   aws personalize create-dataset-group --name MovieRatingDatasetGroup --kms-key-
   arn arn:aws:kms:us-west-2:01234567890:key/1682a1e7-a94d-4d92-bbdf-837d3b62315e --
   role-arn arn:aws:iam::01234567890:KMS-key-access
   ```

   データセットグループの ARN が表示されます。次に例を示します。

   ```
   {
     "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/
   MovieRatingDatasetGroup"
   ```

```
}
```

作成したデータセットグループを表示するには、返されたデータセットグループの ARN を指定して、describe-dataset-group コマンドを使用します。

```
aws personalize describe-dataset-group \
--dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieRatingDatasetGroup
```

データセットグループとそのプロパティが表示されます。次に例を示します。

```
{
    "datasetGroup": {
        "name": "MovieRatingDatasetGroup",
        "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieRatingDatasetGroup",
        "status": "ACTIVE",
        "creationDateTime": 1542392161.262,
        "lastUpdatedDateTime": 1542396513.377
    }
}
```

> **Note**
>
> データセットグループ内にデータセットを作成するには、データセットグループの status が ACTIVE になるまで待ちます。通常、このオペレーションは高速に処理されます。

データセットグループの ARN が不明な場合は、list-dataset-groups コマンドを使用して、作成したすべてのデータセットグループとその ARN を表示します。

```
aws personalize list-dataset-groups
```

> **Note**
>
> `describe-object` および `list-objects` コマンドは、ほとんどの Amazon Personalize オブジェクトで使用できます。これらのコマンドは、この演習で以後取り扱いませんが、いつでも使用可能です。

2. 次のコードを `MovieRatingSchema.json` という名前のファイルに保存して、スキーマファイルを JSON 形式で作成します。スキーマは、以前に `ratings.csv` に追加したヘッダーと一致します。スキーマ名である `Interactions` は、Amazon Personalize で認識されるデータセットの種類の 1 つと一致します。詳細については、「スキーマ」を参照してください。

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
}
```

3. 次のコマンドを実行してスキーマを作成します。前のステップで保存したファイルを指定します。次の例は、現在のフォルダに属するファイルを示しています。API の詳細については、「CreateSchema」を参照してください。

```
aws personalize create-schema \
  --name MovieRatingSchema \
  --schema file://MovieRatingSchema.json
```

スキーマの Amazon リソースネーム (ARN) が表示されます。次に例を示します。

```
{
  "schemaArn": "arn:aws:personalize:us-west-2:acct-id:schema/MovieRatingSchema"
}
```

4. 次のコマンドを実行して空のデータセットを作成します。前のステップで返されたデータセットグループの ARN とスキーマの ARN を指定します。dataset-type は、前のステップのスキーマ name と一致する必要があります。API の詳細については、「CreateDataset」を参照してください。

```
aws personalize create-dataset \
  --name MovieRatingDataset \
  --dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieRatingDatasetGroup \
  --dataset-type Interactions \
  --schema-arn arn:aws:personalize:us-west-2:acct-id:schema/MovieRatingSchema
```

データセットの ARN が表示されます。次に例を示します。

```
{
  "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/
MovieRatingDatasetGroup/INTERACTIONS"
}
```

5. トレーニングデータをデータセットに追加します。

   a. 次のコマンドを実行してデータセットのインポートジョブを作成します。前のステップで返されたデータセットの ARN と Amazon S3 バケットの名前を指定します。で作成した AWS Identity and Access Management (IAM) ロール ARN を指定しますAmazon Personalize 向けの IAM ロールの作成。API の詳細については、「CreateDatasetImportJob」を参照してください。

```
aws personalize create-dataset-import-job \
  --job-name MovieRatingImportJob \
  --dataset-arn arn:aws:personalize:us-west-2:acct-id:dataset/
MovieRatingDatasetGroup/INTERACTIONS \
  --data-source dataLocation=s3://bucketname/ratings.csv \
  --role-arn roleArn
```

データセットのインポートジョブの ARN が表示されます。次に例を示します。

```
{
  "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-
job/MovieRatingImportJob"
}
```

b. describe-dataset-import-job コマンドを使用してステータスを確認します。前のステップで返されたデータセットのインポートジョブの ARN を指定します。API の詳細については、「DescribeDatasetImportJob」を参照してください。

```
aws personalize describe-dataset-import-job \
  --dataset-import-job-arn arn:aws:personalize:us-west-2:acct-id:dataset-
import-job/MovieRatingImportJob
```

データセットのインポートジョブのプロパティとそのステータスが表示されます。最初、status は CREATE PENDING と表示されます。次に例を示します。

```
{
  "datasetImportJob": {
      "jobName": "MovieRatingImportJob",
      "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-
import-job/MovieRatingImportJob",
      "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/
MovieRatingDatasetGroup/INTERACTIONS",
      "dataSource": {
          "dataLocation": "s3://<bucketname>/ratings.csv"
      },
      "roleArn": "role-arn",
      "status": "CREATE PENDING",
      "creationDateTime": 1542392161.837,
      "lastUpdatedDateTime": 1542393013.377
  }
}
```

ステータスが ACTIVE と表示されると、データセットのインポートが完了します。これで、指定したデータセットを使用してモデルをトレーニングする準備が整いました。

> **Note**
>
> インポートには時間がかかります。データセットのインポートが完了するまで待ってから、データセットを使用してモデルのトレーニングを開始します。

ステップ 2: ソリューションを作成する (モデルをトレーニングする)

モデルをトレーニングするには、 CreateSolutionオペレーションを使用してモデルをトレーニングするための設定を作成し、自動トレーニングをオンのままにします。このソリューションは、最初のソリューションのトレーニングを 1 時間以内に自動的に開始します。

レシピとトレーニングデータを使用してモデルをトレーニングします。Amazon Personalize は、事前定義された一連のレシピを提供します。詳細については、「レシピの選択」を参照してください。この演習では、User-Personalization-v2 レシピを使用します。

1. モデルトレーニングの設定を作成するには、次のコマンドを実行します。このコマンドは、自動トレーニングを使用するソリューションを作成します。7 日ごとに新しいソリューションバージョンが自動的に作成されます (デフォルト) 。

```
aws personalize create-solution \
  --name MovieSolution \
  --dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieRatingDatasetGroup \
  --recipe-arn arn:aws:personalize:::recipe/aws-user-personalization-v2 \
  --perform-auto-training \
  --solution-config "{\"autoTrainingConfig\": {\"schedulingExpression\": \"rate(7
 days)\"}}"
```

   ソリューションの ARN が表示されます。次に例を示します。

```
{
  "solutionArn": "arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution"
}
```

2. `describe-solution` コマンドを使用して [create] (作成) のステータスを確認します。前のステップで返されたソリューションの ARN を指定します。API の詳細については、「DescribeSolution」を参照してください。

```
aws personalize describe-solution \
  --solution-arn arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution
```

ソリューションのプロパティと作成の status が表示されます。例:

```
{
  "solution": {
      "name": "MovieSolution",
      "solutionArn": "arn:aws:personalize:us-west-2:acct-id:solution/
MovieSolution",
      "performHPO": false,
      "performAutoML": false,
      "recipeArn": "arn:aws:personalize:::recipe/aws-user-personalization-v2",
      "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieRatingDatasetGroup",
      "solutionConfig": {
            "algorithmHyperParameters": {
                "apply_recency_bias": "true"
            },
            "featureTransformationParameters": {},
            "autoTrainingConfig": {
                "schedulingExpression": "rate(7 days)"
            }
       },
      "status": "ACTIVE",
      "creationDateTime": "2021-05-12T16:27:59.819000-07:00",
      "lastUpdatedDateTime": "2021-05-12T16:27:59.819000-07:00"
  }
}
```

3. 自動トレーニングでは、ソリューションバージョントレーニングは、ソリューションが ACTIVE になった後 1 時間以内に開始されます。トレーニングの開始後、次の Versions コマンドを使用して、ソリューションバージョンの Amazon リソースネーム (ARN) ListSolutionを取得できます。

```
aws personalize list-solution-versions --solution-arn arn:aws:personalize:us-
west-2:acct-id:solution/MovieSolution
```

4. describe-solution-version コマンドを使用して、ソリューションバージョンの [training] (トレーニング) ステータスを確認します。前のステップで返ったソリューションバージョンの

ARN を指定します。API の詳細については、「DescribeSolutionVersion」を参照してください。

```
aws personalize describe-solution-version \
  --solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/
MovieSolution/version-id
```

ソリューションバージョンのプロパティとトレーニングの status が表示されます。最初、ステータスには CREATE PENDING と表示されます。次に例を示します。

```
{
  "solutionVersion": {
      "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/
MovieSolution/<version-id>",
      ...,
      "status": "CREATE PENDING"
  }
}
```

5. ソリューションバージョン status が ACTIVE になったら、トレーニングは完了です。

これで、トレーニングメトリクスを確認し、ソリューションバージョンを使用してキャンペーンを作成できます。

> **Note**
>
> トレーニングには時間がかかります。トレーニングが完了する (ソリューションバージョンのトレーニングのステータスが ACTIVE と表示される) まで待ってから、このバージョンのソリューションをキャンペーンで使用します。

6. ソリューションバージョンのパフォーマンスを検証するには、そのメトリクスを確認できます。次のコマンドを実行して、ソリューションバージョンのメトリクスを取得します。以前に返ったソリューションバージョンの ARN を指定します。API の詳細については、「GetSolutionMetrics」を参照してください。

```
aws personalize get-solution-metrics \
  --solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/
MovieSolution/version-id
```

レスポンスの例を次に示します。

```
{
  "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/www-
solution/<version-id>",
  "metrics": {
        "coverage": 0.0485,
        "mean_reciprocal_rank_at_25": 0.0381,
        "normalized_discounted_cumulative_gain_at_10": 0.0363,
        "normalized_discounted_cumulative_gain_at_25": 0.0984,
        "normalized_discounted_cumulative_gain_at_5": 0.0175,
        "precision_at_10": 0.0107,
        "precision_at_25": 0.0207,
        "precision_at_5": 0.0107
    }
}
```

ステップ 3: キャンペーンを作成する (ソリューションをデプロイする)

レコメンデーションを取得するには、事前にソリューションバージョンをデプロイする必要があります。ソリューションのデプロイは、キャンペーンの作成とも呼ばれます。キャンペーンを作成すると、クライアントアプリケーションは GetRecommendations API を使用してレコメンデーションを取得できます。

1. 次のコマンドを実行してキャンペーンを作成します。前のステップで返ったソリューションバージョンの ARN を指定します。API の詳細については、「CreateCampaign」を参照してください。

   ```
   aws personalize create-campaign \
     --name MovieRecommendationCampaign \
     --solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/
   MovieSolution/version-id \
     --min-provisioned-tps 1
   ```

   レスポンスの例を次に示します。

   ```
   {
     "campaignArn": "arn:aws:personalize:us-west-2:acct-id:campaign/
   MovieRecommendationCampaign"
   }
   ```

2. 次のコマンドを実行してデプロイのステータスを確認します。前のステップで返されたキャンペーンの ARN を指定します。API の詳細については、「DescribeCampaign」を参照してください。

```
aws personalize describe-campaign \
  --campaign-arn arn:aws:personalize:us-west-2:acct-id:campaign/
MovieRecommendationCampaign
```

レスポンスの例を次に示します。

```
{
  "campaign": {
      "name": "MovieRecommendationCampaign",
      "campaignArn": "arn:aws:personalize:us-west-2:acct-id:campaign/
MovieRecommendationCampaign",
      "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/
MovieSolution/<version-id>",
      "minProvisionedTPS": "1",
      "creationDateTime": 1543864775.923,
      "lastUpdatedDateTime": 1543864791.923,
      "status": "CREATE IN_PROGRESS"
  }
}
```

> **Note**
>
> キャンペーンからレコメンデーションを取得するには、status が ACTIVE と表示されるまで待ちます。

ステップ 4: レコメンデーションを取得する

get-recommendations コマンドを実行してレコメンデーションを取得します。前のステップで返されたキャンペーンの ARN を指定します。リクエストでは、映画のレーティングデータセットのユーザー ID を指定します。API の詳細については、「GetRecommendations」を参照してください。

> **Note**
>
> すべてのレシピで GetRecommendations API がサポートされているわけではありません。詳細については、「レシピの選択」を参照してください。
> このステップで呼び出す AWS CLI コマンド はpersonalize-runtime、前のステップとは異なります。

```
aws personalize-runtime get-recommendations \
  --campaign-arn arn:aws:personalize:us-west-2:acct-id:campaign/
MovieRecommendationCampaign \
  --user-id 123
```

キャンペーンは、ユーザーが好むと思われるアイテムのレコメンデーション (映画 ID) のリストをレスポンスで返します。リストは、ユーザーに関連する度合いの高い順 (降順) にソートされます。

```
{
  "itemList": [
      {
          "itemId": "14"
      },
      {
          "itemId": "15"
      },
      {
          "itemId": "275"
      },
      {
          "itemId": "283"
      },
      {
          "itemId": "273"
      },
      ...
  ]
}
```

開始方法 (SDK for Python (Boto3))

このチュートリアルでは、SDK for Python (Boto3) を使用して Amazon Personalize のワークフローを最初から最後まで完了する方法を示します。

不要な料金が発生しないように、開始方法の演習が終了したら、このチュートリアルで作成したリソースを削除してください。詳細については、「リソースのクリーンアップ」を参照してください。

トピック

- 前提条件
- チュートリアル
- Jupyter (iPython) ノートブックでの Amazon Personalize API の開始方法

前提条件

このガイドで Python の例を使用するための前提条件のステップを以下に示します。

- 開始方法の前提条件 を完了して必要な権限を設定し、トレーニングデータを作成します。独自のソースデータを使用する場合は、前提条件に示しているようにそのデータがフォーマットされていることを確認します。
- 「」で指定されているように AWS SDK for Python (Boto3) 環境を設定しますAWS SDKsのセットアップ。

チュートリアル

次のステップでは、環境を確認し、Amazon Personalize 用の SDK for Python (Boto3) クライアントを作成します。次に、データをインポートし、キャンペーンを含むソリューションバージョンを作成してデプロイし、レコメンデーションを取得します。

ステップ 1: Python 環境を確認し、boto3 クライアントを作成する

前提条件のステップを完了したら、以下の Python の例を実行して、環境が適切に設定されていることを確認します。このコードでは、このチュートリアルで使用する Amazon Personalize boto3 クライアントも作成されます。環境が正しく設定されている場合、使用可能なレシピのリストが表示され、このガイドの他の例を実行できます。

```
import boto3
```

```
personalizeRt = boto3.client('personalize-runtime')
personalize = boto3.client('personalize')

response = personalize.list_recipes()

for recipe in response['recipes']:
    print (recipe)
```

ステップ 2: データをインポートする

Amazon Personalize boto3 クライアントを作成して環境を検証したら、「開始方法の前提条件」の完了時に作成した履歴データをインポートします。過去のデータを Amazon Personalize にインポートするには、次の操作を実行します。

1. 以下のコードを使用して、Amazon Personalize でスキーマを作成します。`getting-started-schema` を、スキーマの名前に置き換えます。

```
import json
schema = {
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
}

create_interactions_schema_response = personalize.create_schema(
    name='getting-started-schema',
    schema=json.dumps(schema)
)
```

```
interactions_schema_arn = create_interactions_schema_response['schemaArn']
print(json.dumps(create_interactions_schema_response, indent=2))
```

2. 次のコードを使用してデータセットグループを作成します。dataset group name をデータセットグループの名前に置き換えます。

```
response = personalize.create_dataset_group(name = 'dataset group name')
dataset_group_arn = response['datasetGroupArn']

description = personalize.describe_dataset_group(datasetGroupArn = dataset_group_arn)
['datasetGroup']

print('Name: ' + description['name'])
print('ARN: ' + description['datasetGroupArn'])
print('Status: ' + description['status'])
```

3. 次のコードを使用して、新しいデータセットグループにアイテムインタラクションデータセットを作成します。データセットに名前を付け、前のステップの schema_arn および dataset_group_arn を指定します。

```
response = personalize.create_dataset(
    name = 'datase_name',
    schemaArn = 'schema_arn',
    datasetGroupArn = 'dataset_group_arn',
    datasetType = 'Interactions'
)

dataset_arn = response['datasetArn']
```

4. 以下のコードとともにデータセットのインポートジョブを使用してデータをインポートします。このコードでは describe_dataset_import_job メソッドを使用してジョブのステータスを追跡します。

   ジョブの名前、前のステップの dataset_arn、トレーニングデータおよび IAM サービスロール ARN を格納した Amazon S3 バケットパス (s3://*bucket name*/*folder name*/ratings.csv) をパラメータに渡します。このロールは 開始方法の前提条件 の一部として作成しました。Amazon Personalize には、バケットにアクセスするための許可が必要です。Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与 を参照してください。

```
import time
```

```
response = personalize.create_dataset_import_job(
    jobName = 'JobName',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn',
    importMode = 'FULL'
)

dataset_interactions_import_job_arn = response['datasetImportJobArn']

description = personalize.describe_dataset_import_job(
    datasetImportJobArn = dataset_interactions_import_job_arn)['datasetImportJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['datasetImportJobArn'])
print('Status: ' + description['status'])

max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:
    describe_dataset_import_job_response = personalize.describe_dataset_import_job(
        datasetImportJobArn = dataset_interactions_import_job_arn
    )
    status = describe_dataset_import_job_response["datasetImportJob"]['status']
    print("Interactions DatasetImportJob: {}".format(status))

    if status == "ACTIVE" or status == "CREATE FAILED":
        break

    time.sleep(60)
```

ステップ 3: ソリューションを作成する

データをインポートしたら、次のようにソリューションとソリューションバージョンを作成します。ソリューションにはモデルをトレーニングするための設定が含まれており、ソリューションバージョンはトレーニング済みモデルです。

1. 次のコードを使用して新しいソリューションを作成します。パラメータとして、前の `dataset_group_arn` からの 、ソリューションの名前、User-Personalization-v2 レシピの ARN () を渡しますarn:aws:personalize:::recipe/aws-user-personalization-v2。新しいソリューションの ARN は、後で使用できるように保存します。

```
create_solution_response = personalize.create_solution(
  name='solution name',
  recipeArn= 'arn:aws:personalize:::recipe/aws-user-personalization-v2',
  datasetGroupArn = 'dataset group arn'
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

2. 次のコードを使用してソリューションバージョンを作成します。前のステップからの solution_arn をパラメータとして渡します。次のコードは、ソリューションバージョンを作成します。トレーニング中、コードは DescribeSolutionVersion 操作を使用してソリューションバージョンのステータスを取得します。トレーニングが完了すると、メソッドは新しいソリューションバージョンの ARN を返します。後で使用するために、これを保存します。

```
import time
import json

create_solution_version_response = personalize.create_solution_version(
    solutionArn = solution_arn
)

solution_version_arn = create_solution_version_response['solutionVersionArn']
print(json.dumps(create_solution_version_response, indent=2))

max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:
    describe_solution_version_response = personalize.describe_solution_version(
        solutionVersionArn = solution_version_arn
    )
    status = describe_solution_version_response["solutionVersion"]["status"]
    print("SolutionVersion: {}".format(status))

    if status == "ACTIVE" or status == "CREATE FAILED":
        break

    time.sleep(60)
```

ステップ 4: キャンペーンの作成

ソリューションバージョンをトレーニングして評価した後、Amazon Personalize キャンペーンでそれをデプロイします。次のコードを使用して、ソリューションバージョンをデプロイするキャンペーンを作成します。パラメータとして、solution_version_arn、およびキャンペーンの名前を渡します。このメソッドは、新しいスキーマの Amazon リソースネーム (ARN) を返します。後で使用するために、これを保存します。

```
response = personalize.create_campaign(
    name = 'campaign name',
    solutionVersionArn = 'solution version arn'
)

arn = response['campaignArn']

description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
print('Status: ' + description['status'])
```

ステップ 5: レコメンデーションを取得する

キャンペーンを作成したら、そのキャンペーンを使用して推奨事項を取得できます。次のコードは、キャンペーンからレコメンデーションを取得し、各レコメンデーションアイテムの ID をプリントアウトする方法を示しています。前のステップで作成したキャンペーンの ARN を渡します。ユーザー ID には、123 などのトレーニングデータからユーザーの ID を渡します。

```
response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = '123',
    numResults = 10
)

print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
```

Jupyter (iPython) ノートブックでの Amazon Personalize API の開始方法

Jupyter ノートブックを使用して Amazon Personalize の使用を開始するには、Amazon Personalize サンプルリポジトリの getting_started フォルダにある一連のノートブックのクローンを作成するか

ダウンロードします。ノートブックでは、トレーニングデータのインポート、ソリューションの作成、キャンペーンの作成、および Amazon Personalize を使用したレコメンデーションの取得について説明します。

> **Note**
>
> ノートブックを使用する前に、README.md の手順に従って環境を構築してください。

開始方法 (SDK for Java 2.x)

このチュートリアルでは、 AWS SDK for Java 2.xを使用して Amazon Personalize のワークフローを最初から最後まで完了する方法を示します。

不要な料金が発生しないように、開始方法の演習が終了したら、「リソースのクリーンアップ」の手順に従って、チュートリアルで作成したリソースを削除してください。

その他の例については、「Amazon Personalize のプロジェクトを完了する」を参照してください。

トピック

- 前提条件
- Amazon Personalize のプロジェクトを完了する

前提条件

このチュートリアルを完了するための前提条件となる手順は次のとおりです。

- 開始方法の前提条件 を完了して、必要な権限を設定し、トレーニングデータを作成します。開始方法 (コンソール) または 開始方法 (AWS CLI) の演習で使用したのと同じソースデータを使用できます。独自のソースデータを使用する場合は、前提条件のようにそのデータがフォーマットされていることを確認します。
- 「 AWS SDK for Java 2.x デベロッパーガイド」の「 のセットアップ」の手順で指定されているように、SDK for Java AWS SDK for Java 2.x2.x 環境と AWS 認証情報を設定します。

チュートリアル

次のステップでは、Amazon Personalize パッケージを使用するようにプロジェクトを設定し、Amazon Personalize の SDK for Java 2.x クライアントを作成します。次に、データをインポー

トし、キャンペーンを含むソリューションバージョンを作成およびデプロイして、レコメンデーションを取得します。

ステップ 1: Amazon Personalize のパッケージを使用するようにプロジェクトを設定する

前提条件を完了したら、Amazon Personalize の依存関係を pom.xml ファイルに追加し、Amazon Personalize のパッケージをインポートします。

1. 次の依存関係を pom.xml ファイルに追加します。最新のバージョン番号は、サンプルコードとは異なる場合があります。

```
<dependency>
 <groupId>software.amazon.awssdk</groupId>
 <artifactId>personalize</artifactId>
 <version>2.16.83</version>
</dependency>
<dependency>
 <groupId>software.amazon.awssdk</groupId>
 <artifactId>personalizeruntime</artifactId>
 <version>2.16.83</version>
</dependency>
<dependency>
 <groupId>software.amazon.awssdk</groupId>
 <artifactId>personalizeevents</artifactId>
 <version>2.16.83</version>
</dependency>
```

2. プロジェクトに次のインポートステートメントを追加します。

```
// import client packages
import software.amazon.awssdk.services.personalize.PersonalizeClient;
import software.amazon.awssdk.services.personalizeruntime.PersonalizeRuntimeClient;
// Amazon Personalize exception package
import software.amazon.awssdk.services.personalize.model.PersonalizeException;
// schema packages
import software.amazon.awssdk.services.personalize.model.CreateSchemaRequest;
// dataset group packages
import software.amazon.awssdk.services.personalize.model.CreateDatasetGroupRequest;
import software.amazon.awssdk.services.personalize.model.DescribeDatasetGroupRequest;
// dataset packages
import software.amazon.awssdk.services.personalize.model.CreateDatasetRequest;
// dataset import job packages
```

```
import
 software.amazon.awssdk.services.personalize.model.CreateDatasetImportJobRequest;
import software.amazon.awssdk.services.personalize.model.DataSource;
import software.amazon.awssdk.services.personalize.model.DatasetImportJob;
import
 software.amazon.awssdk.services.personalize.model.DescribeDatasetImportJobRequest;
// solution packages
import software.amazon.awssdk.services.personalize.model.CreateSolutionRequest;
import software.amazon.awssdk.services.personalize.model.CreateSolutionResponse;
// solution version packages
import software.amazon.awssdk.services.personalize.model.DescribeSolutionRequest;
import
 software.amazon.awssdk.services.personalize.model.CreateSolutionVersionRequest;
import
 software.amazon.awssdk.services.personalize.model.CreateSolutionVersionResponse;
import
 software.amazon.awssdk.services.personalize.model.DescribeSolutionVersionRequest;
// campaign packages
import software.amazon.awssdk.services.personalize.model.CreateCampaignRequest;
import software.amazon.awssdk.services.personalize.model.CreateCampaignResponse;
// get recommendations packages
import
 software.amazon.awssdk.services.personalizeruntime.model.GetRecommendationsRequest;
import
 software.amazon.awssdk.services.personalizeruntime.model.GetRecommendationsResponse;
import software.amazon.awssdk.services.personalizeruntime.model.PredictedItem;
// Java time utility package
import java.time.Instant;
```

ステップ 2: Amazon Personalize のクライアントを作成する

Amazon Personalize の依存関係を pom.xml ファイルに追加し、必要なパッケージをインポートして、次の Amazon Personalize のクライアントをインスタンス化します。

```
PersonalizeClient personalizeClient = PersonalizeClient.builder()
  .region(region)
  .build();

PersonalizeRuntimeClient personalizeRuntimeClient = PersonalizeRuntimeClient.builder()
  .region(region)
  .build();
```

ステップ 3: データをインポートする

Amazon Personalize のクライアントを初期化したら、開始方法の前提条件 の完了時に作成した履歴データをインポートします。過去のデータを Amazon Personalize にインポートするには、次の操作を実行します。

1. 次の Avro スキーマを JSON ファイルとして作業ディレクトリに保存します。このスキーマは、開始方法の前提条件 を完了したときに作成した CSV ファイルの列と一致します。

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
}
```

2. 次の `createSchema` メソッドを使用して、Amazon Personalize でスキーマを作成します。パラメータとして、Amazon Personalize のサービスクライアント、スキーマの名前、および前のステップで作成したスキーマ JSON ファイルのファイルパスを渡します。このメソッドは、新しいスキーマの Amazon リソースネーム (ARN) を返します。後で使用するために、これを保存します。

```
    public static String createSchema(PersonalizeClient personalizeClient, String
 schemaName, String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
```

```
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .schema(schema)
                    .build();

            String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

3. データセットグループを作成します。次の createDatasetGroup メソッドを使用して、データセットグループを作成します。パラメータとして、Amazon Personalize のサービスクライアントとデータセットグループの名前を渡します。このメソッドは、新しいデータセットグループの ARN を返します。後で使用するために、これを保存します。

```
    public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .build();
            return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
```

```
    }
```

4. アイテムインタラクションデータセットを作成します。次の createDataset メソッドを使用して、アイテムインタラクションデータセットを作成します。パラメータとして、Amazon Personalize のサービスクライアント、データセットの名前、スキーマの ARN、データセットグループの ARN、およびデータセットタイプの Interactions を渡します。このメソッドは、新しいデータセットの ARN を返します。後で使用するために、これを保存します。

```
    public static String createDataset(PersonalizeClient personalizeClient,
            String datasetName,
            String datasetGroupArn,
            String datasetType,
            String schemaArn) {
        try {
            CreateDatasetRequest request = CreateDatasetRequest.builder()
                    .name(datasetName)
                    .datasetGroupArn(datasetGroupArn)
                    .datasetType(datasetType)
                    .schemaArn(schemaArn)
                    .build();

            String datasetArn = personalizeClient.createDataset(request)
                    .datasetArn();
            System.out.println("Dataset " + datasetName + " created.");
            return datasetArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

5. データセットのインポートジョブを使用してデータをインポートします。次の createPersonalizeDatasetImportJob メソッドを使用して、データセットのインポートジョブを作成します。

   パラメータとして、Amazon Personalize サービスクライアント、ジョブの名前、アイテムインタラクションデータセットの ARN、トレーニングデータを保存した Amazon S3 バケットパス (s3://*bucket name*/*folder name*/ratings.csv) 、サービスロールの ARN (の一部としてこのロールを作成) を渡します開始方法の前提条件。このメソッドは、データセットのインポートジョブの ARN を返します。オプションで、後で使用するために保存します。

```
   public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
           String jobName,
           String datasetArn,
           String s3BucketPath,
           String roleArn) {

       long waitInMilliseconds = 60 * 1000;
       String status;
       String datasetImportJobArn;

       try {
           DataSource importDataSource = DataSource.builder()
                   .dataLocation(s3BucketPath)
                   .build();

           CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                   .datasetArn(datasetArn)
                   .dataSource(importDataSource)
                   .jobName(jobName)
                   .roleArn(roleArn)
                   .build();

           datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                   .datasetImportJobArn();
           DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                   .datasetImportJobArn(datasetImportJobArn)
                   .build();

           long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

           while (Instant.now().getEpochSecond() < maxTime) {

               DatasetImportJob datasetImportJob = personalizeClient
                       .describeDatasetImportJob(describeDatasetImportJobRequest)
                       .datasetImportJob();

               status = datasetImportJob.status();
               System.out.println("Dataset import job status: " + status);
```

```
                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

ステップ 4: ソリューションを作成する

データをインポートしたら、次のようにソリューションとソリューションバージョンを作成します。ソリューションにはモデルをトレーニングするための設定が含まれており、ソリューションバージョンはトレーニング済みモデルです。

1. 次の createPersonalizeSolution メソッドで新しいソリューションを作成します。パラメータとして、Amazon Personalize サービスクライアント、データセットグループ Amazon リソースネーム (ARN)、ソリューションの名前、User-Personalization-v2 レシピの ARN () を渡しますarn:aws:personalize:::recipe/aws-user-personalization-v2。このメソッドは、新しいソリューションの ARN を返します。後で使用するために、これを保存します。

```
    public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
            String datasetGroupArn,
            String solutionName,
            String recipeArn) {

        try {
            CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
                    .name(solutionName)
                    .datasetGroupArn(datasetGroupArn)
                    .recipeArn(recipeArn)
                    .build();
```

```
            CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
            return solutionResponse.solutionArn();

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

2. 次の createPersonalizeSolutionVersion メソッドでソリューションバージョンを作成します。前のステップのソリューションの ARN をパラメータとして渡します。次のコードを実行すると、最初にソリューションの準備ができているかどうかを確認してから、ソリューションバージョンを作成します。トレーニング中、コードは DescribeSolutionVersion 操作を使用してソリューションバージョンのステータスを取得します。トレーニングが完了すると、メソッドは新しいソリューションバージョンの ARN を返します。後で使用するために、これを保存します。

```
    public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String solutionStatus = "";
        String solutionVersionStatus = "";
        String solutionVersionArn = "";

        try {
            DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            // Wait until solution is active.
            while (Instant.now().getEpochSecond() < maxTime) {

                solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
                System.out.println("Solution status: " + solutionStatus);
```

```
                if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }

            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                        .solutionArn(solutionArn)
                        .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                        .createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " + solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                        .solutionVersionArn(solutionVersionArn)
                        .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                            .solutionVersion().status();
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
```

```
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return solutionVersionArn;
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
  }
```

詳細については、「ソリューションとソリューションバージョンの作成」を参照してください。ソリューションバージョンを作成したら、先に進む前にそのパフォーマンスを評価できます。詳細については、「メトリクスを使用してソリューションバージョンを評価する」を参照してください。

ステップ 5: キャンペーンを作成する

ソリューションバージョンをトレーニングして評価した後、Amazon Personalize キャンペーンでそれをデプロイします。次の `createPersonalCampaign` メソッドを使用して、ソリューションバージョンをデプロイします。パラメータとして、Amazon Personalize のサービスクライアント、前のステップで作成したソリューションバージョンの Amazon リソースネーム (ARN)、およびキャンペーンの名前を渡します。このメソッドは、新しいキャンペーンの ARN を返します。後で使用するために、これを保存します。

```
public static String createPersonalCompaign(PersonalizeClient personalizeClient, String
 solutionVersionArn, String name) {

    try {
        CreateCampaignRequest createCampaignRequest = CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
 personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is "+campaignResponse.campaignArn());
        return campaignResponse.campaignArn();
```

```
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Amazon Personalize のキャンペーンの詳細については、「キャンペーンの作成」を参照してください。

ステップ 6: レコメンデーションを取得する

キャンペーンを作成したら、そのキャンペーンを使用してレコメンデーションを取得します。次の getRecs メソッドを使用して、ユーザー向けのレコメンデーションを取得します。パラメータとして、Amazon Personalize のランタイムクライアント、前のステップで作成したキャンペーンの Amazon リソースネーム (ARN)、およびインポートした履歴データからのユーザー ID (例: 123) を渡します。このメソッドは、推奨アイテムのリストを画面に出力します。

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();
           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }

       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
    }
```

Amazon Personalize のプロジェクトを完了する

SDK for Java 2.x を使用して Amazon Personalize ワークフローを完了する方法を示す all-in-one プロジェクトについては、「」の「Amazon-Personalize-Java-App」を参照してください GitHub。このプロジェクトには、異なるレシピで複数のソリューションバージョンをトレーニングし、PutEvents オペレーションでイベントを記録することが含まれます。

その他の例については、 AWS SDK サンプルリポジトリの Personalize フォルダにある コードを参照してください。

リソースのクリーンアップ

不要な料金が発生しないようにするには、使用開始実習の終了後に作成したリソースを削除します。Amazon Personalize コンソールでリソースを削除するには、リソースの詳細ページで削除を選択します。Amazon Personalize APIs、 SDKsまたは AWS Command Line Interface () で `Delete` APIs を使用しますAWS CLI。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs「」を参照してくださいデータセットを削除してすべてのデータを削除する。これらのステップのパターンは、他の Amazon Personalize リソースに適用できます。

一部のリソースは、他のリソースよりも先に削除する必要があります。以下のセクションでは、 リソースを削除するためのガイドラインと、ドメインベースのリソースとカスタムリソースを削除する順序を示します。

トピック

- リソースをクリーンアップするためのガイドライン
- ドメインベースのリソースのクリーンアップ
- カスタムリソースのクリーンアップ

リソースをクリーンアップするためのガイドライン

以下は、ドメインベースのリソースとカスタムリソースの両方に適用されます。

- ステータスが CREATE PENDING または IN PROGRESS のリソースを削除することはできません。リソースのステータスは ACTIVE または CREATE FAILED である必要がありま

す。DescribeCampaign API オペレーションなどの Describe APIsを使用してステータスを確認します。

- Amazon S3 にアップロードしたトレーニングデータを削除するには、S3 バケットからオブジェクトを削除する方法ratings.csv」を参照してください。
- データセットを削除する前に、データセットのすべてのデータセットインポートジョブが完了していることを確認してください。
- データセットのインポートジョブは、完了後は課金されず、削除することもできません。
- Amazon Personalize コンソールでスキーマを削除することはできません。また、スキーマの保存に対して課金されることはありません。スキーマを削除するには、 DeleteSchema API オペレーションを使用します。

ドメインベースのリソースのクリーンアップ

ドメインデータセットグループを作成した場合は、次の順序でリソースを削除します。

1. レコメンダー – APIsオペレーションを使用します。 DeleteRecommenderコンソールでレコメンダーを削除するには、レコメンダーページでレコメンダーを選択し、右上の削除を選択します。
2. データセット – APIsオペレーションを使用します。 DeleteDatasetコンソールでデータセットを削除するには、データセットページでデータセットを選択して詳細ページを表示します。次に、右上の「削除」を選択します。

   > **Note**
   >
   > データセットを削除するには、関連付けられたデータセットのインポートジョブのステータスが CREATE PENDING または IN PROGRESS になることはできません。また、関連するレコメンダーのステータスが CREATE PENDING または IN PROGRESS になることはできません。

3. データセットグループ – APIsオペレーションを使用します。 DeleteDatasetGroupコンソールでデータセットグループを削除するには、データセットグループページでデータセットグループを選択し、右上の「削除」を選択します。

カスタムリソースのクリーンアップ

カスタムデータセットグループを作成した場合は、このチュートリアルで作成したカスタムリソースを次の順序で削除します。

1. キャンペーン – APIsオペレーションを使用します。 DeleteCampaignコンソールでキャンペーンを削除するには、キャンペーンページでキャンペーンを選択し、右上の「削除」を選択します。
2. 解決策 — APIsオペレーションを使用します。 DeleteSolutionコンソールでソリューションを削除するには、ソリューションページでソリューションを選択して詳細ページを表示します。次に、右上の「削除」を選択します。

   > **Note**
   >
   > ソリューションを削除すると、関連するすべてのソリューションバージョンが削除されます。どのソリューションバージョンも、CREATE PENDING または IN PROGRESS のステータスにすることはできません。

3. データセット – APIsオペレーションを使用します。 DeleteDatasetコンソールでデータセットを削除するには、データセットページでデータセットを選択して詳細ページを表示します。次に、右上の「削除」を選択します。

   > **Note**
   >
   > データセットを削除するには、関連付けられた のステータスDatasetImportJobが CREATE PENDING または IN PROGRESS になることはできません。またSolutionVersion、関連付けられた は、CREATE PENDING または IN PROGRESS のステータスを持つことはできません。

4. データセットグループ – APIsオペレーションを使用します。 DeleteDatasetGroupコンソールでデータセットグループを削除するには、データセットグループページでデータセットグループを選択し、右上の「削除」を選択します。

データセットとスキーマ

Amazon Personalize のデータセットは、データのコンテナです。データセットには次の 5 つのタイプがあります。

- アイテムインタラクション - このデータセットは、ユーザーとアイテム間のインタラクションからの履歴データとリアルタイムデータを保存します。Amazon Personalize では、インタラクションは、記録してからトレーニングデータとしてインポートするイベントです。ドメインデータセットグループとカスタムデータセットグループの両方については、少なくともアイテムインタラクションデータセットを作成する必要があります。
- Users – このデータセットには、ユーザーに関するメタデータが保存されます。これには、年齢、性別、ロイヤルティメンバーシップ、アイテムのタイトルなどの情報が含まれる場合があります。
- Items – このデータセットは、アイテムに関するメタデータを保存します。これには、価格、SKU タイプ、在庫状況などの情報が含まれます。
- Actions - このデータセットには、アクションに関するメタデータが保存されます。アクションは、顧客に推奨できるエンゲージメントアクティビティです。アクションには、モバイルアプリのインストール、会員プロフィールの記入、ロイヤルティプログラムへの加入、プロモーションメールへの登録などがあります。Next-Best-Action レシピには、Actions データセットが必要です。Actions データを使用するカスタムレシピやドメインユースケースは他にありません。
- Action interactions - このデータセットは、ユーザーとアクション間のインタラクションからの履歴データとリアルタイムデータを保存します。Next-Best-Action レシピは、このデータと Actions データセットのデータを使用して、ユーザーにアクションを推奨します。Actions-interactions データを使用するカスタムレシピやドメインユースケースは他にありません。

各データセットグループには、各データセットタイプを 1 つのみ含めることができます。ドメインデータセットグループでは、アクションやアクションインタラクションデータセットなど、次善のアクションリソースを作成することはできません。Amazon Personalize は、ユーザーがデータセットを削除するまで、データをデータセットに保存します。すべてのユースケース (ドメインデータセットグループ) とレシピ (カスタムデータセットグループ) で、インタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

データセットを作成する前に、そのデータセットのスキーマを定義します。スキーマは、Amazon Personalize にデータの構造を知らせ、Amazon Personalize がデータを解析できるようにします。スキーマには名前キーがあり、その値はデータセットタイプと一致する必要があります。スキーマを作成した後は、スキーマに変更を加えることはできなくなります。

ドメインデータセットグループについては、各データセットタイプには、必須フィールドと予約済みキーワードを含むデフォルトのスキーマがあります。データセットを作成するたびに、既存のドメインスキーマを使用するか、既存のデフォルトスキーマを変更して新しいドメインスキーマを作成できます。ドメインにインポートするデータのガイドとして、デフォルトのスキーマを使用します。スキーマを定義してデータセットを作成すると、スキーマに変更を加えることはできなくなります。

データを一括インポートする場合、データはカンマ区切り値 (CSV) 形式で保存する必要があります。CSV ファイルの最初の行には、スキーマと一致する必要がある列ヘッダーが含まれている必要があります。Amazon Personalize 用に一括データをフォーマットする方法については、「データ形式ガイドライン」を参照してください。

トピック

- データセット
- スキーマ
- データ形式ガイドライン

データセット

以下のトピックでは、Amazon Personalize データセットに関する詳細情報を提供します。データセットの種類ごとにデータ要件は異なります。ドメインデータセットグループとカスタムデータセットグループの両方については、トレーニングの前にインタラクションデータに次のものが含まれている必要があります。

- カタログ内のアイテムを操作したユーザーからのインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

ドメインデータセットグループを作成する場合、各データセットにはドメインに応じてさらに要件があります。必要なデータのタイプがわからない場合は、ドメインデータセットグループを作成し、ドメインのデフォルトスキーマをガイドとして使用することをお勧めします。データセットとスキーマの要件の詳細については、「スキーマ」を参照してください。

トピック

アイテムインタラクションデータセット

アイテムインタラクションは、ユーザーとカタログ内のアイテムの間のポジティブなインタラクションイベントです。例えば、映画を見ているユーザー、出品しているユーザー、靴を購入しているユーザーなどです。ユーザーとアイテムとのインタラクションに関するデータをアイテムインタラクションデータセットにインポートします。クリック、視聴、または「いいね」など、複数のイベントタイプを記録できます。

例えば、ユーザーが特定のアイテムをクリックしてからそのアイテムに「いいね！」をした場合、Amazon Personalize にこれらのイベントをトレーニングデータとして使用させることができます。各イベントについて、ユーザーの ID、タイムスタンプ (Unix 時間エポック形式)、ならびにイベントタイプ (クリックおよび「いいね」) を記録します。その後、両方のアイテムインタラクションイベントをアイテムインタラクションデータセットに追加します。

すべてのユースケース (ドメインデータセットグループ) とレシピ (カスタムリソース) について、アイテムインタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

レコメンダーまたはカスタムソリューションを作成するには、少なくともアイテムインタラクションデータセットを作成する必要があります。このセクションでは、Amazon Personalize にインポートできる次のタイプのアイテムインタラクションデータに関する情報を提供します。

トピック

イベントタイプとイベント値のデータ

アイテムインタラクションデータセットは、インタラクションごとにイベントタイプとイベント値データを保存できます。カスタムリソースのみがイベント値データを使用します。

イベントタイプデータ

Amazon Personalize は、クリックデータや購入データなどのイベントタイプデータを使用して、ユーザーの意図と関心を特定します。ドメインレコメンダーを作成する場合、すべてのユースケースにイベントタイプデータが必要です。ユースケースによっては、特定のイベントタイプが必要です。その他のイベントタイプは自由に使用できます。詳細については、ユースケースの選択を参照してください。

カスタムリソースを作成する場合は、トレーニングに使用されるイベントをイベントタイプ別に選択できます。データセットの EVENT_TYPE 列に複数のイベントタイプがあり、カスタムソリューションを設定するときにイベントタイプを指定しない場合、Amazon Personalize は、タイプに関係なく、同じ重みでトレーニングするためにすべてのアイテムインタラクションデータを使用します。詳細については、「トレーニングに使用するアイテムインタラクションデータの選択」を参照してください。

正と負のイベントタイプ

Amazon Personalize は、インタラクションが肯定的であると仮定します。が嫌いなどの負のイベントタイプとのインタラクションは、必ずしもアイテムがユーザーの将来のレコメンデーションに表示されないようにするとは限りません。

以下は、ネガティブなイベントやユーザーの無関心がレコメンデーションに影響を与える方法です。

- すべてのドメインユースケースとUser-Personalizationレシピについて、Amazon Personalize はインプレッションデータを使用できます。アイテムがインプレッションデータに表示され、ユーザーが選択しない場合、そのアイテムがレコメンデーションに表示される可能性は低くなります。詳細については、「インプレッションデータ」を参照してください。
- カスタムリソースを使用し、正と負のイベントタイプをインポートする場合、正のイベントタイプでのみトレーニングし、ユーザーが否定的に操作した項目を除外できます。詳細については、「トレーニングに使用するアイテムインタラクションデータの選択」および「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

イベント値データ (カスタムリソース)

イベント値データは、ユーザーが視聴した映画の割合、または 10 件のうちの評価である場合があります。カスタムソリューションを作成し、イベント値データとイベントタイプデータをインポートする場合、タイプと値に基づいてトレーニングに使用されるレコードを選択できます。ドメインレコメンダーでは、Amazon Personalize はイベント値データを使用しないため、トレーニング前にイベントをフィルタリングすることはできません。

タイプと値に基づいてレコードを選択するには、各イベントのイベントタイプとイベント値を記録します。各イベントで選択する値は、除外するデータや記録するイベントタイプによって異なります。例えば、視聴イベントタイプでは、ユーザーが視聴した動画の割合などのユーザーアクティビティを一致させることができます。

ソリューションを設定する際には、トレーニングからレコードを除外するためのしきい値として特定の値を設定します。例えば、EVENT_TYPE が [watch] (視聴) のイベントの EVENT_VALUE データが、ユーザーが視聴した動画のパーセンテージ (%) である場合であって、イベント値のしきい値を 0.5 に、イベントタイプを [watch] (視聴) に設定する場合、Amazon Personalize は、EVENT_VALUE が 0.5 以上の [watch] (視聴) インタラクションイベントのみを使用してモデルをトレーニングします。

詳細については、「トレーニングに使用するアイテムインタラクションデータの選択」を参照してください。

コンテキストメタデータ

特定のレシピやレコメンデーションユースケースでは、Amazon Personalize は、ユーザーにとって最も関連性の高いアイテムを明らかにする基本的なパターンを識別する際に、コンテキストメタデー

タを使用できます。コンテキストメタデータは、場所やデバイスのタイプなど、イベント時にユーザーの環境で収集するインタラクションデータです。

コンテキストメタデータを含めると、既存のユーザー向けに、よりパーソナライズされたエクスペリエンスを提供できます。例えば、顧客が電話からカタログにアクセスするときとコンピュータからアクセスするときで、ショッピングの態様が異なる場合は、ユーザーのデバイスに関するコンテキストメタデータを含めます。これにより、閲覧方法に基づいて、レコメンデーションの関連性がより高くなります。

さらに、コンテキストメタデータは、新規ユーザーまたは不明ユーザーのコールドスタートフェーズを減らすのに役立ちます。コールドスタートフェーズとは、そのユーザーに関する履歴情報が不足しているために、レコメンデーションエンジンが関連性の低いレコメンデーションを提供する期間をいいます。

ドメインデータセットグループでは、以下のレコメンダーユースケースでコンテキストメタデータを使用できます。

- おすすめ (eコマースドメイン)
- 上位のおすすめ (VIDEO_ON_DEMAND ドメイン)

カスタムリソースについては、コンテキストメタデータを使用するレシピには次のものが含まれます。

- User-Personalization-v2 および User-Personalization
- Personalized-Ranking-v2 および Personalized-Ranking

コンテキスト情報の詳細については、 Machine AWS Machine Learning ブログ記事「コンテキスト情報 を活用して Amazon Personalize のレコメンデーションの関連性を高める」を参照してください。

インプレッションデータ

インプレッションは、ユーザーが特定のアイテムを操作した (例えば、クリックや視聴した) ときに表示されたアイテムのリストです。パーソナライゼーションまたはUser-Personalizationレシピを提供するドメインユースケースを使用する場合、Amazon Personalize はインプレッションデータを使用して探索をガイドできます。

探索では、新しいアイテムまたはアクション、インタラクションがほとんどないアイテムまたはアクション、以前の行動に基づいてユーザーに関連性が低いアイテムまたはアクションなど、通常はユー

ザーにレコメンデーションされる可能性が低いアイテムまたはアクションがレコメンデーションに含まれます。インプレッションデータでアイテムが頻繁に発生するほど、Amazon Personalize がそのアイテムを探索に含める可能性は低くなります。

レコメンダーまたはソリューションを作成すると、Amazon Personalize は常にインプレッションデータをトレーニングから除外します。これは、Amazon Personalize がインプレッションデータを使用してモデルをトレーニングしないためです。代わりに、ユーザーの探索をガイドするレコメンデーションを取得するときに使用します。

インプレッションの値は最大 1,000 文字 (縦棒文字を含む) まで入力できます。ドメインデータセットグループでは、以下のレコメンダーユースケースでインプレッションデータを使用できます。

- おすすめ (eコマースドメイン)
- 上位のおすすめ (VIDEO_ON_DEMAND ドメイン)

探索の詳細については、「探査」を参照してください。Amazon Personalize は、暗黙的なインプレッション および 明示的なインプレッション といった 2 種類のインプレッションをモデル化できます。

暗黙的なインプレッション

暗黙的なインプレッションは、Amazon Personalize から取得した、ユーザーに表示するレコメンデーションです。(GetRecommendations および GetPersonalizedRanking 操作によって返される) `RecommendationId` を将来の PutEvents リクエストの入力として含めることにより、それらをレコメンデーションワークフローに統合できます。Amazon Personalize は、レコメンデーションデータに基づいて暗黙的なインプレッションを派生させます。

例えば、ストリーミング動画に関するレコメンデーションを提供するアプリケーションがあるとします。暗黙的なインプレッションを使用するレコメンデーションワークフローは次のようになります。

1. Amazon Personalize の the section called “GetRecommendations” API 操作を使用して、いずれかのユーザー向けに動画のレコメンデーションをリクエストします。
2. Amazon Personalize は、モデル (ソリューションバージョン) を使用してユーザー向けのレコメンデーションを生成し、それらを `recommendationId` とともに API 応答で返します。
3. アプリケーションでユーザーに動画のレコメンデーションを表示します。
4. ユーザーが動画を操作する (例えば、クリックする) 際に、PutEvents API に対するコールで選択内容を記録し、パラメータとして `recommendationId` を含めます。コードサンプルについては、「インプレッションデータの記録」を参照してください。

5. Amazon Personalize は、`recommendationId` を使用して以前の動画のレコメンデーションからインプレッションデータを導き出し、その後にインプレッションデータを使用して探索をガイドします。将来のレコメンデーションには、インタラクションデータまたは関連性が少ない新しい動画が含まれます。

   暗黙的なインプレッションデータを使用したイベントの記録の詳細については、「インプレッションデータの記録」を参照してください。

明示的なインプレッション

明示的なインプレッションとは、手動で記録して Amazon Personalize に送信するインプレッションです。明示的なインプレッションを使用して、Amazon Personalize の結果を操作します。アイテムの順序はいかなる影響も及ぼしません。

例えば、靴に関するレコメンデーションを提供するショッピングアプリケーションがあるとします。現在在庫のある靴のみをお勧めする場合は、明示的なインプレッションを使用してこれらのアイテムを指定できます。明示的なインプレッションを使用するレコメンデーションワークフローは次のようになります。

1. Amazon Personalize の the section called “GetRecommendations” API を使用して、いずれかのユーザー向けにレコメンデーションをリクエストします。
2. Amazon Personalize は、モデル (ソリューションバージョン) を使用してユーザー向けのレコメンデーションを生成し、それらを API 応答で返します。
3. 推奨される靴のうち、在庫がある靴のみをユーザーに表示します。
4. リアルタイムの増分データインポートについては、ユーザーが 1 組の靴を操作する (例えば、クリックする) ときに、PutEvents API に対するコールで選択を記録し、在庫がある推奨アイテムを `impression` パラメータでリストします。コードサンプルについては、「インプレッションデータの記録」を参照してください。

   履歴アイテムインタラクションデータにインプレッションをインポートするために、明示的なインプレッションを csv ファイルにリストし、各アイテムを「|」文字で区切ることができます。縦棒文字は 1,000 文字の制限に含まれます。例については、「明示的なインプレッションのフォーマット」を参照してください。
5. Amazon Personalize は、インプレッションデータを使用して探索をガイドします。将来のレコメンデーションには、インタラクションデータまたは関連性が少ない新しい靴が含まれます。

ユーザーデータセット

Amazon Personalize にインポートできるユーザーデータには、性別やロイヤルティメンバーシップなど、ユーザーに関する数値およびカテゴリのメタデータが含まれます。ユーザーに関するメタデータを Amazon Personalize の Users データセットにインポートします。メタデータ列の最大数は 25 です。

このトピックでは、次のタイプのユーザーデータに関する情報を提供します。

トピック

カテゴリ別メタデータ

一部のレシピと、VIDEO_ON_DEMAND ドメインおよび ECOMMERCE ドメインの両方で、Amazon Personalize は、ユーザーにとって最も関連性の高いアイテムを明らかにする基本的なパターンを識別する際に、ユーザーの性別やメンバーシップステータスなどのカテゴリメタデータを使用します。ユースケースに基づいて独自の値の範囲を定義します。カテゴリメタデータはどの言語でもかまいません。

すべてのレシピとドメインで、カテゴリメタデータをインポートし、それを使用してユーザーの属性に基づいてレコメンデーションをフィルタリングできます。フィルタリングのレコメンデーションについては、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つユーザーがいる場合、データセットのインポートジョブは失敗します。

カスタムデータセットグループおよびカスタムソリューションについては、カテゴリメタデータを使用するレシピには次のものが含まれます。

- User-Personalization-v2 および User-Personalization
- Personalized-Ranking-v2 および Personalized-Ranking
- Similar-Items

製品データセット

Amazon Personalize にインポートできるアイテムデータには、作成タイムスタンプ、料金、ジャンル、説明、利用可否などの数値およびカテゴリのメタデータが含まれます。アイテムに関するメタデータを Amazon Personalize の Items データセットにインポートします。

Amazon Personalize は、トレーニング時にアイテムのタイトルや作成者データなど、カテゴリ以外の文字列アイテムデータを使用しません。ただし、Amazon Personalize の一部の機能では、このデータを使用してレコメンデーションを強化しています。詳細については、「非カテゴリ別文字列データ」を参照してください。

メタデータ列の最大数は 100 です。Amazon Personalize がトレーニング中に考慮する項目の最大数は、ユースケースまたはレシピによって異なります。レコメンデーションには、トレーニング中に考慮された項目のみを表示できます。

- User-Personalization-v2 または Personalized-Ranking-v2 の場合、トレーニング中にモデルによって考慮される項目の最大数は 500 万です。これらの項目は、アイテムとアイテムインタラクションデータセットの両方からのものです。
- User-Personalization-v2 と Personalized-Ranking-v2 以外のすべてのドメインユースケースとカスタムレシピでは、トレーニング中およびレコメンデーションの生成中にモデルによって考慮されるアイテムの最大数は 750,000 です。

レシピ要件の詳細については、「」を参照してくださいレシピの選択。

このトピックでは、次のタイプのアイテムデータに関する情報を提供します。

トピック

作成のタイムスタンプデータ

Amazon Personalize は、作成タイムスタンプデータ (Unix エポック時間形式 (秒)) を使用してアイテムが存在するようになってからの期間を計算し、それに応じてレコメンデーションを調整します。

1 つ以上のアイテムについて作成タイムスタンプのデータが欠落している場合、Amazon Personalize は、インタラクションデータがある場合はこの情報からこの情報を推測し、アイテムの最も古いインタラクションデータのタイムスタンプをアイテムの作成タイムスタンプとして使用します。アイテムにインタラクションデータがない場合、その作成タイムスタンプはトレーニングセット内の最新のインタラクションのタイムスタンプとして設定され、Amazon Personalize はそれを新しいアイテムとみなします。

カテゴリ別メタデータ

特定のレシピとドメインでは、Amazon Personalize は、ユーザーにとって最も関連性の高いアイテムを明らかにする基本的なパターンを識別する際に、アイテムのジャンルや色などのカテゴリメタデータを使用します。ユースケースに基づいて独自の値の範囲を定義します。カテゴリメタデータはどの言語でもかまいません。

すべてのレシピとドメインで、カテゴリデータをインポートし、それを使用してアイテムの属性に基づいてレコメンデーションをフィルタリングできます。フィルタリングのレコメンデーションについては、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

カテゴリ値には、最大 1,000 文字まで入力できます。1,000 文字を超えるカテゴリ値を持つアイテムがある場合、データセットのインポートジョブは失敗します。

ドメインデータセットグループについては、VIDEO_ON_DEMAND ドメインと ECOMMERCE ドメインの両方がカテゴリメタデータを使用します。カスタムデータセットグループおよびカスタムソリューションについては、カテゴリメタデータを使用するレシピには次のものが含まれます。

- User-Personalization-v2 および User-Personalization
- Personalized-Ranking-v2 および Personalized-Ranking
- Similar-Items
- Item-Affinity
- Item-Attribute-Affinity

非構造化テキストメタデータ

特定のレシピとドメインを使用すると、Amazon Personalize は、製品の説明、製品のレビュー、映画のあらすじなど、非構造化テキストメタデータから有意義な情報を抽出できます。Amazon Personalize は、非構造化テキストを使用して、特にアイテムが新しい場合やインタラクションデータが少ない場合に、ユーザーに関連するアイテムを識別します。Items データセットに非構造化テキストデータを含めて、カタログ内の新しいアイテムのクリック率とコンバージョン率を高めます。

非構造化データを使用するには、タイプ `string` のフィールドをアイテムスキーマに追加し、フィールドの `textual` 属性を `true` に設定します。最大 1 つのテキストフィールドしか追加できません。その後、テキストデータをバルク CSV ファイルと増分アイテムインポートに含めます。

一括 CSV ファイルの場合は、テキストを二重引用符で囲み、新しい行文字を削除します。\ 文字を使用して、データ内の二重引用符または \ 文字をエスケープします。非構造化テキストデータのフィールドを持つ Items スキーマの例については、「Items データセットのスキーマの例 (カスタム)」を参照してください。Amazon Personalize では、テキストフィールドは文字数制限で切り捨てられます。テキスト内の最も関連性の高い情報がフィールドの先頭にあることを確認してください。Amazon Personalize へのデータのインポートについては、「ステップ 2: データの準備とインポート」を参照してください。

非構造化テキストの値は、中国語と日本語を除くすべての言語で最大 20,000 文字を使用できます。中国語と日本語については、最大 7,000 文字を使用できます。Amazon Personalize は、文字数制限を超える値を上限値になるように切り捨てます。

テキストは次の言語で入力できます。

- 簡体字中国語
- 繁体字中国語
- 英語
- フランス語
- ドイツ語
- 日本語
- ポルトガル語
- スペイン語

非構造化テキストアイテムは複数の言語で送信できますが、各アイテムのテキストは 1 言語のみである必要があります。

ドメインデータセットグループについては、VIDEO_ON_DEMAND ドメインと ECOMMERCE ドメインの両方がテキストメタデータを使用します。カスタムデータセットグループおよびカスタムソリューションについては、テキストメタデータを使用するレシピには次のものが含まれます。

- User-Personalization-v2 および User-Personalization
- Personalized-Ranking-v2 および Personalized-Ranking

- Similar-Items
- Item-Affinity
- Item-Attribute-Affinity

非カテゴリ別文字列データ

アイテム ID 以外、Amazon Personalize は、トレーニング時にアイテムのタイトルや作成者データなど、カテゴリ別以外の文字列アイテムデータを使用しません。ただし、Amazon Personalize の以下の機能では使用することがあります。

- Amazon Personalize では、カテゴリ別以外の文字列値を含むアイテムメタデータをレコメンデーションに含めることができます。メタデータを使用して、映画のレコメンデーションカルーセルに監督の名前を追加するなど、ユーザーインターフェイスのレコメンデーションを充実させることができます。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。
- Similar-Items を使用すると、テーマ付きのレコメンデーションを一括生成できます。テーマを使用してレコメンデーションを一括生成する場合、一括推論ジョブでアイテム名列を指定する必要があります。詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。
- カテゴリ別以外の文字列データに基づいて、レコメンデーションにアイテムを含めたり、レコメンデーションからアイテムを削除したりするフィルターを作成できます。フィルターの詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

Actions データセット

アクションは、ユーザーに推奨できるエンゲージメントアクティビティまたは収益創出アクティビティです。アクションには、モバイルアプリのインストール、会員プロフィールの記入、ロイヤルティプログラムへの加入、プロモーションメールへの登録などがあります。アクションに関するデータを Amazon Personalize の Actions データセットにインポートします。アクションのデータには、アクションの ID、アクションの推定価値、アクションの有効期限タイムスタンプなどがあります。

モデルトレーニング中に、Amazon Personalize は最大 1000 アクションを考慮します。1000 を超えるアクションをインポートした場合、Amazon Personalize は、新しいアクション (最近追加し、インタラクションのないアクション) と最近のインタラクションデータがある既存のアクションを優先して、トレーニングに含めるアクションを決定します。

> **Note**
>
> ドメインデータセットグループでは、アクションやアクションインタラクションデータセットなど、次善のアクションリソースを作成することはできません。

最大列数は 10 です。このトピックでは、以下のタイプのアクションデータに関する情報を提供します。

トピック

- アクション有効期限のタイムスタンプデータ
- 繰り返し頻度データ
- [Value data] (値のデータ)
- 作成のタイムスタンプデータ
- カテゴリ別メタデータ

アクション有効期限のタイムスタンプデータ

アクションの有効期限タイムスタンプは、アクションが無効になる日付を指定します。アクション有効期限のタイムスタンプデータは Unix エポックタイム形式 (秒単位) で指定します。アクションの有効期限が切れている場合、Amazon Personalize はそのアクションをレコメンデーションに含めません。

アクションがレコメンデーションに表示されるのを特定の期間に制限する場合は、アクションの有効期限タイムスタンプを指定してください。例えば、特定の月にメンバーシップドライブを実行するアプリケーションがあるとします。その月の月末に登録アクションの有効期限タイムスタンプを設定するとよいでしょう。この日付になると、Amazon Personalize はこのアクションの推奨を自動的に停止します。

新しいアクションの有効期限タイムスタンプを過去の時刻に設定したり、アクションのタイムスタンプを過去の時刻に更新したりすると、そのアクションがレコメンデーションから削除されるまでに最大 2 時間かかることがあります。

繰り返し頻度データ

繰り返し頻度データは、Action インタラクションデータセット内のユーザーの履歴に基づいて、ユーザーが特定のアクションとやり取りした後、Amazon Personalize がその特定のアクションを推

奨するまで待機する日数を指定します。アクションの繰り返し頻度を日単位で指定できます。最大値は 30 日です。

例えば、各ユーザーがアカウントとプロファイルを作成する e コマースアプリケーションがあるとします。`complete profile` アクションがあり、ユーザーがそのアクションを操作してから 1 週間待ってから再び推奨する場合は、アクションの `REPEAT_FREQENCY` として 7 日を指定します。7 日後、Amazon Personalize はそのアクションの推奨を検討し始めます。

アクションの繰り返し頻度を指定しなかった場合、Amazon Personalize はレコメンデーションに表示される回数に制限を設けません。

[Value data] (値のデータ)

バリューデータとは、各アクションのビジネス価値または重要性です。アクションの `value` は 1 ~ 10 の範囲であり、10 はデータセット内で最も価値のあるアクションです。

例えば、ベーシックサブスクリプションへの登録用とプレミアムサービスへの登録用の 2 つのアクションがあるとします。ベーシックサービスについては 5 の値を指定し、プレミアムサービスについては `10` の値を指定することができます。

Amazon Personalize は、ユーザーに推奨する最適なアクションを決定する際に、値データを 1 つの入力として使用します。例えば、ユーザーがあるアクションを実行する可能性と別のアクションを実行する可能性が同じ場合、Amazon Personalize は、値が最も高いアクションをレコメンデーションで上位にランク付けします。

作成のタイムスタンプデータ

Amazon Personalize は、作成タイムスタンプデータ (Unix エポック時間形式 (秒)) を使用して、アクションが存在するようになってからの期間を計算し、それに応じてレコメンデーションを調整します。

作成タイムスタンプデータがない場合、Amazon Personalize はすべてのアクションインタラクションデータからこの情報を推測します。アクションの最も古いインタラクションデータのタイムスタンプをアクションの作成タイムスタンプとして使用します。アイテムにインタラクションデータがない場合、その作成タイムスタンプはトレーニングセット内の最新のインタラクションのタイムスタンプとして設定され、Amazon Personalize はそれを新しいアクションとみなします。

カテゴリ別メタデータ

Amazon Personalize は、ユーザーにとって最適なアクションを明らかにする基本的なパターンを識別する際に、季節性やアクションの排他性などのカテゴリ別メタデータを使用します。ユースケースに基づいて独自の値の範囲を定義します。カテゴリメタデータはどの言語でもかまいません。

カテゴリ別データをインポートし、それを使用してアクションの属性に基づいてレコメンデーションをフィルタリングできます。フィルタリングのレコメンデーションについては、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

カテゴリ値には、最大 1,000 文字まで入力できます。1,000 文字を超えるカテゴリ値を持つアクションがある場合、データセットのインポートジョブは失敗します。

Action インタラクションデータセット

アクションインタラクションは Actions データセット内のユーザーとアクションが関与するインタラクションです。アクションインタラクションを Amazon Personalize の Action インタラクションデータセットにインポートします。各アクションインタラクションは、userID、actionID、タイムスタンプ、イベントタイプ、およびインタラクションに関するその他のデータ (カテゴリ別メタデータなど) で構成されます。

例えば、Actions データセットに登録アクションがあり、ユーザーがこのアクションを実行した場合、ユーザーの ID、アクションの ID、タイムスタンプを記録し、イベントタイプとして TAKEN を記録します。アクションインタラクションイベントは、データセットインポートジョブで一括してインポートすることも、PutActionInteractions API オペレーションを使用してリアルタイムでストリーミングすることもできます。データのインポートの詳細については、「ステップ 2: データの準備とインポート」を参照してください。

> **Note**
>
> ドメインデータセットグループでは、アクションやアクションインタラクションデータセットなど、次善のアクションリソースを作成することはできません。

PERSONALIZED_ACTIONS カスタムレシピを使用すると、Amazon Personalize は Action インタラクションデータセット内のデータを入力として使用して、ユーザーが実行する可能性が最も高いアクションを予測します。アクションインタラクションデータには最小要件はありません。質の高いアクション推奨のために、このデータをインポートすることをお勧めします。アクションインタラクショ

ンデータがない場合は、PutActionInteractions API オペレーションを使用して空のアクションインタラクションデータセットを作成し、顧客のアクションとのインタラクションを記録できます。

アクションインタラクションデータをインポートするまで、Amazon Personalize はパーソナライゼーションなしで のアクションを推奨し、傾向スコアは 0.0 です。アクションに次のものがあると、アクションにスコアが付けられます。

- TAKEN イベントタイプと少なくとも 50 件のアクションインタラクション。
- NOT_TAKEN または VIEWED イベントタイプとのアクションインタラクションが 50 回以上ある。

これらのアクションインタラクションは、最新のソリューションバージョントレーニングに存在し、アクションインタラクションデータセットの最新のインタラクションタイムスタンプから 6 週間以内に発生する必要があります。

以下のトピックでは、Amazon Personalize が使用できるアクションインタラクションデータについての詳細情報を示します。

トピック

- イベントタイプデータ

イベントタイプデータ

Amazon Personalize では、イベントタイプデータのパターンを使用して、ユーザーが実行する可能性が最も高いアクションを特定できます。例えば、お客様が E メール購読アクション (NOT_TAKEN イベントタイプで示される) を頻繁に無視する場合、Amazon Personalize は、このタイプのアクションを少なくするようにレコメンデーションを調整することがあります。

アクションインタラクションイベントには、以下のイベントタイプのみを使用できます。Amazon Personalize はこれらのイベントを使用してユーザーについて学習し、次に推奨するアクションを計算します。

- 実行済み - ユーザーが推奨されるアクションを実行したときの実行済みイベントを記録します。
- 未実行 - ユーザーが閲覧後にアクションを実行しないという意図的な選択をした場合に、未実行イベントを記録します。例えば、ユーザーに対してアクションを表示したときに、ユーザーがいいえを選択した場合です。未実行イベントは、お客様がそのアクションに興味がないことを示している可能性があります。

- 閲覧済み - ユーザーがアクションを実行するかしないかの選択をする前にアクションを示したときに、閲覧済みイベントを記録します。Amazon Personalize は、閲覧イベントを使用してユーザーの関心を学習します。例えば、ユーザーが表示されたアクションを実行しなかった場合、このユーザーは今後、このアクションに興味を持たなくなる可能性があります。

スキーマ

スキーマは、Amazon Personalize にデータの構造を知らせ、Amazon Personalize がデータを解析できるようにします。スキーマには名前キーがあり、その値はデータセットタイプと一致する必要があります。スキーマを作成した後は、スキーマに変更を加えることはできなくなります。

ドメインデータセットグループについては、各データセットタイプには、必須フィールドと予約済みキーワードを含むデフォルトのスキーマがあります。データセットを作成するたびに、既存のドメインスキーマを使用するか、既存のデフォルトスキーマを変更して新しいドメインスキーマを作成できます。ドメインにインポートするデータのガイドとして、デフォルトのスキーマを使用します。スキーマを定義してデータセットを作成すると、スキーマに変更を加えることはできなくなります。

トピック

- スキーマ形式の要件
- ドメインデータセットとスキーマ
- カスタムデータセットとスキーマ
- SDK for Python (Boto3) を使用したスキーマの作成

スキーマ形式の要件

ドメインデータセットグループまたはカスタムデータセットグループのいずれかのデータセットのスキーマを作成するときは、次のガイドラインに従う必要があります。

- スキーマは Avro 形式で定義する必要があります。サポートされている Avro データ型については、「スキーマのデータ型」を参照してください。
- スキーマフィールドは任意の順序で表示できますが、CSV ファイル内の対応する列ヘッダーの順序と一致する必要があります。
- スキーマは、ネストされた構造のないフラットな JSON ファイルである必要があります。例えば、フィールドを複数のサブフィールドの親にすることはできません。

- Amazon Personalize のスキーマは、配列やマップなどの複雑なタイプをサポートしません。
- スキーマフィールドには一意の英数字名が必要です。例えば、GENRES_FIELD_1 フィールドと GENRESFIELD1 フィールドの両方を追加することはできません。
- 必須フィールドを必須データ型として定義する必要があります。予約済みカテゴリ文字列フィールドでは categorical 属性を true に設定する必要がありますが、予約済み文字列フィールドはカテゴリに設定できません。キーワードをデータに含めることはできません。
- タイプ string の独自のメタデータフィールドを追加して、Amazon Personalize がトレーニングの際にそれを使用するように希望する場合は、categorical 属性または textual 属性を含める必要があります (アイテムスキーマのみがテキスト属性のフィールドをサポートします)。
- Amazon Personalize は、テーマを生成したり、レコメンデーションでメタデータを返したり、レコメンデーションをフィルタリングしたりするときに、アイテム名列などの非カテゴリ別文字列列を使用できます。詳細については、「非カテゴリ別文字列データ」を参照してください。
- Amazon Personalize は、レコメンデーションのトレーニングやフィルタリングに boolean タイプデータを使用しません。Amazon Personalize でトレーニングやフィルタリングの際にブーリアンデータを使用するには、文字列 型のフィールドを使用し、データ内の "True" と "False" の値を使用します。または、int もしくは long 型と 0 と 1 の値を使用できます。
- テキストフィールドは string タイプで、textual 属性は true に設定されている必要があります。非構造化テキストデータの詳細については、「非構造化テキストメタデータ」を参照してください。

ドメインデータセットグループのデータセットには、ドメインとデータセットタイプの両方に基づいて追加要件があります。カスタムデータセットグループのデータセットには、タイプに応じて異なる要件があります。

スキーマのデータ型

Amazon Personalize のスキーマは、フィールド向けに次の Avro タイプをサポートします。

- フロート
- double
- 整数
- long
- string
- ブール値

- null

一部の必須フィールドと予約済みフィールドは NULL データをサポートしています。フィールドに null タイプを追加すると、不完全なデータ (例えば、空白の値を持つメタデータ) を使用して、レコメンデーションを生成できます。どのフィールドが NULL データをサポートするかについては、「ドメインデータセットとスキーマ」または「カスタムデータセットとスキーマ」を参照してください。次の例は、GENDER フィールドに null タイプを追加する方法を示しています。

```
{
  "name": "GENDER",
  "type": [
    "null",
    "string"
  ],
  "categorical": true
}
```

ドメインデータセットとスキーマ

ドメインデータセットグループを作成する場合、選択したドメインによってデータセットとスキーマの要件が決まります。各ドメインには、各データセットタイプについてデフォルトのスキーマがあります。

データセットを作成する場合、デフォルトスキーマを使用するか、デフォルトスキーマに基づいて新しいスキーマを作成できます。収集して各データセットタイプにインポートするデータに関するガイドとして、デフォルトのスキーマを使用します。次のトピックでは、各ドメインのデータセットとスキーマの要件について説明します。

Amazon Personalize インポートできるデータの種類については、「Amazon Personalize が使用できるデータの種類」を参照してください。

フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

トピック

- VIDEO_ON_DEMAND データセットとスキーマ
- ECOMMERCE データセットとスキーマ

VIDEO_ON_DEMAND データセットとスキーマ

VIDEO_ON_DEMAND ドメイン用にドメインデータセットグループを作成する場合、各データセットタイプには、VIDEO_ON_DEMAND 固有の一連の必須フィールドおよび推奨フィールドを含むデフォルトのスキーマがあります。デフォルトスキーマを使用するか、デフォルトスキーマに基づいて新しいスキーマを作成できます。インポートするデータは、形式とタイプがスキーマと一致している必要があります。以下のセクションにリストされているデフォルトのドメインスキーマをガイドとして使用して、VIDEO_ON_DEMAND ベースのレコメンダーを作成するためにインポートするデータを決定します。

フィールドは自由に追加できます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

次のトピックでは、VIDEO_ON_DEMAND ドメインの各データセットの必須フィールドと推奨フィールドに関する情報を提供します。各データセットのセクションには、JSON 形式のデフォルトの VIDEO_ON_DEMAND スキーマが含まれています。

トピック

- VIDEO_ON_DEMAND ドメインデータセットとスキーマの要件
- アイテムインタラクションデータセットの要件 (VIDEO_ON_DEMAND ドメイン)
- Users データセットの要件 (VIDEO_ON_DEMAND ドメイン)
- Items データセットの要件 (VIDEO_ON_DEMAND ドメイン)

VIDEO_ON_DEMAND ドメインデータセットとスキーマの要件

各データセットタイプには、次の必須フィールドと予約済みキーワードがあります。予約済みのキーワードはオプションの非メタデータフィールドです。これらのフィールドは、使用時に必須のデータ型としてフィールドを定義する必要があるため、予約済みとみなされます。予約済みカテゴリ文字列フィールドは `categorical` が `true` に設定されている必要がありますが、予約済み文字列フィールドはカテゴリに設定できません。キーワードをデータに含めることはできません。

データセットタイプ	必須フィールド	予約済みキーワード
アイテムインタラクション (デフォルトのスキーマ)	USER_ID (string) ITEM_ID (string) TIMESTAMP (long) EVENT_TYPE (string、ならびにユースケースに応じて、Watch および Click イベントタイプ)	EVENT_VALUE (float、null) IMPRESSION (string、null) RECOMMENDATION_ID (string、null) EVENT_ATTRIBUTION_SOURCE (string、null)
ユーザー (デフォルトのスキーマ)	USER_ID (string) 1 つのメタデータフィールド (カテゴリ別 string または数値)	SUBSCRIPTION_MODEL (カテゴリ string、null)
アイテム (デフォルトのスキーマ)	ITEM_ID (string) CREATION_TIMESTAMP (long) GENRES (カテゴリ string)	PRICE (float、null) DURATION (float、null) GENRE_L2 (カテゴリ別 string、null) GENRE_L3 (カテゴリ別 string、null) AVERAGE_RATING (float、null) 商品説明 (テキスト string、null) CONTENT_OWNER (カテゴリ string、null)

データセットタイプ	必須フィールド	予約済みキーワード
		CONTENT_CLASSIFICATION (カテゴリ別 string、null)

アイテムインタラクションデータセットの要件 (VIDEO_ON_DEMAND ドメイン)

アイテムインタラクションデータセットは、ユーザーとアイテム間のインタラクションからの履歴データとリアルタイムデータを VIDEO_ON_DEMAND カタログに保存します。インタラクションデータセットに保存できるデータの種類の詳細については、「アイテムインタラクションデータセット」を参照してください。

すべてのユースケースについてアイテムインタラクションデータセットを作成する必要があり、スキーマには以下のフィールドが必要です。

- USER_ID (string)
- ITEM_ID string
- TIMESTAMP (long)
- EVENT_TYPE (string、ならびにユースケースに応じて、Watch および Click イベントタイプ)

スキーマには、次の予約済みキーワードを含めることもできます。

- EVENT_VALUE (float、null)
- IMPRESSION (string、null)
- RECOMMENDATION_ID (string、null)

ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。VIDEO_ON_DEMAND ドメインのアイテムインタラクションデータセットのデフォルトスキーマの例については、「デフォルトのインタラクションスキーマ (VIDEO_ON_DEMAND ドメイン)」を参照してください。

視聴した動画の割合 (%) など、イベントに関する値のデータがある場合は、オプションで予約済みキーワード EVENT_VALUE を追加します。明示的および暗黙的なインプレッションデータを含める場合は、オプションで予約済みキーワード IMPRESSION を追加します。インプレッションデータの記録の詳細については、「インプレッションデータ」を参照してください。

アイテムインタラクションデータセットに追加できるオプションのメタデータフィールドの最大合計数は、データ内の個別のイベントタイプの合計数と合わせて 10 です。このカウントに含まれるメタデータフィールドは、スキーマに追加するカスタムメタデータフィールドとともに、EVENT_TYPE、EVENT_VALUE フィールドです。IMPRESSION などの予約済みフィールドを除くメタデータフィールドの最大数は 5 です。カテゴリ値には、最大 1000 文字を使用できます。1000 を超えるカテゴリ値とのインタラクションがある場合、データセットのインポートジョブは失敗します。

VIDEO_ON_DEMAND ドメインのアイテムインタラクションデータセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

デフォルトのインタラクションスキーマ (VIDEO_ON_DEMAND ドメイン)

アイテムインタラクションデータセットのデフォルトの VIDEO_ON_DEMAND ドメインスキーマを次に示します。

```
{

  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
```

```
}
```

Users データセットの要件 (VIDEO_ON_DEMAND ドメイン)

Users データセットには、ユーザーに関するメタデータが保存されます。これには、各アイテムの年齢、性別、ロイヤルティメンバーシップなどの情報が含まれる場合があります。Amazon Personalize にインポートできるデータの種類については、「ユーザーデータセット」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

VIDEO_ON_DEMAND ユースケースのすべてにおいて、Users データセットはオプションとなります。ユーザーデータがある場合は、最も関連性の高いレコメンデーションを取得するために当該データを作成することをお勧めします。Users データセットを作成する場合、スキーマには次のフィールドが含まれている必要があります。

- USER_ID
- 1 つのメタデータフィールド (カテゴリ `string` または数値)

ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。VIDEO_ON_DEMAND ドメインの Users データセットのデフォルトスキーマの例については、「デフォルトのユーザースキーマ (VIDEO_ON_DEMAND ドメイン)」を参照してください。

デフォルトのスキーマには `SUBSCRIPTION_MODEL` フィールドが含まれています。このフィールドはオプションの予約済みキーワードであり、カテゴリが `true` に設定された `string` のタイプである必要があります。最良のレコメンデーションを取得するには、データ内の各ユーザーに関するサブスクリプションモデル情報がある場合は、このフィールドをスキーマに保持することをお勧めします。インポートするデータは、スキーマと一致している必要があります。

カテゴリ別データを使用する

カテゴリデータを使用するには、タイプ `string` のフィールドを追加し、スキーマでフィールドのカテゴリ属性を `true` に設定します。その後、カテゴリデータをバルク CSV ファイルと個別アイテムインポートに含めます。複数のカテゴリを持つユーザーについては、バーティカルバー「|」を使用して各値を区切ります。例えば、SUBSCRIPTION_MODEL フィールドについて、ユーザーのデータは student|monthly|discount である場合があります。

カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つユーザーがいる場合、データセットのインポートジョブは失敗します。

デフォルトのユーザースキーマ (VIDEO_ON_DEMAND ドメイン)

Users データセットのデフォルトの VIDEO_ON_DEMAND ドメインスキーマを次に示します。

```
{

  "type": "record",
  "name": "Users",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "SUBSCRIPTION_MODEL",
          "type": "string",
          "categorical": true
      }
  ],
  "version": "1.0"
}
```

Items データセットの要件 (VIDEO_ON_DEMAND ドメイン)

製品データセットには、カタログ内のアイテムに関するメタデータが保存されます。これには、各アイテムについての料金、ジャンル、利用可否などの情報が含まれる場合があります。Amazon Personalize が使用できるアイテムデータの種類については、「製品データセット」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

一部のユースケースでは製品データセットが必要です (「VIDEO_ON_DEMAND ユースケース」を参照)。オプションである場合でも、最も関連性の高いレコメンデーションを取得するために当該レコメンデーションを作成することをお勧めします。Items データセットを作成する場合、スキーマには次のフィールドが含まれている必要があります。

- ITEM_ID

- GENRES (カテゴリ string)
- CREATION_TIMESTAMP (Unix エポック時間形式)

スキーマには、次の予約済みキーワードを含めることもできます。各キーワードには、必要なデータ型と NULL データをサポートするかどうかが記載されています。NULL タイプの追加は任意です。

- PRICE (浮動小数点)
- DURATION (浮動小数点)
- GENRE_L2 (カテゴリ別 string、null)
- GENRE_L3 (カテゴリ別 string、null)
- AVERAGE_RATING (float、null)
- 製品説明 (テキスト string、null)
- コンテンツ所有者 (カテゴリ別 string、null): 動画を所有している会社。例えば、値は HBO、Paramount、NBC などです。
- コンテンツ分類 (カテゴリ別 string、null): コンテンツのレーティング。例えば、値は G、PG、PG-13、R、NC-17、および未評価の場合があります。

最良のレコメンデーションを取得するには、これらのフィールドをデータと同じ数だけスキーマに保持することをお勧めします。インポートするデータは、スキーマと一致している必要があります。メタデータ列の最大数は 100 です。ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとしてリストされておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

複数のマルチレベルカテゴリがあるアイテムには、予約済みキーワード GENRE_L2 および GENRE_L2 および GENRE_L3 および GENRE_L2 および GENRE_L3 を使用します。詳細については、「カテゴリ別データを使用する」を参照してください。テキストおよびカテゴリのメタデータについては、「製品データセット」を参照してください。ECOMMERCE ドメインの Items データセットのデフォルトスキーマの例については、「デフォルトのアイテムスキーマ (VIDEO_ON_DEMAND ドメイン)」を参照してください。

カテゴリ別データを使用する

カテゴリデータを使用するには、タイプ string のフィールドを追加し、スキーマでフィールドのカテゴリ属性を true に設定します。その後、カテゴリデータをバルク CSV ファイルと個別アイテムインポートに含めます。カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つアイテムがある場合、データセットのインポートジョブは失敗します。

複数のカテゴリを持つアイテムについては、バーティカルバー「|」を使用して各値を区切ります。例えば、GENRES フィールドについて、アイテムのデータは `Action|Crime|Biopic` である場合があります。複数のレベルのカテゴリデータがあり、一部のアイテムで階層内の各レベルに複数のカテゴリがあるアイテムがある場合は、各レベルにフィールドを追加し、フィールド名の後にレベルインジケータ GENRE_L2、GENRE_L2、GENRE_L2、GENRE_L3 を追加します。これにより、アイテムが複数のマルチレベルカテゴリに属している場合でも、サブカテゴリに基づいてレコメンデーションをフィルタ処理できます。例えば、動画にはカテゴリレベルごとに次のデータが含まれている場合があります。

- GENRES: アクション|アドベンチャー
- GENRE_L2: 犯罪|西部劇
- GENRE_L3: 伝記映画

この例では、動画はアクション > 犯罪 > バイオピックの階層、およびアドベンチャー > ウエスタン > バイオピックの階層です。L3 まで使用することをお勧めしますが、必要に応じてもっと多くのレベルを使用できます。フィルターの作成と使用については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

デフォルトのアイテムスキーマ (VIDEO_ON_DEMAND ドメイン)

VIDEO_ON_DEMAND ドメインの Items データセットのデフォルトスキーマを次に示します。

```
{
  "type": "record",
  "name": "Items",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "GENRES",
      "type": [
        "string"
      ],
      "categorical": true
    },
    {
      "name": "CREATION_TIMESTAMP",
```

```
      "type": "long"
    }
  ],
  "version": "1.0"
}
```

ECOMMERCE データセットとスキーマ

ECOMMERCE ドメイン用にドメインデータセットグループを作成する場合、各データセットタイプには、ECOMMERCE 固有の一連の必須フィールドおよび推奨フィールドを含むデフォルトのスキーマがあります。デフォルトスキーマを使用するか、デフォルトスキーマに基づいて新しいスキーマを作成できます。インポートするデータは、形式とタイプがスキーマと一致している必要があります。以下のセクションにリストされているデフォルトのドメインスキーマをガイドとして使用して、ECOMMERCE ベースのレコメンダーを作成するためにインポートするデータを決定します。

フィールドは自由に追加できます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

次のトピックでは、ECOMMERCE ドメインの各データセットの必須フィールドと推奨フィールドに関する情報を提供します。各データセットのセクションには、JSON 形式のデフォルトの ECOMMERCE スキーマが含まれています。

トピック

- ECOMMERCE ドメインのデータセットとスキーマの要件
- アイテムインタラクションデータセットの要件 (ECOMMERCE ドメイン)
- Users データセットの要件 (ECOMMERCE ドメイン)
- Items データセットの要件 (ECOMMERCE ドメイン)

ECOMMERCE ドメインのデータセットとスキーマの要件

各データセットタイプには、次の必須フィールドと予約済みキーワードがあります。予約済みのキーワードはオプションの非メタデータフィールドです。これらのフィールドは、使用時に必須のデータ型としてフィールドを定義する必要があるため、予約済みとみなされます。予約済みカテゴリ文字列

フィールドは categorical が true に設定されている必要がありますが、予約済み文字列フィールドはカテゴリに設定できません。キーワードをデータに含めることはできません。

データセットタイプ	必須フィールド	予約済みキーワード
アイテムインタラクション (デフォルトのスキーマ)	USER_ID (string) ITEM_ID (string) TIMESTAMP (long) EVENT_TYPE (string、ならびにユースケースに応じて、Purchase および View イベントタイプ)	EVENT_VALUE (float、null) IMPRESSION (string、null) RECOMMENDATION_ID (string、null) EVENT_ATTRIBUTION_SOURCE (string、null)
ユーザー (デフォルトのスキーマ)	USER_ID (string) 1 つのメタデータフィールド (カテゴリ string または数値)	
アイテム (デフォルトのスキーマ)	ITEM_ID (string) 料金 (float) CATEGORY_L1 (カテゴリ別 string)	CATEGORY_L2 (カテゴリ別 string、null) CATEGORY_L3 (カテゴリ別 string、null) PRODUCT_DESCRIPTION (テキスト string、null) CREATION_TIMESTAMP (long) AGE_GROUP (カテゴリ別 string、null) ADULT (カテゴリ別 string、null)

データセットタイプ	必須フィールド	予約済みキーワード
		GENDER (カテゴリ別 string、null)

アイテムインタラクションデータセットの要件 (ECOMMERCE ドメイン)

アイテムインタラクションデータセットは、ユーザーとアイテム間のインタラクションからの履歴データとリアルタイムデータを ECOMMERCE カタログに保存します。インタラクションデータセットに保存できるデータの種類の詳細については、「アイテムインタラクションデータセット」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

少なくともアイテムインタラクションデータセットを作成する必要があり、スキーマには以下のフィールドが必要です。

- USER_ID (string)
- ITEM_ID (string)
- TIMESTAMP (long)
- EVENT_TYPE (string、ならびにユースケースに応じて、Purchase および View イベントタイプ)

スキーマには、次の予約済みキーワードを含めることもできます。

- EVENT_VALUE (float、null)
- IMPRESSION (string、null)
- RECOMMENDATION_ID (string、null)

インポートするデータは、スキーマと一致している必要があります。ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。ECOMMERCE ドメインのアイテムインタラクションデータセットのデフォルトスキーマの例については、「デフォルトのインタラクションスキーマ (ECOMMERCE ドメイン)」を参照してください。

イベントに関する値のデータがある場合は、オプションで予約済みキーワード EVENT_VALUE を追加します。明示的および暗黙的なインプレッションデータを含める場合は、オプションで予約済みキーワード IMPRESSION を追加します。インプレッションデータの記録の詳細については、「インプレッションデータ」を参照してください。

アイテムインタラクションデータセットに追加できるオプションのメタデータフィールドの最大合計数は、データ内の個別のイベントタイプの合計数と合わせて 10 です。このカウントに含まれるメタデータフィールドは、スキーマに追加するカスタムメタデータフィールドとともに、EVENT_TYPE、EVENT_VALUE フィールドです。IMPRESSION などの予約済みフィールドを除くメタデータフィールドの最大数は 5 です。カテゴリ値には、最大 1000 文字を使用できます。1000 を超えるカテゴリ値とのインタラクションがある場合、データセットのインポートジョブは失敗します。

ECOMMERCE ドメインのアイテムインタラクションデータセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

デフォルトのインタラクションスキーマ (ECOMMERCE ドメイン)

アイテムインタラクションデータセットのデフォルトの ECOMMERCE ドメインスキーマを次に示します。

```
{

  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
```

```
            "type": "long"
        }
    ],
    "version": "1.0"
}
```

Users データセットの要件 (ECOMMERCE ドメイン)

Users データセットには、ユーザーに関するメタデータが保存されます。これには、各ユーザーについての年齢、性別、ロイヤルティメンバーシップなどの情報が含まれる場合があります。Amazon Personalize にインポートできるユーザーデータの種類の詳細については、「ユーザーデータセット」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

Users データセットは、すべての Eコマースのユースケースで必須ではありません。ユーザーデータがある場合は、最も関連性の高いレコメンデーションを取得するために当該データを作成することをお勧めします。Users データセットを作成する場合、スキーマには次のフィールドが含まれている必要があります。

- USER_ID
- 1 つのメタデータフィールド (カテゴリ `string` または数値)

インポートするデータは、スキーマと一致している必要があります。ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。ECOMMERCE ドメインの Users データセットのデフォルトスキーマの例については、「デフォルトのユーザースキーマ (ECOMMERCE ドメイン)」を参照してください。

Users データセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

カテゴリ別データを使用する

カテゴリデータを使用するには、タイプ `string` のフィールドを追加し、スキーマでフィールドのカテゴリ属性を `true` に設定します。その後、カテゴリデータをバルク CSV ファイルと個別アイテムインポートに含めます。複数のカテゴリを持つユーザーについては、バーティカルバー「|」を使用して各値を区切ります。例えば、SUBSCRIPTION_MODEL フィールドについて、ユーザーのデータは student|monthly|discount である場合があります。

カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つユーザーがいる場合、データセットのインポートジョブは失敗します。

デフォルトのユーザースキーマ (ECOMMERCE ドメイン)

必須のメタデータフィールドとして CATEGORY フィールドを持つ Users データセットのデフォルトの ECOMMERCE ドメインスキーマを次に示します。

```
{

  "type": "record",
  "name": "Users",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "MEMBERSHIP_STATUS",
          "type": "string",
          "categorical": true
      }
  ],
  "version": "1.0"
}
```

Items データセットの要件 (ECOMMERCE ドメイン)

Items データセットには、ECOMMERCE アイテムに関するメタデータが保存されます。これには、各アイテムについての料金、カテゴリ、製品の説明などの情報が含まれる場合があります。Amazon Personalize にインポートできるアイテムデータの種類の詳細については、「製品データセット」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件は、ドメインに関係なくすべてのスキーマに適用されます。

製品データセットは、すべての Eコマースのユースケースで必須ではありません。アイテムデータがある場合は、最も関連性の高いレコメンデーションを取得するために当該データを作成することをお勧めします。Items データセットを作成する場合、スキーマには次のフィールドが含まれている必要があります。

- ITEM_ID

- 料金 (float)
- CATEGORY_L1 (カテゴリ別 string)

スキーマには、次の予約済みキーワードを含めることもできます。カテゴリフィールドでは、ユースケースに基づいて独自の値の範囲を定義できます。

- CATEGORY_L2 (カテゴリ別 string、null)
- CATEGORY_L3 (カテゴリ別 string、null)
- PRODUCT_DESCRIPTION (テキスト string、null)
- CREATION_TIMESTAMP (float)
- AGE_GROUP (カテゴリ別 string、null): 商品が対象とする年齢グループ。値は、新生児、幼児、子供、成人の場合があります。
- 成人 (カテゴリ別 null): string 品目がアルコールなどの成人のみを対象としているかどうか。値は「はい」または「いいえ」の場合があります。
- GENDER (カテゴリ別 string、null): 商品が対象とする性別。値は、男性、女性、男女兼用です。

最良のレコメンデーションを取得するには、これらのフィールドをデータと同じ数だけスキーマに保持することをお勧めします。インポートするデータは、スキーマと一致している必要があります。インポートするデータは、スキーマと一致している必要があります。メタデータ列の最大数は 100 です。ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

複数のサブカテゴリがあるアイテムには、予約済みキーワード CATEGORY_L2 および CATEGORY_L3 を使用します。詳細については、「カテゴリ別データを使用する」を参照してください。テキストおよびカテゴリのメタデータについては、「非構造化テキストメタデータ」を参照してください。ECOMMERCE ドメインの Items データセットのデフォルトスキーマの例については、「デフォルトのアイテムスキーマ (ECOMMERCE ドメイン)」を参照してください。

カテゴリ別データを使用する

カテゴリデータを使用するには、タイプ string のフィールドを追加し、スキーマでフィールドのカテゴリ属性を true に設定します。その後、カテゴリデータをバルク CSV ファイルと個別アイテムインポートに含めます。ユースケースに基づいて独自の値の範囲を定義できます。カテゴリ値に

は、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つアイテムがある場合、データセットのインポートジョブは失敗します。

複数のカテゴリを持つアイテムについては、バーティカルバー「|」を使用して各値を区切ります。例えば、CATEGORY_L1 フィールドについては、アイテムのデータは `Electronics|Productivity|Mouse` である場合があります。複数のレベルのカテゴリデータがあり、一部のアイテムで階層内の各レベル用に複数のカテゴリがあるアイテムがある場合は、各レベル用にフィールドを追加し、フィールド名の後にレベルインジケータ CATEGORY_L1、CATEGORY_L2、CATEGORY_L3 を追加します。これにより、アイテムが複数のマルチレベルカテゴリに属している場合でも、サブカテゴリに基づいてレコメンデーションをフィルタリングできます。例えば、あるアイテムにはカテゴリレベルごとに次のデータが含まれている場合があります。

- CATEGORY_L1: 電子機器|生産性
- CATEGORY_L2: 生産性|コンピューター
- CATEGORY_L3: マウス

この例では、アイテムは エレクトロニクス > 生産性 > マウスの階層 と 生産性 > コンピュータ > マウスの階層 にあります。L3 まで使用することをお勧めしますが、必要であればそれ以上のレベルを使用することもできます。フィルターの作成と使用については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

デフォルトのアイテムスキーマ (ECOMMERCE ドメイン)

必須フィールドのみを含む ECOMMERCE ドメインの Items データセットのデフォルトスキーマを次に示します。

```
{
  "type": "record",
  "name": "Items",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "PRICE",
      "type": "float"
```

```
    },
    {
      "name": "CATEGORY_L1",
      "type": [
        "string"
      ],
      "categorical": true
    }
  ],
  "version": "1.0"
}
```

カスタムデータセットとスキーマ

カスタムデータセットグループを作成する場合、独自のスキーマを最初から作成します。カスタムデータセットグループのデータセットとスキーマでは、必須フィールドは少ないですが、高い柔軟性があります。以下のトピックでは、カスタムデータセットグループのデータセットのスキーマとデータ要件について説明します。各データセットのセクションでは、データセットタイプに必要なデータが一覧表示され、スキーマの JSON の例が提供されます。

Amazon Personalize が使用できるデータの種類については、「データセット」を参照してください。フォーマット要件や使用可能なフィールドデータ型など、Amazon Personalize スキーマの一般的な要件については、「 スキーマ」を参照してください。これらの要件はすべての Amazon Personalize スキーマに適用されます。

トピック

- カスタムデータセットとスキーマの要件
- アイテムインタラクションデータセットスキーマの要件 (カスタム)
- Users データセットスキーマの要件 (カスタム)
- Items データセットスキーマの要件 (カスタム)
- Actions データセットスキーマの要件 (カスタム)
- Action インタラクションデータセットのスキーマ要件 (カスタム)

カスタムデータセットとスキーマの要件

カスタムデータセットグループのデータセットを作成する場合、各データセットタイプには、次の必須フィールドと、必須データタイプの予約済みキーワードがあります。

データセットタイプ	必須フィールド	予約済みキーワード
アイテムインタラクション (スキーマの例)	USER_ID (string) ITEM_ID (string) TIMESTAMP (long)	EVENT_TYPE (string) EVENT_VALUE (float、null) IMPRESSION (string、null) RECOMMENDATION_ID (string、null) EVENT_ATTRIBUTION_SOURCE (string、null)
ユーザー (スキーマの例)	USER_ID (string) 1 つのメタデータフィールド (カテゴリ別 string または数値)	
アイテム (スキーマの例)	ITEM_ID (string) 1 つのメタデータフィールド (カテゴリまたはテキスト string フィールドまたは数値フィールド)	CREATION_TIMESTAMP (long)
Actions (スキーマの例)	ACTION_ID (string) 1 つのメタデータフィールド (カテゴリ別 string または数値)	CREATION_TIMESTAMP (long) VALUE (long、null) TYPE (string、null) EXPIRATION_TIMESTAMP (long、null)

データセットタイプ	必須フィールド	予約済みキーワード
		REPEAT_FREQUENCY (long、null)
Action interactions (スキーマの例)	USER_ID (string) ACTION_ID (string) EVENT_TYPE (string) TIMESTAMP (long)	IMPRESSION (string、null) RECOMMENDATION_ID (string、null)

メタデータフィールド

メタデータには、必須ではない、または予約済みのキーワードを使用しない文字列または非文字列のフィールドが含まれます。メタデータスキーマには、次の制限があります。

- ユーザー、アイテム、アクションのスキーマには、少なくとも 1 つのメタデータフィールドが必要です。
- Users スキーマには最大 25 のメタデータフィールドを追加でき、Items スキーマには最大 100 のメタデータフィールドを追加でき、Actions スキーマには最大 10 のメタデータフィールドを追加できます。
- タイプ `string` の独自のメタデータフィールドを追加する場合は、`categorical` 属性または `textual` 属性を含める必要があります (アイテムスキーマのみがテキスト属性のフィールドをサポートします)。それ以外の場合、Amazon Personalize は、モデルのトレーニング時にフィールドを使用しません。

予約済みキーワード

予約済みのキーワードはオプションの非メタデータフィールドです。これらのフィールドは、使用時に必須のデータ型としてフィールドを定義する必要があり、キーワードをデータに含める値として使用することができないため、予約済みとみなされます。予約済みカテゴリ文字列フィールドは `categorical` が `true` に設定されている必要がありますが、予約済み文字列フィールドはカテゴリに設定できません。予約済みキーワードは次のとおりです。

- EVENT_TYPE: クリックとダウンロードの両方など、1 つ以上のイベントタイプを持つアイテムインタラクションデータセットについては、EVENT_TYPE フィールドを使用します。EVENT_TYPE フィールドを string として定義する必要があり、カテゴリとして設定することはできません。
- EVENT_VALUE: ユーザーが視聴した動画の割合など、イベントの値データを含むアイテムインタラクションデータセットについては、タイプ float とオプションで null を持つ EVENT_VALUE フィールドを使用します。
- CREATION_TIMESTAMP: 各アイテムの作成日のタイムスタンプを持つ Items または Actions データセットについては、タイプ long の CREATION_TIMESTAMP フィールドを使用します。Amazon Personalize は、CREATION_TIMESTAMP データを使用してアイテムが存在するようになってからの期間を計算し、それに応じてレコメンデーションを調整します。作成のタイムスタンプデータを参照してください。
- IMPRESSION: 明示的なインプレッションデータを持つアイテムインタラクションデータセットについては、タイプ String およびオプションでタイプ null の IMPRESSION フィールドを使用します。インプレッションは、ユーザーが特定のアイテムを操作した (例えば、クリックや視聴した) ときに表示されたアイテムのリストです。詳細については、「インプレッションデータ」を参照してください。
- RECOMMENDATION_ID: 以前のレコメンデーションを暗黙的なインプレッションデータとして使用するアイテムインタラクションデータセットについては、タイプ String およびオプションでタイプ null の RECOMMENDATION_ID フィールドを使用します。

  レコメンデーションを生成するときに暗黙的なインプレッションを使用するために、Amazon Personalize の RECOMMENDATION_ID フィールドを追加する必要はありません。それなしで PutEvents 操作で recommendationId を渡すことができます。詳細については、「インプレッションデータ」を参照してください。
- VALUE: Actions データセットで、アクションの一部または全部に値がある場合には、スキーマに VALUE フィールドを追加します。そのタイプについては、long とオプションでタイプ null を使用します。アクションとそれらの値の詳細については、「[Value data] (値のデータ)」を参照してください。
- ACTION_EXPIRATION_TIMESTAMP: Actions データセットで、アクションの一部または全部に有効期限のタイムスタンプがある場合は、スキーマに ACTION_EXPIRATION_TIMESTAMP フィールドを追加します。そのタイプについては、long とオプションでタイプ null を使用します。有効期限タイムスタンプの詳細については、「アクション有効期限のタイムスタンプデータ」を参照してください。
- REPEAT_FREQUENCY: Actions データセットで、アクションの一部または全部に頻度データがある場合には、スキーマに REPEAT_FREQUENCY フィールドを追加します。そのタイプについて

は、long とオプションでタイプ null を使用します。繰り返し頻度データの詳細については、「繰り返し頻度データ」を参照してください。

アイテムインタラクションデータセットスキーマの要件 (カスタム)

アイテムインタラクションデータセットは、ユーザーとアイテム間のインタラクションからの履歴データとリアルタイムデータをカタログに保存します。Amazon Personalize が使用できるインタラクションデータの種類については、「アイテムインタラクションデータセット」を参照してください。

各インタラクションに提供するデータは、スキーマと一致する必要があります。スキーマによっては、インタラクションメタデータに空/null 値を含めることができます。少なくとも、各インタラクションについて以下を入力する必要があります。

- ユーザー ID
- アイテム ID
- タイムスタンプ (Unix エポック時間形式)

ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

アイテムインタラクションデータセットに追加できるオプションのメタデータフィールドの最大合計数は、データ内の個別のイベントタイプの合計数と合わせて 10 です。このカウントに含まれるメタデータフィールドは、スキーマに追加するカスタムメタデータフィールドとともに、EVENT_TYPE、EVENT_VALUE フィールドです。IMPRESSION などの予約済みフィールドを除くメタデータフィールドの最大数は 5 です。カテゴリ値には、最大 1000 文字を使用できます。1000 を超えるカテゴリ値とのインタラクションがある場合、データセットのインポートジョブは失敗します。

アイテムインタラクションデータセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

インタラクションスキーマの例 (カスタム)

次の例は、アイテムインタラクションデータセットのスキーマを示しています。USER_ID、ITEM_ID、TIMESTAMP の各フィールドは必須で

す。EVENT_TYPE、EVENT_VALUE、IMPRESSION フィールドは Amazon Personalize によって認識されるオプションの予約済みキーワードです。EVENT_TYPE は文字列型でなければならず、カテゴリ型にすることはできません。LOCATION および DEVICE はオプションのコンテキストメタデータフィールドです。スキーマ要件については、「カスタムデータセットとスキーマの要件」を参照してください。

```
{

  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ITEM_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
          "type": "string"
      },
      {
          "name": "EVENT_VALUE",
          "type": [
             "float",
             "null"
          ]
      },
      {
          "name": "LOCATION",
          "type": "string",
          "categorical": true
      },
      {
          "name": "DEVICE",
          "type": [
              "string",
              "null"
          ],
          "categorical": true
```

```
        },
        {
            "name": "TIMESTAMP",
            "type": "long"
        },
        {
            "name": "IMPRESSION",
            "type": "string"
        }
    ],
    "version": "1.0"
}
```

このスキーマでは、CSV ファイルの履歴データの最初の数行は次のようになる場合があります。EVENT_VALUE の一部の値が null であることに注意してください。

```
USER_ID,ITEM_ID,EVENT_TYPE,EVENT_VALUE,LOCATION,DEVICE,TIMESTAMP,IMPRESSION
35,73,click,,Ohio,Tablet,1586731606,73|70|17|95|96|92|55|45|16|97|56|54|33|94|36|10|5|
43|19|13|51|90|65|59|38
54,35,watch,0.75,Indiana,Cellphone,1586735164,35|82|78|57|20|63|1|90|76|75|49|71|26|24|
25|6|37|85|40|98|32|13|11|54|48
9,33,click,,Oregon,Cellphone,1586735158,68|33|62|6|15|57|45|24|78|89|90|40|26|91|66|31|
47|17|99|29|27|41|77|75|14
23,10,watch,0.25,California,Tablet,1586735697,92|89|36|10|39|77|4|27|79|18|83|16|28|68|
78|40|50|3|99|7|87|49|12|57|53
27,11,watch,0.55,Indiana,Tablet,1586735763,11|7|39|95|71|1|6|40|41|28|99|53|68|76|0|65|
69|36|22|42|34|67|24|20|66
...
...
```

Users データセットスキーマの要件 (カスタム)

Users データセットには、ユーザーに関するメタデータが保存されます。これには、各アイテムの年齢、性別、ロイヤルティメンバーシップなどの情報が含まれる場合があります。Amazon Personalize がインポートできるデータの種類については、「ユーザーデータセット」を参照してください。

各ユーザーに提供するデータは、スキーマと一致する必要があります。少なくとも、各ユーザーについてユーザー ID を指定する必要があります (最大長 256 文字)。スキーマによっては、ユーザーメタデータに空/null 値を含めることができます。ユーザースキーマには少なくとも 1 つのメタデータフィールドが必要ですが、null タイプを追加すると、ユーザーのこの値は null になる可能性があり

ます。ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

カテゴリデータを使用するには、タイプ string のフィールドを追加し、スキーマでフィールドのカテゴリ属性を true に設定します。その後、カテゴリデータをバルク CSV ファイルと個別アイテムインポートに含めます。複数のカテゴリを持つユーザーについては、バーティカルバー「|」を使用して各値を区切ります。例えば、SUBSCRIPTION_MODEL フィールドについて、ユーザーのデータは student|monthly|discount である場合があります。

カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つユーザーがいる場合、データセットのインポートジョブは失敗します。

Users データセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

ユーザースキーマの例 (カスタム)

次の例は、ユーザースキーマを構築する方法を示しています。USER_ID フィールドは必須であり、AGE および GENDER フィールドはメタデータです。少なくとも 1 つのメタデータフィールドが必要であり、最大 25 のメタデータフィールドを追加できます。スキーマ要件については、「カスタムデータセットとスキーマの要件」を参照してください。

```
{
  "type": "record",
  "name": "Users",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "AGE",
          "type": "int"
      },
      {
          "name": "GENDER",
          "type": "string",
          "categorical": true
      }
```

```
  ],
  "version": "1.0"
}
```

このスキーマでは、CSV ファイルの履歴データの最初の数行は次のようになる場合があります。

```
USER_ID,AGE,GENDER
5,34,Male
6,56,Female
8,65,Male
...
...
```

Items データセットスキーマの要件 (カスタム)

Items データセットには、カタログのアイテムに関するメタデータが保存されます。これには、各アイテムについての料金、ジャンル、利用可否などの情報が含まれる場合があります。Amazon Personalize がインポートできるデータの種類については、「製品データセット」を参照してください。

各アイテム用に入力するデータは、Items データセットのスキーマと一致している必要があります。少なくとも、各アイテムについてアイテム ID を指定する必要があります (最大長 256 文字)。スキーマによっては、アイテムメタデータに空/null 値を含めることができます。スキーマには少なくとも 1 つのメタデータフィールドが必要ですが、`null`タイプを追加すると、項目に対してこの値が null になる可能性があります。ユースケースとデータに応じて、さらにフィールドを追加することができます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

カテゴリデータを使用するには、タイプ `string` のフィールドを追加し、スキーマでフィールドのカテゴリ属性を `true` に設定します。その後、カテゴリデータをバルク CSV ファイルと個別アイテムインポートに含めます。カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つアイテムがある場合、データセットのインポートジョブは失敗します。

複数のカテゴリを持つアイテムについては、バーティカルバー「|」を使用して各値を区切ります。例えば、GENRES フィールドについて、アイテムのデータは `Action|Crime|Biopic` である場合があります。複数のレベルのカテゴリデータがあり、一部のアイテムで階層内の各レベルに複数のカテゴリがあるアイテムがある場合は、各レベルにフィールドを追加し、フィールド名の後にレベルインジケータ GENRE_L2、GENRE_L2、GENRE_L2、GENRE_L3 を追加します。これにより、アイテムが複数のマルチレベルカテゴリに属している場合でも、サブカテゴリに基づいてレコメンデー

ションをフィルタリングできます (フィルタの作成と使用については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください)。例えば、動画にはカテゴリレベルごとに次のデータが含まれている場合があります。

- GENRES: アクション|アドベンチャー
- GENRE_L2: 犯罪|西部劇
- GENRE_L3: バイオピック

この例では、動画はアクション > 犯罪 > バイオピックの階層、およびアドベンチャー > ウエスタン > バイオピックの階層です。L3 まで使用することをお勧めしますが、必要に応じてもっと多くのレベルを使用できます。

モデルトレーニング中に、Amazon Personalize は最大 750,000 アイテムを考慮します。750,000 を超えるアイテムをインポートする場合、Amazon Personalize は、新しいアイテム (インタラクションなしで最近追加したアイテム) と最近のインタラクションデータを含む既存のアイテムを含めることに重点を置いて、トレーニングに含めるアイテムを決定します。

Items データセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

Items データセットのスキーマの例 (カスタム)

次の例は、アイテムスキーマを構築する方法を示します。ITEM_ID フィールドは必須です。GENRE フィールドはカテゴリメタデータであり、DESCRIPTION フィールドはテキストメタデータです。少なくとも 1 つのメタデータフィールドが必要です。最大 100 個のメタデータフィールドを追加できます。CREATION_TIMESTAMP フィールドは予約済みのキーワードです。スキーマ要件については、「カスタムデータセットとスキーマの要件」を参照してください。

```
{
  "type": "record",
  "name": "Items",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "GENRES",
```

```
      "type": [
        "null",
        "string"
      ],
      "categorical": true
    },
    {
      "name": "CREATION_TIMESTAMP",
      "type": "long"
    },
    {
      "name": "DESCRIPTION",
      "type": [
        "null",
        "string"
      ],
      "textual": true
    }
  ],
  "version": "1.0"
}
```

このスキーマでは、CSV ファイルの履歴データの最初の数行は次のようになる場合があります。

```
ITEM_ID,GENRES,CREATION_TIMESTAMP,DESCRIPTION
1,Adventure|Animation|Children|Comedy|Fantasy,1570003267,"This is an animated movie
 that features action, comedy, and fantasy. Audience is children. This movie was
 released in 2004."
2,Adventure|Children|Fantasy,1571730101,"This is an adventure movie with elements of
 fantasy. Audience is children. This movie was release in 2010."
3,Comedy|Romance,1560515629,"This is a romantic comedy. The movie was released in 1999.
 Audience is young women."
4,Comedy|Drama|Romance,1581670067,"This movie includes elements of both comedy and
 drama as well as romance. This movie was released in 2020."
...
...
```

Actions データセットスキーマの要件 (カスタム)

アクションは、顧客に推奨できるエンゲージメントアクティビティです。アクションには、モバイルアプリのインストール、会員プロフィールの記入、ロイヤルティプログラムへの加入、プロモーションメールへの登録などがあります。Actions データセットには、アクションに関するデータが保存さ

れます。Amazon Personalize にインポートできるアクションデータの種類については、「Actions データセット」を参照してください。

各アクションについて入力するデータは、Actions データセットのスキーマと一致している必要があります。スキーマによっては、アクションメタデータに空/null 値を含めることができます。

少なくとも、項目ごとにアクション ID を指定する必要があります (最大長は 256 文字) 。スキーマには少なくとも 1 つのメタデータフィールドが必要ですが、nullタイプを追加すると、アクションに対してこの値が null になる可能性があります。ユースケースとデータに応じて、さらにフィールドを追加できます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

カテゴリフィールドを追加するには、 型のフィールドを追加stringし、trueスキーマでフィールドのカテゴリ属性を に設定します。その後、カテゴリ別データをバルク CSV ファイルと個別のアクションインポートに含めます。カテゴリ値には、最大 1,000 文字を使用できます。1,000 文字を超えるカテゴリ値を持つアクションがある場合、データセットのインポートジョブは失敗します。

Actions データセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

Actions データセットのスキーマの例 (カスタム)

次の例は、Actions スキーマを構築する方法を示しています。ACTION_ID フィールドは必須です。MEMBERSHIP_LEVEL フィールドはカテゴリ別文字列フィールドです。VALUE、CREATION_TIMESTAMP、および REPEAT_FREQUENCY フィールドは、必要なタイプの予約キーワードです。最大 10 列を追加できます。スキーマ要件については、「カスタムデータセットとスキーマの要件」を参照してください。

```
{
  "type": "record",
  "name": "Actions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ACTION_ID",
      "type": "string"
    },
    {
      "name": "VALUE",
      "type": [
        "null",
```

```
        "long"
      ]
    },

    {
      "name": "MEMBERSHIP_LEVEL",
      "type": [
        "null",
        "string"
      ],
      "categorical": true
    },

    {
      "name": "CREATION_TIMESTAMP",
      "type": "long"
    },
    {
      "name": "REPEAT_FREQUENCY",
      "type": [
        "long",
        "null"
      ]
    }
  ],
  "version": "1.0"
}
```

このスキーマでは、CSV ファイルの履歴データの最初の数行は次のようになる場合があります。

```
ACTION_ID,VALUE,MEMBERSHIP_LEVEL,CREATION_TIMESTAMP,REPEAT_FREQUENCY
1,10,Deluxe|Premium,1510003267,7
2,5,Basic,1580003267,7
3,5,Preview,1590003267,3
4,10,Deluxe|Platinum,1560003267,4
...
...
```

Action インタラクションデータセットのスキーマ要件 (カスタム)

Action インタラクションデータセットは、Actions データセット内のユーザーとアクション間のインタラクションからの履歴データとリアルタイムデータを保存します。Amazon Personalize が使用で

きるデータの種類についての詳細は、「Action インタラクションデータセット」を参照してください。

各インタラクションに提供するデータは、スキーマと一致する必要があります。スキーマによっては、インタラクションメタデータに空/null 値を含めることができます。少なくとも、スキーマには以下のものを含める必要があります。

- USER_ID
- ACTION_ID
- TIMESTAMP
- EVENT_TYPE

ユースケースとデータに応じて、さらにフィールドを追加できます。フィールドが必須または予約済みとして記載されておらず、データ型が スキーマのデータ型 に記載されている限り、フィールド名とデータ型は自由に設定できます。

Action インタラクションデータセットの最小要件と最大データ制限の詳細については、「Service Quotas」を参照してください。

Action インタラクションデータセットのスキーマ例 (カスタム)

次の例は、必須フィールドのみを含む Action インタラクションデータセットのスキーマを示しています。一般的なスキーマの形式要件については、「スキーマ形式の要件」を参照してください。

```
{

  "type": "record",
  "name": "ActionInteractions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ACTION_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
```

```
            "type": "string"
        },
        {
            "name": "TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}
```

このスキーマでは、CSV ファイルの履歴データの最初の数行は次のようになる場合があります。IMPRESSION の一部の値が null であることに注意してください。

```
USER_ID,ACTION_ID,EVENT_TYPE,TIMESTAMP
35,73,Viewed,1586731606
54,35,Not taken,1586731609
9,33,Viewed,1586735158
23,10,Taken,1586735697
27,11,Taken,1586735763
...
...
```

SDK for Python (Boto3) を使用したスキーマの作成

1. 使用する Avro 形式のスキーマを定義します。
2. スキーマは、デフォルトの Python フォルダにある JSON ファイルに保存されます。
3. 次のコードを使用してスキーマを作成します。

```
import boto3

personalize = boto3.client('personalize')

with open('schema.json') as f:
    createSchemaResponse = personalize.create_schema(
        name = 'YourSchema'
        schema = f.read()
    )

schema_arn = createSchemaResponse['schemaArn']

print('Schema ARN:' + schema_arn )
```

4. Amazon Personalize は、新しいスキーマの ARN を返します。後で使用するために、これを保存します。

Amazon Personalize には、スキーマを管理するための操作が用意されています。例えば、ListSchemas API を使用して、使用可能なスキーマのリストを取得できます。

スキーマを作成した後は、そのスキーマと一致するデータセットで使用します。詳細については、「データ形式ガイドライン」を参照してください。

データ形式ガイドライン

Amazon Personalize データセットにデータをインポートする場合、レコードを一括でインポートするか、個別にインポートするか、個別にインポートするか、レコードを個別にインポートするか、を選択できます。

- 一括インポートでは、大量の履歴レコードを一度にインポートする必要があります。 SageMaker Data Wrangler と複数のデータソースを使用して、バルクデータを準備してインポートできます。または、自分でバルクデータを準備し、Amazon S3 の CSV ファイルから Amazon Personalize に直接インポートすることもできます。
- 個別インポートでは、Amazon Personalize コンソールと API オペレーションを使用してレコードを個別にインポートします。または、ライブイベントのインタラクションデータをリアルタイムでストリーミングすることもできます。個別のデータのインポートについての詳細は、「個々のレコードのインポート」を参照してください。

バルクデータをインポートする前に、データが正しい形式になっていることを確認してください。以下のセクションは、バルクデータのフォーマットに役立ちます。データのフォーマット方法がわからない場合は、Amazon SageMaker Data Wrangler (Data Wrangler) を使用してデータを準備することができます。詳細については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。

トピック

- バルクデータフォーマットのガイドラインと要件
- インタラクションデータの例
- 明示的なインプレッションのフォーマット
- カテゴリ別データのフォーマット

バルクデータフォーマットのガイドラインと要件

次のガイドラインと要件は、バルクデータが正しくフォーマットされていることを確認するのに役立ちます。

- 入力データはコンマ区切り値 (CSV) としてフォーマットである必要があります。
- CSV ファイルの最初の行には、列ヘッダーが含まれている必要があります。ヘッダーを引用符 (") で囲まないでください。
- データセットタイプに必要なフィールドがあることを確認し、その名前が Amazon Personalize の要件と一致していることを確認してください。例えば、アイテムデータには、各アイテムの ID を持つ ITEM_IDENTIFICATION_NUMBER と呼ばれる列を含むことがあります。この列を ITEM_ID フィールドとして使用するには、列の名前を ITEM_ID に変更します。Data Wrangler を使用してデータをフォーマットする場合は、Amazon Personalize Data Wrangler のマップ列変換を使用して、列に正しい名前が付けられていることを確認できます。

  必要なフィールドについての詳細は、「スキーマ」を参照してください。Data Wrangler を使用してデータを準備する方法については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。
- CSV ファイルの列ヘッダー名は、スキーマに対応している必要があります。
- CSV ファイルの各レコードは 1 行にする必要があります。
- 各列のデータ型はスキーマに対応している必要があります。Data Wrangler を使用してデータをフォーマットする場合、Data Wrangler 変換の Parse Value as Type を使用してデータ型を変換できます。
- TIMESTAMP および CREATION_TIMESTAMP データは UNIX エポック時間形式である必要があります。詳細については、「タイムスタンプのデータ」を参照してください。
- 項目 ID、ユーザー ID、およびアクション ID データに文字"や特殊文字を含めないでください。
- データに ASCII でエンコードされていない文字が含まれている場合は、CSV ファイルを UTF-8 形式でエンコードする必要があります。
- テキストデータは必ず、「非構造化テキストメタデータ」で説明されているとおりにフォーマットしてください。
- インプレッションデータとカテゴリ別データは、「明示的なインプレッションのフォーマット」と「カテゴリ別データのフォーマット」で説明されているとおりにフォーマットしてください。

インタラクションデータの例

次のインタラクションデータは、映画のチケットを販売するウェブサイトからの過去のユーザーアクティビティを表しています。このデータを使用して、ユーザーのインタラクションデータに基づいておすすめの映画を提供するモデルをトレーニングできます。

```
USER_ID,ITEM_ID,EVENT_TYPE,EVENT_VALUE,TIMESTAMP
196,242,click,15,881250949
186,302,click,13,891717742
22,377,click,10,878887116
244,51,click,20,880606923
166,346,click,10,886397596
298,474,click,40,884182806
115,265,click,20,881171488
253,465,click,50,891628467
305,451,click,30,886324817
```

関連するインタラクションスキーマは次のとおりです。

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "USER_ID",
      "type": "string"
    },
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    { "name": "EVENT_TYPE",
      "type": "string"
    },
    {
      "name": "EVENT_VALUE",
      "type": "float"
    },
    {
      "name": "TIMESTAMP",
```

```
      "type": "long"
    }
  ],
  "version": "1.0"
}
```

Amazon Personalize には、USER_ID、ITEM_ID、および TIMESTAMP フィールドが必要です。USER_ID は、アプリケーションのユーザーの識別子です。ITEM_ID は映画の識別子です。EVENT_TYPE と EVENT_VALUE は、ユーザーアクティビティの識別子です。サンプルデータでは、click は映画の購入イベントを表し、15 は映画の購入価格です。TIMESTAMP は、映画の購入が実行された Unix エポック時間を表します。

タイムスタンプのデータ

TIMESTAMP (アイテムインタラクションデータセットの場合) または CREATION_TIMESTAMP (Items データセットの場合) データなどのタイムスタンプデータは、Unix エポック時間 (秒) 形式である必要があります。例えば、2020 年 7 月 31 日の日付のエポックタイムスタンプ (秒) は 1596238243 です。日付を Unix エポックタイムスタンプに変換するには、[Epoch converter - Unix timestamp converter] (エポックコンバーター - Unix タイムスタンプコンバーター) を使用します。

明示的なインプレッションのフォーマット

User-Personalization レシピを使用すると、インプレッションデータを記録およびアップロードできます。インプレッションは、ユーザーが特定のアイテムを操作した (例えば、クリックや視聴を行った) ときに表示されたアイテムのリストです。一括データインポートでインプレッションデータをアップロードするには、各アイテム ID を手動で記録します。履歴インタラクションデータの一部として値をバーティカルバー「|」の文字で区切ってください。バーティカルバーは、インプレッションデータの 1,000 文字の上限に含まれます。インプレッションデータの詳細については、「インプレッションデータ」を参照してください。

IMPRESSION 列に明示的なインプレッションを含むアイテムインタラクションデータセットからの短い抜粋を次に示します。

EVENT_TYPE	IMPRESSION	ITEM_ID	TIMESTAMP	USER_ID
クリック	73|70|17|95|96	73	1586731606	USER_1

EVENT_TYPE	IMPRESSION	ITEM_ID	TIMESTAMP	USER_ID
クリック	35|82|78|57|20|63|1|90|76|75|49|71|26|24|25|6	35	1586735164	USER_2
...	...	...	...	...

アプリケーションは、ユーザーに対して、 USER_1 アイテム、73、70、17、95 および 96 を表示し、ユーザーは最終的にアイテム 73 を選択しました。このデータに基づいて新しいソリューションバージョンを作成すると、アイテム 70、17、95、および 96 がユーザー USER_1 に推奨される頻度が低くなります。

カテゴリ別データのフォーマット

カテゴリ別文字列データを使用する場合に 1 つのアイテムに複数のカテゴリを含めるには、縦棒文字「|」を使用して値を区切ります。例えば、2 つのカテゴリがあるアイテムの場合、データ行は次のようになります。

```
ITEM_ID,GENRE
item_123,horror|comedy
```

データをフォーマットしたら、Amazon S3 バケットにアップロードして、Amazon Personalize にインポートできるようにします。詳細については、「Amazon S3 バケットへのアップロード」を参照してください。

ドメインのユースケースとカスタムレシピ

Amazon Personalize では、トレーニングモデルのさまざまなドメインユースケースとカスタムレシピを提供しています。

- ドメインデータセットグループでレコメンダーを作成する際に、ユースケースを指定します。Amazon Personalize は、ユースケースに最適な設定でレコメンダーをサポートするモデルをトレーニングします。
- カスタムデータセットグループまたはドメインデータセットグループでカスタムソリューションを作成する場合、レシピを指定し、トレーニングパラメータを設定します。ソリューションのソリューションバージョンを作成すると、Amazon Personalize はレシピとトレーニング設定に基づいてソリューションバージョンを裏付けるモデルをトレーニングします。

トピック

ユースケースとレシピ機能

Amazon Personalize では、いくつかのユースケースとレシピを使用して、より関連性の高いレコメンデーションを生成し、アイテムの発見とエンゲージメントを向上させます。

トピック

リアルタイムパーソナライゼーション

一部のユースケースやレシピでは、Amazon Personalize はリアルタイムのパーソナライゼーションを使用して、ユーザーの関心の高まりに応じてレコメンデーションを更新および調整します。最新の完全なトレーニングで提示されたアイテムまたはアクションのインタラクションを記録すると、ユー

ザー向けのレコメンデーションが更新されます。これらのインタラクションをイベントトラッカーとPutEventsオペレーション、またはアクションとのインタラクションの場合は PutActionInteractions オペレーションに記録します。

イベントの記録についての詳細は、「イベントの記録」を参照してください。リアルタイムのレコメンデーションに影響する新しいデータ (リアルタイムのパーソナライゼーションを含む) については、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。

リアルタイムパーソナライゼーションをサポートしているユースケースとレシピは次のとおりです。

- おすすめ (eコマース ユースケース)
- 上位のおすすめ (VIDEO_ON_DEMAND ユースケース)
- User-Personalization-v2 レシピ
- User-Personalization レシピ
- Personalized-Ranking-v2 レシピ
- Personalized-Ranking レシピ
- Next-Best-Action レシピ

探査

一部のドメインユースケースやカスタムレシピでは、Amazon Personalize はアイテムを推奨する際に探索を使用します。探索では、新しいアイテムまたはアクション、インタラクションがほとんどないアイテムまたはアクション、以前の行動に基づいてユーザーに関連性が低いアイテムまたはアクションなど、通常はユーザーにレコメンデーションされる可能性が低いアイテムまたはアクションがレコメンデーションに含まれます。これにより、カタログの更新が早い場合や、ニュース記事やプロモーションなどの新しいアイテムが新鮮であるため、ユーザーにとってより関連性が高い場合に、アイテムの発見とエンゲージメントが向上します。

探索の設定

User-Personalization-v2 レシピを使用する場合、Amazon Personalize は、ユーザーおよび探索に含まれるアイテムの探索設定`Exploration`をレコメンデーションレスポンス`Reason`で処理します。新しいアイテムがレコメンデーションに含まれていることを確認するには、プロモーションフィルターを使用して、作成タイムスタンプに基づいて新しいアイテムをプロモーションできます。プロモーションの詳細については、「」を参照してくださいレコメンデーション内のアイテムのプロモーション。

探索を使用する他のすべてのユースケースやレシピの場合、レコメンダーやカスタムキャンペーンを作成するとき、またはバッチ推論ジョブ (カスタムリソース) を作成するときに、次のフィールドを使用して探索を設定できます。

- 関連性の低いアイテムの探索に重点を置く (探索の重み) — 探索する範囲を設定します。0 ~ 1 の小数値を指定します。デフォルトは 0.3 です。値が 1 に近くなるほど、探索が多くなります。探索が増えると、レコメンデーションにはより多くのアイテムが含まれますが、アイテムインタラクションデータや以前の行動に基づく関連性が少なくなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
- 探索アイテムが存在するようになってからの期間のカットオフ - アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は作成タイムスタンプを基に、あるいは作成タイムスタンプデータがない場合はアイテムインタラクションデータを基に、アイテムが存在するようになってからの期間を決定します。Amazon Personalize がアイテムが存在するようになってからの期間を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。

  Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。これは、これらのアイテムがユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったことによります。

探索を使用するユースケースとレシピ

探索を使用する各ユースケースまたはレシピの詳細については、以下を参照してください。

- おすすめ (eコマースのユースケース)
- 上位のおすすめ (VIDEO_ON_DEMAND ユースケース)
- User-Personalization-v2 レシピ
- User-Personalization レシピ
- Next-Best-Action レシピ

自動更新

一部のユースケースやカスタムレシピでは、Amazon Personalize はレコメンダーまたはソリューションバージョンを自動的に更新して、レコメンデーションの新しいアイテムやアクションを検討します。自動更新には費用がかかりません。自動更新を使用したユースケースとレシピのリストについては、「」を参照してください自動更新によるドメインのユースケースとカスタムレシピ。

自動更新は次のように機能します。

- Amazon Personalize がソリューションバージョンまたはレコメンダーを自動的に更新するときは、レコメンデーションの取得方法によって異なります。
  - リアルタイムのレコメンデーションについては、Amazon Personalize は 2 時間ごとにソリューションバージョンまたはレコメンダーを更新します。
  - バッチアイテムのレコメンデーションでは、バッチ推論ジョブを作成し、ソリューションに完全にトレーニングされた最新のソリューションバージョンを指定すると、Amazon Personalize は自動的にソリューションバージョンを更新し、探索中に新しいアイテムを検討します。最新のソリューションバージョンを指定しないと、更新は行われません。
- 更新のたびに、Amazon Personalize は を使用して新しいアイテムをレコメンデーションに含め始めます探査。新しい項目またはアクションを検討する場合、Amazon Personalize は項目のメタデータをすべて考慮します。ただし、このデータは、アイテムのインタラクションを記録して完全に再トレーニングした後にのみ、レコメンデーションに大きな影響を与えます。
- 更新を実行するには、前回の自動更新または再トレーニング以降の新しいアクション、アイテム、またはインタラクションデータを指定する必要があります。
- Amazon Personalize は、750,000 個のアイテムをインポートするまで新しいアイテムを考慮します。これは、トレーニング中に考慮される項目の最大数です。

カスタムリソースに関する追加ガイドラインと要件

カスタムリソースを使用する場合の自動更新のガイドラインと要件は次のとおりです。

- ソリューションバージョンはキャンペーンにデプロイする必要があります。キャンペーンでは、更新されたソリューションバージョンが自動的に使用されます。
- 自動更新は、自動トレーニングとは異なります。自動更新では、まったく新しいソリューションバージョンは作成されません。また、モデルは最新のデータから学習しません。ソリューションを維持するには、自動トレーニングの頻度を少なくとも毎週設定する必要があります。

- ソリューションが自動的に新しいソリューションバージョンを作成するか、手動で作成した後、Amazon Personalize はキャンペーンにデプロイした場合でも、古いソリューションバージョンを自動的に更新しません。
- 2 時間ごとに十分な頻度でない場合は、User-Personalization を使用して、 を `trainingMode` に設定してソリューションバージョンを手動で作成UPDATEし、それらの新しいアイテムをレコメンデーションに含めることができます。Amazon Personalize では、完全にトレーニングされたソリューションバージョンのみが自動的に更新されることを覚えておいてください。手動で更新されたソリューションバージョンは、今後、自動的に更新されません。ソリューションが自動トレーニングを使用している場合、次のソリューションバージョンで自動更新が再開されます。そうでない場合は、トレーニングモードを に設定して新しいソリューションを手動で作成FULLし、キャンペーンにデプロイします。

自動更新によるドメインのユースケースとカスタムレシピ

自動更新を特徴とする各ユースケースまたはレシピについての詳細は、以下を参照してください。

- おすすめ (eコマースのユースケース)
- 上位のおすすめ (VIDEO_ON_DEMAND ユースケース)
- User-Personalization-v2 レシピ
- User-Personalization レシピ
- Next-Best-Action レシピ

ユースケースの選択

ドメインデータセットグループでレコメンダーを作成する際に、ユースケースを指定します。Amazon Personalize は、選択したユースケースに最適な設定で各レコメンダーをサポートするモデルをトレーニングします。各ドメインには、異なるユースケースがあります。例えば、ドメインデータセットグループに VIDEO_ON_DEMAND を指定した場合、使用できるのは VIDEO_ON_DEMAND ユースケースのみです。各ユースケースには、レコメンデーションを取得するための異なる API 要件があります。ユースケースによっては、特定のイベントタイプが必要です。追加のイベントタイプを自由に含めることができます。

すべてのユースケースで、インタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

トピック

VIDEO_ON_DEMAND ユースケース

次のセクションでは、各 VIDEO_ON_DEMAND ユースケースの要件と Amazon リソースネーム (ARN) を示します。すべてのユースケースで、インタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

> **Note**
>
> CreateRecommender API を使用する場合は、レシピ ARN 用にここにリストされている ARN を指定します。

トピック

- こちらもおすすめ
- 最も人気
- Trending now
- 上位のおすすめ

X を視聴したから

指定した動画に基づいて、他のユーザーが視聴した動画のレコメンデーションを取得します。このユースケースでは、Amazon Personalize は、指定した userId と `Watch` イベントに基づいて、ユーザーが視聴した動画を自動的にフィルタリングします。独自のフィルターを適用すると、そのユーザーが視聴した動画がフィルターで除外された後に、そのフィルターが適用されます。

フィルタリングの際に、Amazon Personalize はイベントタイプごとにユーザー 1 人あたり最大 100 件のアイテムインタラクションを考慮します。これはすべての自動フィルターまたはカスタムフィルターに適用されます。この制限の引き上げをリクエストするには、Service Quotas コンソールを使用します。詳細については、Service Quotas ユーザーガイドの「クォータ引き上げのリクエスト」を参照してください。

- レシピ ARN `arn:aws:personalize:::recipe/aws-vod-because-you-watched-x`
- GetRecommendations API の要件:

  `userId`: 必須

  `itemId`: 必須
- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必須イベントタイプ: 最低 1,000 件`Watch`のイベント。

こちらもおすすめ

指定した動画に類似した動画のレコメンデーションを取得します。このユースケースでは、Amazon Personalize は、指定した userId と `Watch` イベントに基づいて、ユーザーが視聴した動画を自動的にフィルタリングします。独自のフィルターを適用すると、そのユーザーが視聴した動画がフィルターで除外された後に、そのフィルターが適用されます。

フィルタリングの際に、Amazon Personalize はイベントタイプごとにユーザー 1 人あたり最大 100 件のアイテムインタラクションを考慮します。これはすべての自動フィルターまたはカスタムフィルターに適用されます。この制限の引き上げをリクエストするには、Service Quotas コンソールを使

用します。詳細については、Service Quotas ユーザーガイドの「クォータ引き上げのリクエスト」を参照してください。

- レシピ ARN `arn:aws:personalize:::recipe/aws-vod-more-like-x`
- GetRecommendations API の要件:

  `userId`: 必須

  `itemId`: 必須
- トレーニング時に使用したデータセット:
  - インタラクション (必須)
  - アイテム (必須)
- 必要なイベント数: 種類を問わず、最低 1,000 件のイベント。
- 推奨イベントタイプ: `Watch` および `Click` イベント。

最も人気

最も多くのユーザーが視聴した動画のレコメンデーションが表示されます。

- レシピ ARN `arn:aws:personalize:::recipe/aws-vod-most-popular`
- GetRecommendations の要件 :

  `userId`: 必須

  `itemId`: 使用されない
- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必須イベントタイプ: 最低 1,000 `Watch` 件のイベント。

Trending now

現在トレンドになっている動画のレコメンデーションを取得します。トレンド動画とは、ユーザーの間で急速に人気が高まっているアイテムです。Amazon Personalize は 2 時間ごとにインタラクションデータを自動的に評価し、トレンドアイテムを特定します。

- レシピ ARN `arn:aws:personalize:::recipe/aws-vod-trending-now`
- GetRecommendations API の要件:

`userId`: ユーザーが操作したアイテム CurrentUser でフィルタリングする場合にのみ必要

`itemId`: 使用されない

- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必要なイベント数: 種類を問わず、最低 1000 件のイベント。

上位のおすすめ

指定したユーザー向けにパーソナライズされたコンテンツのレコメンデーションを取得します。このユースケースでは、Amazon Personalize は、指定した userId と `Watch` イベントに基づいて、ユーザーが視聴した動画を自動的にフィルタリングします。独自のフィルターを適用すると、そのユーザーが視聴した動画がフィルターで除外された後に、そのフィルターが適用されます。

フィルタリングの際に、Amazon Personalize はイベントタイプごとにユーザー 1 人あたり最大 100 件のアイテムインタラクションを考慮します。これはすべての自動フィルターまたはカスタムフィルターに適用されます。この制限の引き上げをリクエストするには、Service Quotas コンソールを使用します。詳細については、Service Quotas ユーザーガイドの「クォータ引き上げのリクエスト」を参照してください。

アイテムを推奨する場合、このユースケースでは real-time-personalizationと 探索 を使用します。また、自動更新機能を使って新しい商品をレコメンデーションの対象として検討します。

- レシピ ARN `arn:aws:personalize:::recipe/aws-vod-top-picks`
- GetRecommendations の要件:

  `userId`: 必須

  `itemId`: 使用されない
- トレーニング時に使用したデータセット:
  - インタラクション (必須)
  - アイテム (オプション)
  - ユーザー (オプション)
- 必要なイベント数:最低 1,000 件のイベント。
- 推奨イベントタイプ: `Click` および `Watch` イベント。
- 探索構成パラメーター: レコメンダーを作成する際に、以下を使用して探索を設定できます。

- 関連性の低いアイテムの探索に重点を置く (探索の重み) — 探索する範囲を設定します。0 ~ 1 の小数値を指定します。デフォルトは 0.3 です。値が 1 に近くなるほど、探索が多くなります。探索が増えると、レコメンデーションにはより多くのアイテムが含まれますが、アイテムインタラクションデータや以前の行動に基づく関連性が少なくなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
- 探索アイテムが存在するようになってからの期間のカットオフ - アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は作成タイムスタンプを基に、あるいは作成タイムスタンプデータがない場合はアイテムインタラクションデータを基に、アイテムが存在するようになってからの期間を決定します。Amazon Personalize がアイテムが存在するようになってからの期間を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。

  Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。これは、これらのアイテムがユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったことによります。

ECOMMERCE のユースケース

次のセクションでは、各 ECOMMERCE ユースケースの要件と Amazon リソースネーム (ARN) を示します。すべてのユースケースで、インタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

> **Note**
>
> CreateRecommender API を使用する場合は、レシピ ARN 用にここにリストされている ARN を指定します。

トピック

閲覧数が最も多い

顧客があるアイテムを閲覧した回数に基づいて、人気のあるアイテムのレコメンデーションを取得します。

- レシピ ARN: `arn:aws:personalize:::recipe/aws-ecomm-popular-items-by-views`
- GetRecommendations の要件:

  `userId`: 必須

  `itemId`: 使用されない

  `inputList`: 該当なし
- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必須イベントタイプ:最低 1,000 `View` イベント。

ベストセラー

顧客があるアイテムを購入した回数に基づいて、人気のあるアイテムのレコメンデーションを取得します。

- レシピ ARN: `arn:aws:personalize:::recipe/aws-ecomm-popular-items-by-purchases`

- GetRecommendations の要件:

  `userId`: 必須

  `itemId`: 使用されない

  `inputList`: 該当なし
- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必須イベントタイプ:最低 1,000 `Purchase` イベント。

よく一緒に購入されています

指定したアイテムに基づいて、顧客が頻繁に一緒に購入するアイテムのレコメンデーションを取得します。

- レシピ ARN: `arn:aws:personalize:::recipe/aws-ecomm-frequently-bought-together`
- GetRecommendations の要件:

  `userId`: でフィルタリングする場合にのみ必須 CurrentUser

  `itemId`: 必須

  `inputList`: 該当なし
- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必須イベントタイプ:最低 1,000 `Purchase` イベント。

X を閲覧したお客様はこちらも閲覧しました

指定したアイテムに基づいて、顧客がさらに閲覧したアイテムのレコメンデーションを取得します。このユースケースでは、Amazon Personalize は、指定した userId と `Purchase` イベントに基づいて、ユーザーが購入したアイテムを自動的にフィルタリングします。独自のフィルターを適用すると、ユーザーがすでに購入したアイテムがフィルターで除外された後にフィルターが適用されます。

フィルタリングの際に、Amazon Personalize はイベントタイプごとにユーザー 1 人あたり最大 100 件のアイテムインタラクションを考慮します。これはすべての自動フィルターまたはカスタムフィルターに適用されます。この制限の引き上げをリクエストするには、Service Quotas コンソールを使

用します。詳細については、Service Quotas ユーザーガイドの「クォータ引き上げのリクエスト」を参照してください。

- レシピ ARN `arn:aws:personalize:::recipe/aws-ecomm-customers-who-viewed-x-also-viewed`
- GetRecommendations の要件:

  `userId`: 必須

  `itemId`: 必須

  `inputList`: 該当なし
- トレーニング時に使用したデータセット: アイテムインタラクションデータセットのみ (必須)
- 必須イベントタイプ:最低 1,000 `View` イベント。
- 推奨イベントタイプ: `Purchase`イベント。

おすすめ

指定したユーザーに基づいて、アイテムに関してパーソナライズされたレコメンデーションを取得します。このユースケースでは、Amazon Personalize は、指定した userId と `Purchase` イベントに基づいて、ユーザーが購入したアイテムを自動的にフィルタリングします。独自のフィルタを適用すると、ユーザーがすでに購入したアイテムがフィルターで除外された後にフィルタが適用されます。

フィルタリングの際に、Amazon Personalize はイベントタイプごとにユーザー 1 人あたり最大 100 件のアイテムインタラクションを考慮します。これはすべての自動フィルターまたはカスタムフィルターに適用されます。この制限の引き上げをリクエストするには、Service Quotas コンソールを使用します。詳細については、Service Quotas ユーザーガイドの「クォータ引き上げのリクエスト」を参照してください。

アイテムを推奨する場合、このユースケースでは real-time-personalizationと 探索 を使用します。また、自動更新機能を使って新しい商品をレコメンデーションの対象として検討します。

- レシピ ARN `arn:aws:personalize:::recipe/aws-ecomm-recommended-for-you`
- GetRecommendations の要件:

  `userId`: 必須

  `itemId`: 使用されない

`inputList`: 該当なし

- トレーニング時に使用したデータセット:
  - インタラクション (必須)
  - アイテム (オプション)
  - ユーザー (オプション)
- 必要なイベント数:最低 1,000 件のイベント。
- 推奨イベントタイプ: `View` および `Purchase` イベント。
- 探索構成パラメーター: レコメンダーを作成する際に、以下を使用して探索を設定できます。
  - 関連性の低いアイテムの探索に重点を置く (探索の重み) — 探索する範囲を設定します。0 ~ 1 の小数値を指定します。デフォルトは 0.3 です。値が 1 に近くなるほど、探索が多くなります。探索が増えると、レコメンデーションにはより多くのアイテムが含まれますが、アイテムインタラクションデータや以前の行動に基づく関連性が少なくなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
  - 探索アイテムが存在するようになってからの期間のカットオフ - アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は作成タイムスタンプを基に、あるいは作成タイムスタンプデータがない場合はアイテムインタラクションデータを基に、アイテムが存在するようになってからの期間を決定します。Amazon Personalize がアイテムが存在するようになってからの期間を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。

    Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。これは、これらのアイテムがユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったことによります。

レシピの選択

カスタムソリューションを作成するときに、レシピを指定し、トレーニングパラメーターを設定します。レシピは、特定のユースケース向けに準備された Amazon Personalize のアルゴリズムです。Amazon Personalize は、一般的なユースケースに基づいて、モデルをトレーニングするためのレシピを提供します。ソリューションのソリューションバージョンを作成すると、Amazon

Personalize はレシピとトレーニング設定に基づいてソリューションバージョンを裏付けるモデルをトレーニングします。

Amazon Personalize のレシピは、トレーニング中に次を使用します。

- データの事前定義済み属性
- 事前定義済み特徴変換
- 事前定義済みアルゴリズム
- アルゴリズムの初期パラメータ設定

モデルを最適化するために、ソリューションの作成時にこれらのパラメータの多くを上書きできます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

トピック

- Amazon Personalize のユースケース別レシピの種類
- Amazon Personalize のレシピ
- 利用可能な Amazon Personalize のレシピの表示
- USER_PERSONALIZATION
- POPULAR_ITEMS
- PERSONALIZED_RANKING
- RELATED_ITEMS
- PERSONALIZED_ACTIONS
- USER_SEGMENTATION

Amazon Personalize のユースケース別レシピの種類

レシピを選択するには、まず以下からユースケースを選択し、対応するレシピタイプを書き留めます。

- ユーザーに推奨するアイテム (USER_PERSONALIZALIZATION レシピ)

  ユーザーにレコメンデーションを提供するには、USER_PERSONALIZATION レシピを使用してモデルをトレーニングします。パーソナライズされたレコメンデーションは、エンゲージメントとコンバージョンの向上に役立ちます。

- ユーザー向けのアイテムのランク付け (PERSONALIZED_RANKING レシピ)

  ユーザー向けに厳選されたリストまたは検索結果の順序をパーソナライズするには、PERSONALIZED_RANKING レシピを使用してモデルをトレーニングします。PERSONALIZED_RANKING レシピは、特定のユーザーの予測される関心レベルに基づいて入力アイテムのコレクションを再ランク付けすることにより、パーソナライズされたリストを作成します。パーソナライズされたリストは、カスタマーエクスペリエンスを向上させ、顧客のロイヤルティとエンゲージメントを高めます。

- トレンド商品や人気商品 (POPULAR_ITEMS レシピ) の推薦

  トレンド商品や人気商品をレコメンドするには、POPULAR_ITEMS レシピを使用してください。POPULAR_ITEMS は、顧客が他のユーザーとのやり取りを高く評価している場合に使用できます。一般的な用途としては、話題のソーシャルメディアコンテンツ、最新ニュース記事、最近のスポーツ動画を勧めることが挙げられます。

- 類似アイテムの推奨 (RELATED_ITEMS レシピ)

  頻繁に一緒に購入される商品や他のユーザーも視聴した映画など、類似アイテムを推奨するには、RELATED_ITEMS レシピを使用する必要があります。類似のアイテムを推奨すると、顧客がアイテムを見つけやすくなり、ユーザーのコンバージョン率を高めることができます。

- 次善のアクションのレコメンド (PERSONALIZED_ACTIONS レシピ)

  ロイヤルティプログラムへの登録やクレジットカードの申請など、次善のアクションをユーザーにリアルタイムでレコメンドするには、PERSONALIZED_ACTIONS レシピを使用する必要があります。次善のアクションをレコメンドすることにより、顧客ロイヤルティを高め、収益を増やし、ユーザーエクスペリエンスを向上させることができます。

- ユーザーセグメントの取得 (USER_SEGMENTATION レシピ)

  特定の属性を持つアイテムを操作する可能性が最も高いユーザーなど、アイテム入力データに基づいてユーザーのセグメントを取得するには、USER_SEGMENTATION レシピを使用する必要があります。ユーザーセグメントを取得すると、アクションを実行する可能性に基づいて、さまざまなユーザーセグメントに対して、さまざまなアイテムのプロモーションを実施する高度なマーケティングキャンペーンを作成するのに役立ちます。

Amazon Personalize のレシピ

Amazon Personalize では、3 種類のレシピを利用できます。次の表に示すように、各タイプは、動作が異なるほかにレコメンデーションを取得する要件が異なります。

レシピタイプ	recipe	API	API 要件
USER_PERSONALIZATION	User-Personalization-v2 User-Personalization HRNN レシピ (レガシー) HRNN-Metadata レシピ (レガシー) HRNN-Coldstart レシピ (レガシー)	GetRecommendations	`userId`: 必須 `itemId`: 使用されない `inputList` : 該当なし
POPULAR_ITEMS	トレンド-ナウ Popularity-Count	GetRecommendations	`userId`: それを必要とするフィルターを適用する場合にのみ必要です。 `itemId`: 使用されない `inputList` : 該当なし
PERSONALIZED_RANKING	Personalized-Ranking-v2 Personalized-Ranking	GetPersonalizedRanking	`userId`: 必須 `itemId`: 該当なし `inputList` : itemId のリスト
RELATED_ITEMS	Similar-Items SIMS	GetRecommendations	`userId`: それを必要とするフィルタを適用する場合

レシピタイプ	recipe	API	API 要件
			にのみ必要です。 itemId: 必須 inputList : 該当なし
PERSONALIZED_ACTIONS	Next-Best-Action	GetActionRecommendations	userId: 必須 actionId: 使用されない itemId: 使用されない inputList : 該当なし
USER_SEGMENTATION	Item-Affinity Item-Attribute-Affinity	CreateBatchSegmentJob	バッチワークフローの要件については、「バッチセグメントジョブの作成」を参照してください。

利用可能な Amazon Personalize のレシピの表示

使用可能なレシピのリストを表示するには:

- Amazon Personalize コンソールで、データセットグループを選択します。ナビゲーションペインから、[Solutions and recipes (ソリューションとレシピ)] を選択し、[Recipes (レシピ)] タブを選択します。
- で AWS SDK for Python (Boto3)、 ListRecipes API を呼び出します。

- で AWS CLI、次のコマンドを使用します。

```
aws personalize list-recipes
```

SDK for Python (Boto3) を使用したレシピに関する情報を取得するには、DescribeRecipe API を呼び出します。を使用してレシピに関する情報を取得するには AWS CLI、次のコマンドを使用します。

```
aws personalize describe-recipe --recipe-arn recipe_arn
```

USER_PERSONALIZATION

USER_PERSONALIZATION レシピは、Interactions、Items、および Users のデータセットに基づいて、ユーザーがインタラクションするアイテムを予測します。ユーザーごとにパーソナライズされたレコメンデーションを提供する場合は、USER_PERSONALIZATION レシピを使用してモデルをトレーニングする必要があります。

USER_PERSONALIZATION レシピは次のとおりです。

トピック

- User-Personalization-v2 レシピ
- User-Personalization レシピ
- レガシーユーザーのパーソナライゼーションレシピ

User-Personalization-v2 レシピ

User-Personalization-v2 (aws-user-personalization-v2) レシピは、ユーザーが好みに基づいて操作するアイテムを推奨します。例えば、User-Personalization-v2 を使用して、ストリーミングアプリケーション用にパーソナライズされた映画のレコメンデーションを生成したり、小売アプリケーション用にパーソナライズされた製品のレコメンデーションを生成したりできます。その他のユースケースには、ニュースサイトに関するリアルタイムのレコメンデーションの生成や、パーソナライズされたマーケティングキャンペーンに関するバッチレコメンデーションの生成などがあります。

User-Personalization-v2 は、アイテムインタラクションとアイテムデータセットから最大 500 万のアイテムでトレーニングできます。また、 よりもレイテンシーが低く、より関連性の高いレコメンデーションを生成しますUser-Personalization。

User-Personalization-v2 はデータに基づいて最も関連性の高いアイテムをユーザーに推奨するため、インタラクションデータを含む既存のアイテムをより頻繁に推奨します。レコメンデーションに新しいアイテムが含まれるようにするには、作成タイムスタンプに基づいて一部のアイテムを含むプロモーションを使用できます。プロモーションの詳細については、「」を参照してくださいレコメンデーション内のアイテムのプロモーション。

このレシピは、トランスフォーマーベースのアーキテクチャを使用して、コンテキストを学習し、データ内の関係とパターンを追跡するモデルをトレーニングします。トランスフォーマーは、入力シーケンスを出力シーケンスに変換または変更するニューラルネットワークアーキテクチャの一種です。Amazon Personalize の場合、入力シーケンスはデータ内のユーザーのアイテムインタラクション履歴です。出力シーケンスは、パーソナライズされたレコメンデーションです。トランスフォーマーの詳細については、 AWS クラウドコンピューティングの概念ハブの「人工知能におけるトランスフォーマーとは」を参照してください。

User-Personalization-v2 は、他のレシピとは異なる料金モデルを使用します。料金の詳細については、「Amazon Personalize の料金」を参照してください。

トピック

- レシピ機能
- 必須およびオプションのデータセット
- プロパティおよびハイパーパラメータ

レシピ機能

User-Personalization-v2 は、アイテムレコメンデーションを生成するときに、次の Amazon Personalize レシピ機能を使用します。

- リアルタイムパーソナライゼーション — リアルタイムパーソナライゼーションでは、Amazon Personalize はユーザーの関心の高まりに応じてアイテムのレコメンデーションを更新および適応します。詳細については、「リアルタイムパーソナライゼーション」を参照してください。
- 探索 — 探索では、レコメンデーションには、インタラクションデータやユーザーとの関連性が少ないアイテムが含まれます。User-Personalization-v2 を使用すると、Amazon Personalize が探索設定を処理します。レコメンデーションに新しいアイテムが含まれるようにするには、プロモーションを使用して、作成タイムスタンプに基づいて新しいアイテムを含めることができます。プロモーションの詳細については、「」を参照してくださいレコメンデーション内のアイテムのプロモーション。

- 自動更新 – 自動更新では、Amazon Personalize は 2 時間ごとに最新のモデル (ソリューションバージョン) を自動的に更新し、新しいアイテムをレコメンデーションの対象として検討します。詳細については、「自動更新」を参照してください。
- レコメンデーションを含むメタデータ – User-Personalization-v2 レシピでは、メタデータの列が少なくとも 1 つある Items データセットがある場合、キャンペーンにはレコメンデーション結果にアイテムメタデータを含めるオプションが自動的にあります。キャンペーンのメタデータを手動で有効にする必要はありません。メタデータを使用して、映画のジャンルをカルーセルに追加するなど、ユーザーインターフェイスのレコメンデーションを充実させることができます。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。

必須およびオプションのデータセット

User-Personalization-v2 を使用するには、 を作成し、少なくとも 1000 件のアイテムインタラクションアイテムインタラクションデータセットをインポートする必要があります。Amazon Personalize は、主にアイテムインタラクションデータに基づいてレコメンデーションを生成します。User-Personalization-v2 は、アイテムインタラクションとアイテムデータセット全体で最大 500 万のアイテムでトレーニングできます。

User-Personalization-v2 を使用すると、Amazon Personalize は以下を含むアイテムインタラクションデータを使用できます。

- イベントタイプとイベント値データ – Amazon Personalize は、クリックイベントタイプや監視イベントタイプなどのイベントタイプデータを使用して、ユーザーの行動のパターンを通じてユーザーの意図と関心を特定します。また、イベントタイプとイベント値のデータを使用して、トレーニング前にレコードをフィルタリングすることもできます。詳細については、「イベントタイプとイベント値のデータ」を参照してください。

  > **Note**
  >
  > User-Personalization-v2 では、トレーニングコストはイベントタイプまたは値でフィルタリングする前にインタラクションデータに基づいています。料金の詳細については、「Amazon Personalize の料金」を参照してください。

- コンテキストメタデータ — コンテキストメタデータは、イベント発生時にユーザーの環境で収集する、場所やデバイスタイプなどのインタラクションデータです。詳細については、「コンテキストメタデータ」を参照してください。

以下のデータセットはオプションであり、レコメンデーションを改善できます。

- ユーザーデータセット – Amazon Personalize は、ユーザーデータセットのデータを使用して、ユーザーとその関心をよりよく理解できます。Users データセットのデータを使用してレコメンデーションをフィルタリングすることもできます。インポートできるユーザーデータについては、「ユーザーデータセット」を参照してください。
- Items データセット – Amazon Personalize は Items データセット内のデータを使用して、それらの動作における接続とパターンを識別できます。これにより、Amazon Personalize はユーザーとその関心について理解しやすくなります。Items データセットのデータを使用してレコメンデーションをフィルタリングすることもできます。インポートできるアイテムデータについては、「製品データセット」を参照してください。

プロパティおよびハイパーパラメータ

User-Personalization-v2 レシピには、次のプロパティがあります。

- 名前 – `aws-user-personalization-v2`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-user-personalization-v2`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-user-personalization-v2`

詳細については、「レシピの選択」を参照してください。

次の表に、User-Personalization-v2 レシピのハイパーパラメータを示します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。User-Personalization-v2 では、自動トレーニングを有効にすると、Amazon Personalize は 90 日ごとに自動的に HPO を実行します。自動トレーニングがないと、HPO は発生しません。

このテーブルには、各ハイパーパラメータに関する以下の情報が含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)

名前	説明
アルゴリズムのハイパーパラメータ	
apply_recency_bias	モデルがアイテムインタラクションデータセット内の最新のアイテムインタラクションデータにより多くの重みを与えるかどうかを決定します。最新のインタラクションデータには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる場合があります。 最近のイベントにより多くの重みを置くモデルをトレーニングするには、apply_recency_bias を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、apply_recency_bias を false に設定します。 デフォルト値: true 範囲: true または false 値の型: ブール値 HPO 調整可能: いいえ

User-Personalization レシピ

> **Important**
>
> User-Personalization-v2レシピを使用することをお勧めします。トレーニングを高速化して最大 500 万項目を検討し、レイテンシーを低く抑えてより関連性の高いレコメンデーションを生成できます。

User-Personalization (aws-user-personalization) レシピは、パーソナライズされたすべてのレコメンデーションシナリオに最適化されています。ユーザーが操作する可能性が最も高い項目を予測します。User-Personalization を使用して、ストリーミングアプリケーション用にパーソナライズされた

映画のレコメンデーションを生成したり、小売アプリケーション用にパーソナライズされた製品のレコメンデーションを生成したりできます。

User-Personalization を使用すると、Amazon Personalize は主にアイテムインタラクションデータセット内のユーザーアイテムインタラクションデータに基づいてレコメンデーションを生成します。Items and Users データセットでは、任意のアイテムとユーザーメタデータを使用することもできます。使用するデータの詳細については、「」を参照してください必須およびオプションのデータセット。

トピック

- レシピ機能
- 必須およびオプションのデータセット
- プロパティおよびハイパーパラメータ
- User-Personalization レシピを使用したトレーニング (コンソール)
- User-Personalization レシピを使用したトレーニング (Python SDK)
- レコメンデーションの取得とインプレッションの記録 (SDK for Python (Boto3))
- サンプル Jupyter ノートブック

レシピ機能

User-Personalization は、アイテムレコメンデーションを生成するときに、次の Amazon Personalize レシピ機能を使用します。

- リアルタイムパーソナライゼーション — リアルタイムパーソナライゼーションでは、Amazon Personalize はユーザーの関心の高まりに応じてアイテムのレコメンデーションを更新および適応します。詳細については、「リアルタイムパーソナライゼーション」を参照してください。
- 探索 — 探索では、レコメンデーションには新しいアイテムやインタラクションデータが少ないアイテムが含まれます。これにより、カタログの内容の変更が早い場合、またはニュース記事やプロモーションなどの新しいアイテムとユーザーとの関連性が高い場合、それらのアイテムの鮮度が高い状態であるときに、アイテムの検出とエンゲージメントが向上します。探索の詳細については、「探査」を参照してください。
- 自動更新 – 自動更新では、Amazon Personalize は 2 時間ごとに最新のモデル (ソリューションバージョン) を自動的に更新し、新しいアイテムをレコメンデーションの対象として検討します。詳細については、「自動更新」を参照してください。

必須およびオプションのデータセット

User-Personalization を使用するには、 を作成し、少なくとも 1000 件のアイテムインタラクションアイテムインタラクションデータセットをインポートする必要があります。Amazon Personalize は、主にアイテムインタラクションデータに基づいてレコメンデーションを生成します。

User-Personalization を使用すると、Amazon Personalize は以下を含むアイテムインタラクションデータを使用できます。

- イベントタイプとイベント値データ – Amazon Personalize は、クリックイベントタイプや監視イベントタイプなどのイベントタイプデータを使用して、ユーザーの行動のパターンを通じてユーザーの意図と関心を特定します。また、イベントタイプとイベント値のデータを使用して、トレーニング前にレコードをフィルタリングすることもできます。詳細については、「イベントタイプとイベント値のデータ」を参照してください。
- コンテキストメタデータ – コンテキストメタデータは、イベント発生時にユーザーの環境で収集する、場所やデバイスタイプなどのインタラクションデータです。詳細については、「コンテキストメタデータ」を参照してください。
- インプレッションデータ – インプレッションは、ユーザーが特定のアイテムとやり取りした (クリックした、視聴した、購入したなど) ときに表示されていたアイテムのリストです。詳細については、「インプレッションデータ」を参照してください。

以下のデータセットはオプションであり、レコメンデーションを改善できます。

- ユーザーデータセット – Amazon Personalize は、ユーザーデータセットのデータを使用して、ユーザーとその関心をよりよく理解できます。Users データセットのデータを使用してレコメンデーションをフィルタリングすることもできます。インポートできるユーザーデータについては、「ユーザーデータセット」を参照してください。
- Items データセット – Amazon Personalize は Items データセット内のデータを使用して、それらの動作における接続とパターンを識別できます。これにより、Amazon Personalize はユーザーとその関心について理解しやすくなります。Items データセットのデータを使用してレコメンデーションをフィルタリングすることもできます。インポートできるアイテムデータについては、「製品データセット」を参照してください。

プロパティおよびハイパーパラメータ

User-Personalization レシピには、次のプロパティがあります。

- 名前 – `aws-user-personalization`

- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-user-personalization`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-user-personalization`

詳細については、「レシピの選択」を参照してください。

以下の表では、User-Personalization レシピのハイパーパラメータを示しています。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報が含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータが HPO に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	
`hidden_dimension`	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。アイテムインタラクションデータセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、データセットが大きくなり、処理時間が長くなります。最適な値を決定するには、HPO を使用します。HPO を使用するには、CreateSolution オペレーション CreateSolutionVersion とオペレーションを呼び出すときに `performHPO` を `true` に設定します。 デフォルト値: 149

名前	説明
	範囲: [32, 256] 値の型: 整数 HPO 調整可能: はい
bptt	通し時間のバックプロパゲーションの手法を使用するかどうかを決定します。通し時間のバックプロパゲーションは、再帰的なニューラルネットワークベースのアルゴリズムの重みを更新する手法です。遅延報酬を早期イベントに接続するには、長期クレジットに bptt を使用します。例えば、遅延報酬には、数回のクリック後に行われる購入を指定できます。早期イベントは、最初のクリックにすることができます。クリックなどの同じイベントタイプ内であっても、長期的な効果を考慮し、合計報酬を最大化することをお勧めします。長期的な効果を考慮するには、より大きい bptt 値を使用します。大きな bptt 値を使用するには、より大きいデータセットと長い処理時間が必要です。 デフォルト値: 32 範囲: [2, 32] 値の型: 整数 HPO 調整可能: はい

名前	説明
recency_mask	モデルがアイテムインタラクションデータセットの最新の人気傾向を考慮する必要があるかどうかを決定します。最新の人気トレンドには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる可能性があります。最近のイベントにより多くの重みを置くモデルをトレーニングするには、recency_mask を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、recency_mask を false に設定します。同じ重みを使用して適切なレコメンデーションを取得するには、より大きなトレーニングデータセットが必要になる場合があります。 デフォルト値: True 範囲: True または False 値の型: ブール値 HPO 調整可能: はい
特徴化のハイパーパラメータ	

名前	説明
min_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最小パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。履歴の長さが短いある割合のユーザーを除外するには、min_user_history_length_percentile を使用します。履歴が短いユーザーは、多くの場合、ユーザーの個人的なニーズや希望ではなく、アイテムの人気に基づくパターンを表示します。それらを削除すると、データの基盤となるパターンに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min_user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.0 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最大パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。max_user_history_length_percentile を使用して、ある割合の履歴の長さが長いユーザーを除外します。これらのユーザーのデータにはノイズが含まれる傾向があるためです。例えば、ロボットに自動化されたインタラクションの長いリストがあるとします。これらのユーザーを削除することで、トレーニングのノイズが制限されます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min_user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.99 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ
アイテム探索キャンペーン設定ハイパーパラメータ	

名前	説明
exploration_weight	アイテムインタラクションデータまたは関連性が少ないアイテムをレコメンデーションに含める頻度を決定します。値が 1.0 に近くなるほど、探索が多くなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。詳細については、「the section called “CampaignConfig”」を参照してください。 デフォルト値: 0.3 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
exploration_item_age_cut_of f	アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は、作成タイムスタンプ、または作成タイムスタンプデータがない場合はアイテムインタラクションデータに基づいてアイテムの年齢を決定します。Amazon Personalize が商品の年齢を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。 Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。これは、これらのアイテムがユーザーに関連しており、それらを特定するために探索が必要ではなかったことによります。 デフォルト値: 30.0 範囲: 正の浮動小数点 値の型: 浮動小数点 HPO 調整可能: いいえ

User-Personalization レシピを使用したトレーニング (コンソール)

User-Personalization レシピを使用してコンソールでレコメンデーションを生成するには、最初にレシピを使用して新しいソリューションバージョンをトレーニングします。その後、ソリューションバージョンを使用してキャンペーンをデプロイし、キャンペーンを使用してレコメンデーションを取得します。

User-Personalization レシピを使用した新しいソリューションバージョンのトレーニング (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. 新しいスキーマを使用してカスタムデータセットグループを作成し、インプレッションデータを含むデータセットをアップロードします。オプションで、CREATION_TIMESTAMP と 非構造化テキストメタデータ データを Items データセットに含めて、Amazon Personalize がアイテムが存在するようになってからの期間をより正確に計算し、コールドアイテムを識別できるようにします。

   データのインポートの詳細については、「ステップ 2: データの準備とインポート」を参照してください。
3. [Dataset groups] (データセット) グループページで、インプレッションデータを含む 1 つまたは複数のデータセットを含む新しいデータセットグループを選択します。
4. ナビゲーションペインで、[Solutions and recipes] (ソリューションとレシピ)、[Create solution] (ソリューションを作成) の順に選択します。
5. [Create solution] (ソリューションを作成) のページの [Solution name] (ソリューション名) で、新しいソリューションの名前を入力します。
6. [Solution type] (ソリューションタイプ) で、[Item recommendation] (アイテムのレコメンデーション) を選択して、ユーザーのアイテムのレコメンデーションを取得します。
7. レシピ で、 を選択しますaws-user-personalization。[Solution configuration] (ソリューション設定) のセクションが表示され、いくつかの設定オプションが提供されます。
8. 追加設定 で、アイテムインタラクションデータセットに EVENT_TYPE または EVENT_TYPE 列と EVENT_VALUE 列の両方がある場合、オプションでイベントタイプフィールドとイベント値のしきい値フィールドを使用して、Amazon Personalize がモデルのトレーニング時に使用するアイテムインタラクションデータを選択します。詳細については、「トレーニングに使用するアイテムインタラクションデータの選択」を参照してください。
9. オプションで、ソリューションのハイパーパラメータを設定します。User-Personalization レシピのプロパティとハイパーパラメータのリストについては、「プロパティおよびハイパーパラメータ」を参照してください。
10. [Create and train solution] (ソリューションを作成およびトレーニング) を選択して、トレーニングを開始します。[Dashboard] (ダッシュボード) のページが表示されます。

ソリューションの詳細のページに移動して、[Solution versions] (ソリューションバージョン) のセクションでトレーニングの進捗状況を追跡できます。トレーニングが完了すると、ステータスは [Active] (アクティブ) となります。

キャンペーンの作成とレコメンデーションの取得 (コンソール)

ソリューションバージョンのステータスが [Active] (アクティブ) になると、次のように、キャンペーンを作成してレコメンデーションを取得する準備が完了します。

1. ソリューションの詳細のページまたは [Campaigns] (キャンペーン) のページで、[Create new campaign] (新しいキャンペーンを作成) を選択します。
2. [Create new campaign] (新しいキャンペーンを作成) ページの [Campaign details] (キャンペーンの詳細) で、次の情報を入力します。

   - Campaign name (キャンペーン名): キャンペーンの名前を入力します。ここで入力したテキストは、[Campaign] (キャンペーン) ダッシュボードと詳細のページに表示されます。
   - Solution (ソリューション): 先ほど作成したソリューションを選択します。
   - Solution version ID (ソリューションバージョン ID): 作成したソリューションバージョンの ID を選択します。
   - Minimum provisioned transactions per second (1 秒あたりの最小プロビジョンドトランザクション): Amazon Personalize がサポートしている 1 秒あたりの最小プロビジョンドトランザクションを設定します。詳細については、CreateCampaign オペレーションを参照してください。
3. [Campaign configuration] (キャンペーン設定) で、次の情報を入力してください。

   - Exploration weight: (探索の重み): 探索する量を設定します。指定する探索が多いほど、レコメンデーションには、アイテムインタラクションデータや関連性が少ないアイテムがより頻繁に含まれます。値が 1 に近くなるほど、探索が多くなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
   - Exploration item age cut off (探索アイテムが存在するようになってからの期間のカットオフ): アイテム探索の範囲を定義するために、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を入力します。Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。

     例えば、10 と入力すると、データセット内の最新のインタラクションから 10 日間のアイテムインタラクションデータを含むアイテムのみが探索時に考慮されます。

> **Note**
>
> レコメンデーションには、この時間枠外のアイテムインタラクションデータのないアイテムが含まれる場合があります。これは、これらのアイテムがユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったことによります。

4. [キャンペーンの作成] を選択します。
5. キャンペーンの詳細のページで、キャンペーンのステータスが [Active] (アクティブ) の場合、キャンペーンを使用してレコメンデーションを取得し、インプレッションを記録できます。詳細については、「開始方法」の「ステップ 5: レコメンデーションを取得する」を参照してください。

   Amazon Personalize は、2 時間ごとに最新のソリューションバージョンを自動的に更新して、新しいデータを含めます。キャンペーンでは、更新されたソリューションバージョンが自動的に使用されます。詳細については、「自動更新」を参照してください。

   キャンペーンを手動で更新するには、最初に、`trainingMode` を `update` に設定した状態で、コンソールまたは CreateSolutionVersion 操作を使用して、新しいソリューションバージョンを作成およびトレーニングします。その後、コンソールの [Campaign] (キャンペーン) ページで、または UpdateCampaign 操作を使用して、キャンペーンを手動で更新します。

   > **Note**
   >
   > Amazon Personalize は、2020 年 11 月 17 日より前に作成したソリューションバージョンを自動的に更新しません。

User-Personalization レシピを使用したトレーニング (Python SDK)

データセットグループを作成し、インプレッションデータを含むデータセットをアップロードしたら、User-Personalization レシピを使用してソリューションをトレーニングできます。オプションで、CREATION_TIMESTAMP と 非構造化テキストメタデータ データを Items データセットに含めて、Amazon Personalize がアイテムが存在するようになってからの期間をより正確に計算し、コールドアイテムを識別できるようにします。データセットグループの作成とトレーニングデータのアップロードの詳細については、「スキーマ」を参照してください。

AWS SDK を使用して User-Personalization レシピでソリューションをトレーニングするには

1. `create_solution` メソッドを使用して新しいソリューションを作成します。

   `solution name` をソリューション名に置き換えます。また、`dataset group arn` をデータセットグループの Amazon リソースネーム (ARN) に置き換えます。

```
import boto3

personalize = boto3.client('personalize')

print('Creating solution')
create_solution_response = personalize.create_solution(name = 'solution name',
                           recipeArn = 'arn:aws:personalize:::recipe/aws-user-
personalization',
                           datasetGroupArn = 'dataset group arn',
                           )
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

   aws-user-personalization レシピプロパティとハイパーパラメータのリストについては、「」を参照してくださいプロパティおよびハイパーパラメータ。

2. 更新されたトレーニングデータを使用して新しい solution version (ソリューションバージョン) を作成し、次のコードスニペットを使用して `trainingMode` を `FULL` に設定します。`solution arn` をソリューションの ARN に置き換えます。

```
import boto3

personalize = boto3.client('personalize')

create_solution_version_response = personalize.create_solution_version(solutionArn
 = 'solution arn',
                                                        trainingMode='FULL')

new_solution_version_arn = create_solution_version_response['solutionVersionArn']
print('solution_version_arn:', new_solution_version_arn)
```

3. Amazon Personalize がソリューションバージョンの作成を終了したら、次のパラメータを使用してキャンペーンを作成します。

- 新しい campaign name と、ステップ 2 で生成された solution version arn を入力します。
- explorationWeight アイテム探索設定ハイパーパラメータを変更して、探索する量を設定します。アイテムインタラクションデータまたは関連性が少ないアイテムは、値が 1.0 に近いほど頻繁に推奨されます。デフォルト値は 0.3 です。
- explorationItemAgeCutOff アイテム探索設定ハイパーパラメータパラメータを変更して、アイテム探索の対象期間とする、最新のインタラクションとの関係においての最長期間 (日) を指定します。値が大きいほど、探索中に考慮されるアイテムが多くなります。

次の Python スニペットを使用して、30 日で探索をカットオフする探索に重点を置いた新しいキャンペーンを作成します。キャンペーンの作成には通常数分かかります。ただし、1 時間以上かかる場合もあります。

```
import boto3

personalize = boto3.client('personalize')

create_campaign_response = personalize.create_campaign(
    name = 'campaign name',
    solutionVersionArn = 'solution version arn',
    minProvisionedTPS = 1,
    campaignConfig = {"itemExplorationConfig": {"explorationWeight": "0.3",
 "explorationItemAgeCutOff": "30"}}
)

campaign_arn = create_campaign_response['campaignArn']
print('campaign_arn:', campaign_arn)
```

User-Personalization を使用すると、Amazon Personalize は、2 時間ごとにソリューションバージョンを自動的に更新して、新しいデータを含めます。キャンペーンでは、更新されたソリューションバージョンが自動的に使用されます。詳細については、「自動更新」を参照してください。

キャンペーンを手動で更新するには、最初に、trainingMode を update に設定した状態で、コンソールまたは CreateSolutionVersion 操作を使用して、新しいソリューションバージョンを作成およびトレーニングします。その後、コンソールの [Campaign] (キャンペーン) ページで、または UpdateCampaign 操作を使用して、キャンペーンを手動で更新します。

> **Note**
>
> Amazon Personalize は、2020 年 11 月 17 日より前に作成したソリューションバージョンを自動的に更新しません。

レコメンデーションの取得とインプレッションの記録 (SDK for Python (Boto3))

キャンペーンが作成されると、それを使用してユーザーのレコメンデーションを取得し、インプレッションを記録できます。 AWS SDKs「」を参照してくださいバッチ推論ジョブの作成 (AWS SDKs)。

レコメンデーションを取得してインプレッションを記録するには

1. `get_recommendations` メソッドを呼び出します。`campaign arn` を新しいキャンペーンの ARN に変更します。また、`user id` をユーザーの userId に変更します。

```
import boto3

rec_response = personalize_runtime.get_recommendations(campaignArn = 'campaign
 arn', userId = 'user id')
print(rec_response['recommendationId'])
```

2. PutEvents リクエストを送信するための新しいイベントトラッカーを作成します。`event tracker name` をイベントトラッカーの名前に置き換えます。また、`dataset group arn` をデータセットグループの ARN に置き換えます。

```
import boto3

personalize = boto3.client('personalize')

event_tracker_response = personalize.create_event_tracker(
    name = 'event tracker name',
    datasetGroupArn = 'dataset group arn'
)
event_tracker_arn = event_tracker_response['eventTrackerArn']
event_tracking_id = event_tracker_response['trackingId']
print('eventTrackerArn:{},\n eventTrackingId:{}'.format(event_tracker_arn,
 event_tracking_id))
```

3. ステップ 1 の recommendationId とステップ 2 の event tracking id を使用して、新しい PutEvents リクエストを作成します。このリクエストは、ユーザーのセッションからの新しいインプレッションデータを口グに記録します。user id をユーザーの ID に変更します。

```
import boto3

personalize_events.put_events(
     trackingId = 'event tracking id',
     userId= 'user id',
     sessionId = '1',
     eventList = [{
     'sentAt': datetime.now().timestamp(),
     'eventType' : 'click',
     'itemId' : rec_response['itemList'][0]['itemId'],
     'recommendationId': rec_response['recommendationId'],
     'impression': [item['itemId'] for item in rec_response['itemList']],
     }]
)
```

サンプル Jupyter ノートブック

User-Personalization レシピの使用方法を示すサンプル Jupyter ノートブックについては、「探索での User Personalization」を参照してください。

レガシーユーザーのパーソナライゼーションレシピ

> **Note**
>
> レガシー HRNN レシピは利用できなくなりました。このドキュメントは参照用です。レガシー HRNN レシピよりも aws-user-personalizaton (User-Personalization) レシピを使用することをお勧めします。User-Personalization では、HRNN レシピによって提供される機能が改善および統合されています。詳細については、「User-Personalization レシピ」を参照してください。

以下は、レガシー USER_PERSONALIZATION レシピです。

- HRNN レシピ (レガシー)

- HRNN-Coldstart レシピ (レガシー)
- HRNN-Metadata レシピ (レガシー)

HRNN レシピ (レガシー)

> **Note**
>
> レガシー HRNN レシピは利用できなくなりました。このドキュメントは参照用です。レガシー HRNN レシピではなく、aws-user-personalizaton (User-Personalization) レシピを使用することをお勧めします。User-Personalization では、HRNN レシピによって提供される機能が改善および統合されています。詳細については、「User-Personalization レシピ」を参照してください。

Amazon Personalize 階層的再帰型ニューラルネットワーク (HRNN) レシピでは、セッション中にレコメンデーションを提供するために、ユーザーの行動の変化をモデル化します。セッションとは、例えば、必要を満たすために特定のアイテムを見つけることを目的とする、特定の期間内の一連のユーザーインタラクションです。ユーザーの最近のインタラクションの重み付けを高くすることで、セッション中により関連性の高いレコメンデーションを提供できます。

HRNN は、時間の経過とともに変化する可能性がある、ユーザーのインテントや関心に対応します。また、順序付けられたユーザー履歴を取得し、自動的に重み付けして推論を改善します。HRNN は、ゲート機構を使用して減損の重みをアイテムとタイムスタンプの学習可能な関数としてモデル化します。

Amazon Personalize は、データセットから各ユーザーの特徴を派生させます。リアルタイムのデータ統合を完了している場合、これらの特徴はユーザーアクティビティに応じてリアルタイムで更新されます。レコメンデーションを取得するには、`USER_ID` のみを指定します 。`ITEM_ID` も指定すると、Amazon Personalize はそれを無視します。

HRNN レシピには以下のプロパティがあります。

- 名前 - `aws-hrnn`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-hrnn`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-hrnn`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/JSON-percentile-filtering`

- レシピタイプ – USER_PERSONALIZATION

以下の表では、HRNN レシピのハイパーパラメータについて説明します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータが HPO に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	
hidden_dimension	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。アイテムインタラクションデータセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、データセットが大きくなり、処理時間が長くなります。最適な値を決定するには、HPO を使用します。HPO を使用するには、CreateSolution オペレーション CreateSolutionVersion とオペレーションを呼び出すときに performHPO を true に設定します。 デフォルト値: 43 範囲: [32, 256] 値の型: 整数

名前	説明
	HPO 調整可能: はい
bptt	通し時間のバックプロパゲーションの手法を使用するかどうかを決定します。通し時間のバックプロパゲーションは、再帰的なニューラルネットワークベースのアルゴリズムの重みを更新する手法です。遅延報酬を早期イベントに接続するには、長期クレジットに bptt を使用します。例えば、遅延報酬には、数回のクリック後に行われる購入を指定できます。早期イベントは、最初のクリックにすることができます。クリックなどの同じイベントタイプ内であっても、長期的な効果を考慮し、合計報酬を最大化することをお勧めします。長期的な効果を考慮するには、より大きい bptt 値を使用します。大きな bptt 値を使用するには、より大きいデータセットと長い処理時間が必要です。 デフォルト値: 32 範囲: [2, 32] 値の型: 整数 HPO 調整可能: はい

名前	説明
recency_mask	モデルがアイテムインタラクションデータセットの最新の人気傾向を考慮する必要があるかどうかを決定します。最新の人気トレンドには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる可能性があります。最近のイベントにより多くの重みを置くモデルをトレーニングするには、recency_mask を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、recency_mask を false に設定します。同じ重みを使用して適切なレコメンデーションを取得するには、より大きなトレーニングデータセットが必要になる場合があります。 デフォルト値: True 範囲: True または False 値の型: ブール値 HPO 調整可能: はい
特徴化のハイパーパラメータ	

名前	説明
min_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最小パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。履歴の長さが短いある割合のユーザーを除外するには、min_user_history_length_percentile を使用します。履歴が短いユーザーは、多くの場合、ユーザーの個人的なニーズや希望ではなく、アイテムの人気に基づくパターンを表示します。それらを削除すると、データの基盤となるパターンに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.0 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最大パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。max_user_history_length_percentile を使用して、ある割合の履歴の長さが長いユーザーを除外します。これらのユーザーのデータにはノイズが含まれる傾向があるためです。例えば、ロボットに自動化されたインタラクションの長いリストがあるとします。これらのユーザーを削除することで、トレーニングのノイズが制限されます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.99 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

HRNN-Metadata レシピ (レガシー)

> **Note**
>
> レガシー HRNN レシピは利用できなくなりました。このドキュメントは参照用です。レガシー HRNN レシピではなく、aws-user-personalizaton (User-Personalization) レシピを使用することをお勧めします。User-Personalization では、HRNN レシピによって提供され

る機能が改善および統合されています。詳細については、「User-Personalization レシピ」を参照してください。

HRNN- メタデータレシピは、ユーザーがやり取りするアイテムを予測します。HRNN レシピに類似していますが、コンテキスト、ユーザー、アイテムのメタデータ (それぞれソースは、インタラクション、ユーザー、アイテムのデータセット) から派生した追加の特徴を含みます。高品質なメタデータを使用できる場合、HRNN-Metadata は非メタデータモデルよりも結果が正確になります。このレシピを使用するには、より長いトレーニング時間が必要になる場合があります。

HRNN-Metadata レシピには以下のプロパティがあります。

- 名前 - `aws-hrnn-metadata`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-hrnn-metadata`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-hrnn-metadata`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/featurize_metadata`
- レシピタイプ – `USER_PERSONALIZATION`

以下の表では、HRNN-Metadata レシピのハイパーパラメータについて説明します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータがハイパーパラメータ最適化 (HPO) に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	

名前	説明
hidden_dimension	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。アイテムインタラクションデータセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、データセットが大きくなり、処理時間が長くなります。最適な値を決定するには、HPO を使用します。HPO を使用するには、CreateSolution オペレーション CreateSolutionVersion とオペレーションを呼び出すときに performHPO を true に設定します。 デフォルト値: 43 範囲: [32, 256] 値の型: 整数 HPO 調整可能: はい

名前	説明
bptt	通し時間のバックプロパゲーションの手法を使用するかどうかを決定します。通し時間のバックプロパゲーションは、再帰的なニューラルネットワークベースのアルゴリズムの重みを更新する手法です。遅延報酬を早期イベントに接続するには、長期クレジットに bptt を使用します。例えば、遅延報酬には、数回のクリック後に行われる購入を指定できます。早期イベントは、最初のクリックにすることができます。クリックなどの同じイベントタイプ内であっても、長期的な効果を考慮し、合計報酬を最大化することをお勧めします。長期的な効果を考慮するには、より大きい bptt 値を使用します。大きな bptt 値を使用するには、より大きいデータセットと長い処理時間が必要です。 デフォルト値: 32 範囲: [2, 32] 値の型: 整数 HPO 調整可能: はい

名前	説明
recency_mask	モデルがアイテムインタラクションデータセットの最新の人気傾向を考慮する必要があるかどうかを決定します。最新の人気トレンドには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる可能性があります。最近のイベントにより多くの重みを置くモデルをトレーニングするには、recency_mask を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、recency_mask を false に設定します。同じ重みを使用して適切なレコメンデーションを取得するには、より大きなトレーニングデータセットが必要になる場合があります。 デフォルト値: True 範囲: True または False 値の型: ブール値 HPO 調整可能: はい
特徴化のハイパーパラメータ	

名前	説明
min_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最小パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。履歴の長さが短いある割合のユーザーを除外するには、min_user_history_length_percentile を使用します。履歴が短いユーザーは、多くの場合、ユーザーの個人的なニーズや希望ではなく、アイテムの人気に基づくパターンを表示します。それらを削除すると、データの基盤となるパターンに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.0 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最大パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。max_user_history_length_percentile を使用して、ある割合の履歴の長さが長いユーザーを除外します。これらのユーザーのデータにはノイズが含まれる傾向があるためです。例えば、ロボットに自動化されたインタラクションの長いリストがあるとします。これらのユーザーを削除することで、トレーニングのノイズが制限されます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.99 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

HRNN-Coldstart レシピ (レガシー)

> **Note**
>
> レガシー HRNN レシピは利用できなくなりました。このドキュメントは参照用です。レガシー HRNN レシピではなく、aws-user-personalizaton (User-Personalization) レシピを使用することをお勧めします。User-Personalization では、HRNN レシピによって提供され

る機能が改善および統合されています。詳細については、「User-Personalization レシピ」を参照してください。

HRNN-Coldstart レシピを使用して、ユーザーがやり取りするアイテムを予測するのは、新しいアイテムとインタラクションを頻繁に追加し、それらのアイテムのレコメンデーションをすぐに取得する必要がある場合です。HRNN-Coldstart レシピは HRNN-Metadata レシピに似ていますが、新しいアイテムのレコメンデーションを取得できます。

さらに、最近の人気傾向のためや、インタラクションが非常にまれでトレーニングのノイズである可能性があるために、インタラクションのリストが長くなっているアイテムをトレーニングアイテムから除外する場合にも、HRNN-Coldstart レシピを使用できます。HRNN-Coldstart を使用すると、関連性の低いアイテムを除外して、トレーニング用のサブセットを作成できます。このようなアイテムのサブセットはコールドアイテムと呼ばれ、関連するインタラクションイベントがアイテムインタラクションデータセットに含まれています。以下の条件に該当する場合、そのアイテムはコールドアイテムとみなされます。

- 指定された最大インタラクション数よりインタラクションが少ない。この値は、レシピの `cold_start_max_interactions` ハイパーパラメータで指定します。
- 最大期間よりも短い相対期間。この値は、レシピの `cold_start_max_duration` ハイパーパラメータで指定します。

コールドアイテムの数を減らすには、[cold_start_max_interactions] または [cold_start_max_duration] に低い値を設定します。コールドアイテムの数を増やすには、[cold_start_max_interactions] または [cold_start_max_duration] に大きな値を設定します。

HRNN-Coldstart には以下のコールドアイテムの制限があります。

- `Maximum cold start items`: 80,000
- `Minimum cold start items`: 100

コールドアイテムの数がこの範囲外の場合、ソリューション作成の試みは失敗します。

HRNN-Coldstart レシピには以下のプロパティがあります。

- 名前 - `aws-hrnn-coldstart`

- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-hrnn-coldstart`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-hrnn-coldstart`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/featurize_coldstart`
- レシピタイプ – `USER_PERSONALIZATION`

詳細については、「レシピの選択」を参照してください。

次の表では、HRNN-Coldstart レシピのハイパーパラメータについて説明します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータが HPO に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	
`hidden_dimension`	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。アイテムインタラクションデータセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、データセットが大きくなり、処理時間が長くなります。最適な値を決定するには、HPO を使用します。HPO を使用するには、CreateSolution オペレー

名前	説明
	ション CreateSolutionVersion とオペレーションを呼び出すときに performHPO を true に設定します。 デフォルト値: 149 範囲: [32, 256] 値の型: 整数 HPO 調整可能: はい
bptt	通し時間のバックプロパゲーションの手法を使用するかどうかを決定します。通し時間のバックプロパゲーションは、再帰的なニューラルネットワークベースのアルゴリズムの重みを更新する手法です。遅延報酬を早期イベントに接続するには、長期クレジットに bptt を使用します。例えば、遅延報酬には、数回のクリック後に行われる購入を指定できます。早期イベントは、最初のクリックにすることができます。クリックなどの同じイベントタイプ内であっても、長期的な効果を考慮し、合計報酬を最大化することをお勧めします。長期的な効果を考慮するには、より大きい bptt 値を使用します。大きな bptt 値を使用するには、より大きいデータセットと長い処理時間が必要です。 デフォルト値: 32 範囲: [2, 32] 値の型: 整数 HPO 調整可能: はい

名前	説明
recency_mask	モデルがアイテムインタラクションデータセットの最新の人気傾向を考慮する必要があるかどうかを決定します。最新の人気トレンドには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる可能性があります。最近のイベントにより多くの重みを置くモデルをトレーニングするには、recency_mask を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、recency_mask を false に設定します。同じ重みを使用して適切なレコメンデーションを取得するには、より大きなトレーニングデータセットが必要になる場合があります。 デフォルト値: True 範囲: True または False 値の型: ブール値 HPO 調整可能: はい
特徴化のハイパーパラメータ	
cold_start_max_interactions	アイテムをコールドアイテムとみなすことができるユーザーアイテムインタラクションの最大数。 デフォルト値: 15 範囲: 正の整数 値の型: 整数 HPO 調整可能: いいえ

名前	説明
cold_start_max_duration	コールドスタートアイテムとみなされるユーザーアイテムインタラクションの開始点からの最大日数。ユーザーアイテムインタラクションの開始点を設定するには、[cold_start_relative_from] ハイパーパラメータを設定します。 デフォルト値: 5.0 範囲: 正の浮動小数点 値の型: 浮動小数点 HPO 調整可能: いいえ
cold_start_relative_from	HRNN-Coldstart レシピによる cold_start_max_duration の計算開始点を決定します。現在の時刻から計算するには、currentTime を選択します。 アイテムインタラクションデータセットの最新のアイテムのタイムスタンプから cold_start_max_duration を計算するには、latestItem を選択します。この設定は、新しいアイテムを頻繁に追加する場合に便利です。 デフォルト値: latestItem 範囲: currentTime 、latestItem 値の型: 文字列 HPO 調整可能: いいえ

名前	説明
min_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最小パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。履歴の長さが短いある割合のユーザーを除外するには、min_user_history_length_percentile を使用します。履歴が短いユーザーは、多くの場合、ユーザーの個人的なニーズや希望ではなく、アイテムの人気に基づくパターンを表示します。それらを削除すると、データの基盤となるパターンに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.0 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最大パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。max_user_history_length_percentile を使用して、ある割合の履歴の長さが長いユーザーを除外します。これらのユーザーのデータにはノイズが含まれる傾向があるためです。例えば、ロボットに自動化されたインタラクションの長いリストがあるとします。これらのユーザーを削除することで、トレーニングのノイズが制限されます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.99 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

AutoML を使用した HRNN レシピの選択 (API のみ)

Amazon Personalize では、入力データの分析に基づいて、最適な階層的再帰型ニューラルネットワーク (HRNN) レシピが自動的に選択されるようにすることが可能です。このオプションは AutoML と呼ばれます。AutoML を実行するには、CreateSolution API を呼び出すときに performAutoML パラメータを true に設定します。

指定したメトリクスに基づいて最適なレシピを決定するために、Amazon Personalize で検討するレシピのリストを指定することもできます。この場合は、CreateSolution 操作を呼び出し、performAutoML パラメータに true を指定します。recipeArn パラメータは省略します。solutionConfig パラメータを含め、autoMLConfig オブジェクトの一部として metricName と recipeList を指定します。

次の表に、レシピの選択方法を示します。performAutoML または recipeArn のどちらかを指定する必要があります。両方を指定することはできません。AutoML は HRNN レシピを使用時のみに実行されます。

performAutoML	recipeArn	solutionConfig	結果
true	省略	省略	Amazon Personalize がレシピを選択します
true	省略	autoMLConfig : metricName および recipeList を指定	Amazon Personalize は、メトリクスを最適化するレシピをリストから選択します
省略	指定	省略	自分でレシピを選択する
省略	指定	指定	自分でレシピを指定して、デフォルトのトレーニングプロパティを上書きする

> **Note**
>
> performAutoML が true の場合、solutionConfig オブジェクトのパラメータはすべて無視されます (autoMLConfig を除く)。

POPULAR_ITEMS

最新ニュース記事や人気のソーシャルメディアコンテンツなど、トレンド商品や人気商品をレコメンドするには、POPULAR_ITEMS レシピを使用してください。ユーザーからの人気が急速に高まっている商品のレコメンデーションを作成するには、Trending-Now レシピ レシピを使用します。比較のためのベースラインを生成するには、最も人気のある上位 K 個のアイテムをレコメンドする

Popularity-Count レシピを使用することをお勧めします。この POPULAR_ITEMS レシピは、インタラクションの数に基づいて最も人気のある商品をレコメンデーションします。

POPULAR_ITEMS レシピは次のとおりです。

- Trending-Now レシピ
- Popularity-Count レシピ

Trending-Now レシピ

Trending-Now レシピ (aws-trending-now) は、ユーザーの間で急速に人気が高まっている商品のレコメンデーションを生成します。人気が高まっている商品が顧客との関連性が高い場合は、Trending-Now レシピを使用するとよいでしょう。例えば、顧客は他のユーザーとのやり取りを高く評価しているかもしれません。一般的な用途としては、話題のソーシャルメディアコンテンツ、最新ニュース記事、最近のスポーツ動画を勧めることが挙げられます。

Trending-Now は、設定可能な時間間隔における各項目のインタラクションの増加を計算して、最もトレンドの多い項目を自動的に識別します。増加率が最も高いアイテムはトレンドアイテムとみなされます。時間はアイテムインタラクションデータセットのタイムスタンプデータに基づいています。考慮されるアイテムは、一括で段階的にインポートしたインタラクションデータからのものです。インタラクションデータ内の新しい項目を検討するために、Trending-Now 用の新しいソリューションバージョンを手動で作成する必要はありません。

ソリューションの作成時に `Trend discovery frequency` を指定することで、時間間隔を指定できます。例えば、30 分のデータごとに `Trend discovery frequency` に `30 minutes` を指定すると、Amazon Personalize は前回の評価以降にインタラクションの増加率が最も高いアイテムを特定します。可能な頻度には、30 分、1 時間、3 時間、および 1 日が含まれます。インタラクションデータの分布に合った頻度を選択してください。選択した間隔でデータが欠落していると、レコメンデーションの精度が低下する可能性があります。過去 2 つの時間間隔でインタラクションがゼロをインポートした場合、Amazon Personalize はトレンド商品ではなく人気商品のみをレコメンドします。

Trending-Now では、Amazon Personalize コンソールの[テストキャンペーン] ページで GetRecommendations オペレーションを呼び出したり、おすすめ情報を取得したりできます。Amazon Personalize は、人気の高い商品を返します。リクエストで `userId` を渡すのは、それを必要とするフィルターを適用した場合に限られます。 GetRecommendations API では、`numResults`パラメータを使用して返されるトレンドアイテムの数を設定できます。Trending-Now レシピではバッチレコメンデーションを取得できません。

Trending-Now を使用するには、少なくとも 1000 個の一意の履歴インタラクションとイベントインタラクションを組み合わせたアイテムインタラクションデータセットを作成する必要があります (eventType と を指定した場合 eventValueThreshold、 でフィルタリング後) 。Trending-Now は、トレンドアイテムのレコメンデーションを生成するときに、アイテムデータセットまたはユーザーデータセットのデータを使用しません。ただし、これらのデータセットのデータに基づいてレコメンデーションをフィルタリングすることはできます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

トピック

- プロパティおよびハイパーパラメータ
- ソリューションの作成 (SDK for Python (Boto3))
- サンプル Jupyter ノートブック

プロパティおよびハイパーパラメータ

Trending-Now レシピには以下のプロパティがあります。

- 名前 – `aws-trending-now`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-trending-now`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-trending-now-custom`

詳細については、「レシピの選択」を参照してください。

以下の表では、Trending-Now レシピのハイパーパラメータについて説明しています。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータが HPO に参加できますか?

名前	説明
特徴変換ハイパーパラメーター	
`Trend discovery frequency`	Amazon Personalize がインタラクションデータを評価し、トレンドアイテムを特定する頻度を指定してください。例えば、`Trend discovery frequency` を`30 minutes` に指定すると、30 分ごとに指定すると、Amazon Personalize は 30 分間隔でのインタラクションの増加率が最も高いアイテムを特定します。 使用可能な頻度は、30 分、1 時間、3 時間、および 1 日です。インタラクションデータの分布に合った頻度を選択してください。選択した間隔でデータが欠落していると、レコメンデーションの精度が低下する可能性があります。 CreateSolution API オペレーションを使用し、値を指定しない場合、デフォルトは 2 時間ごとです。 デフォルト値: 2 時間 指定できる値: 30 分、1 時間、3 時間、1 日。 値の型: 文字列 HPO 調整可能: いいえ

ソリューションの作成 (SDK for Python (Boto3))

次のコードは、SDK for Python (Boto3) を使用して Trending-Now レシピでソリューションを作成する方法を示しています。`trend_discovery_frequency` の可能な値は `30 minutes`、`1 hour`、`3 hours`、`1 day` です。コンソールを使用して手動でテーブルを作成する方法については、「ソリューションの作成 (コンソール)」を参照してください。

```
import boto3

personalize = boto3.client("personalize")

create_solution_response = personalize_client.create_solution(
```

```
    name="solution name",
    recipeArn="arn:aws:personalize:::recipe/aws-trending-now",
    datasetGroupArn="dataset group ARN",
    solutionConfig={
        "featureTransformationParameters": {
            "trend_discovery_frequency": "1 hour"
         }
    }
)
print(create_solution_response['solutionArn'])
```

サンプル Jupyter ノートブック

Trending-Now レシピの使用方法を示すサンプル Jupyter Notebook については、Amazon Personalize サンプル GitHub リポジトリの「 Trending_now_example.ipynb」を参照してください。

Popularity-Count レシピ

Popularity-Count は、インタラクションデータに基づいて、最も人気のあるアイテムをレコメンドします。最も人気のあるアイテムは、ユニークなユーザーからのインタラクションデータが最も多いアイテムです。このレシピは、すべてのユーザーに対して同じ人気度のアイテムを返します。Popularity-Count は、ソリューションバージョンを作成するときに Amazon Personalize が生成する評価メトリクスを使用して他のレシピと比較するための優れたベースラインです。詳細については、「メトリクスを使用してソリューションバージョンを評価する」を参照してください。

ソリューションバージョンを作成した後は、必ずソリューションバージョンとデータを最新の状態に保ってください。Popularity-Count では、Amazon Personalize の新しいソリューションバージョンを手動で作成 (モデルを再トレーニング) して、レコメンデーション用に新しいアイテムを検討し、ユーザーの最新の動作でモデルを更新する必要があります。次に、ソリューションバージョンを使用してキャンペーンを更新する必要があります。詳細については、「レコメンデーションの関連性の維持」を参照してください。

この事前定義済みレシピには以下のプロパティがあります。

- 名前 – `aws-popularity-count`
- レシピ ARN – `arn:aws:personalize:::recipe/aws-popularity-count`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-popularity-count`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/sims`
- レシピタイプ – `USER_PERSONALIZATION`

Popularity-Count には公開されているハイパーパラメータがありません。

PERSONALIZED_RANKING

PERSONALIZED_RANKING レシピは、予測される関心レベルに基づいてランク付けされた順序でレコメンデーションを提供します。

PERSONALIZED_RANKING レシピは次のとおりです。

トピック

- Personalized-Ranking-v2 レシピ
- Personalized-Ranking レシピ

Personalized-Ranking-v2 レシピ

Personalized-Ranking-v2 レシピは、アイテムのパーソナライズされたランキングを生成します。パーソナライズされたランキングは、特定のユーザーについて関連性によって再ランク付けされる推奨アイテムのリストです。これは、検索結果、プロモーション、厳選されたリストなど、順序付けされたアイテムのコレクションがあり、ユーザーごとにパーソナライズされた再ランク付けを提供する場合に有益です。

Personalized-Ranking-v2 は、アイテムインタラクションとアイテムデータセットから最大 500 万のアイテムでトレーニングできます。また、 よりも低いレイテンシーでより正確なランキングを生成しますPersonalized-Ranking。

Personalized-Ranking-v2 を使用する場合は、GetPersonalizedRankingAPI オペレーションでランク付けする項目を指定します。インタラクションデータなしでアイテムを指定すると、Amazon Personalize は GetPersonalizedRanking API レスポンスでレコメンデーションスコアなしでこれらのアイテムを返します。

このレシピは、トランスフォーマーベースのアーキテクチャを使用して、コンテキストを学習し、データ内の関係とパターンを追跡するモデルをトレーニングします。トランスフォーマーは、入力シーケンスを出力シーケンスに変換または変更するニューラルネットワークアーキテクチャの一種です。Amazon Personalize の場合、入力シーケンスはデータ内のユーザーのアイテムインタラクション履歴です。出力シーケンスは、パーソナライズされたレコメンデーションです。トランスフォーマーの詳細については、 AWS クラウドコンピューティングの概念ハブの「人工知能におけるトランスフォーマーとは」を参照してください。

Personalized-Ranking-v2 は、他のレシピとは異なる料金モデルを使用します。料金の詳細については、「Amazon Personalize の料金」を参照してください。

トピック

- レシピ機能
- 必須およびオプションのデータセット
- プロパティおよびハイパーパラメータ

レシピ機能

Personalized-Ranking-v2 は、項目をランク付けするときに次の Amazon Personalize レシピ機能を使用します。

- リアルタイムパーソナライゼーション — リアルタイムパーソナライゼーションでは、Amazon Personalize はユーザーの関心の高まりに応じてアイテムのレコメンデーションを更新および適応します。詳細については、「リアルタイムパーソナライゼーション」を参照してください。
- レコメンデーションを含むメタデータ – Personalized-Ranking-v2 レシピでは、メタデータの列が少なくとも 1 つある Items データセットがある場合、キャンペーンにはレコメンデーション結果にアイテムメタデータを含めるオプションが自動的にあります。キャンペーンのメタデータを手動で有効にする必要はありません。メタデータを使用して、映画のジャンルをカルーセルに追加するなど、ユーザーインターフェイスのレコメンデーションを充実させることができます。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。

必須およびオプションのデータセット

Personalized-Ranking-v2 を使用するには、 を作成し、少なくとも 1000 件のアイテムインタラクションアイテムインタラクションデータセットをインポートする必要があります。Amazon Personalize は、主にアイテムインタラクションデータに基づいてランキングを生成します。Personalized-Ranking-v2 は、アイテムインタラクションとアイテムデータセット全体で最大 500 万のアイテムでトレーニングできます。

Personalized-Ranking-v2 を使用すると、Amazon Personalize は以下を含むアイテムインタラクションデータを使用できます。

- イベントタイプとイベント値データ – Amazon Personalize は、クリックイベントタイプや監視イベントタイプなどのイベントタイプデータを使用して、ユーザーの行動のパターンを通じてユーザーの意図と関心を特定します。また、イベントタイプとイベント値のデータを使用して、トレー

ニング前にレコードをフィルタリングすることもできます。詳細については、「イベントタイプとイベント値のデータ」を参照してください。

> **Note**
>
> Personalized-Ranking-v2 では、トレーニングコストはイベントタイプまたは値でフィルタリングする前にインタラクションデータに基づいています。料金の詳細については、「Amazon Personalize の料金」を参照してください。

- コンテキストメタデータ — コンテキストメタデータは、イベント発生時にユーザーの環境で収集する、場所やデバイスタイプなどのインタラクションデータです。詳細については、「コンテキストメタデータ」を参照してください。

以下のデータセットはオプションであり、レコメンデーションを改善できます。

- ユーザーデータセット – Amazon Personalize は、ユーザーデータセットのデータを使用して、ユーザーとその関心をよりよく理解できます。Users データセットのデータを使用してレコメンデーションをフィルタリングすることもできます。インポートできるユーザーデータについては、「ユーザーデータセット」を参照してください。
- Items データセット – Amazon Personalize は Items データセット内のデータを使用して、それらの動作における接続とパターンを識別できます。これにより、Amazon Personalize はユーザーとその関心について理解しやすくなります。Items データセットのデータを使用してレコメンデーションをフィルタリングすることもできます。インポートできるアイテムデータについては、「製品データセット」を参照してください。

プロパティおよびハイパーパラメータ

Personalized-Ranking-v2 レシピには、次のプロパティがあります。

- 名前 – `aws-personalized-ranking-v2`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-personalized-ranking-v2`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-personalized-ranking-v2`

詳細については、「レシピの選択」を参照してください。

次の表に、Personalized-Ranking-v2 レシピのハイパーパラメータを示します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。Personalized-Ranking-v2 では、自動トレーニングを有効にすると、Amazon Personalize は 90 日ごとに自動的に HPO を実行します。自動トレーニングがないと、HPO は発生しません。

このテーブルには、各ハイパーパラメータに関する以下の情報が含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)

名前	説明
アルゴリズムのハイパーパラメータ	
apply_recency_bias	モデルがアイテムインタラクションデータセット内の最新のアイテムインタラクションデータにより多くの重みを与えるかどうかを決定します。最新のインタラクションデータには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる場合があります。 最近のイベントにより多くの重みを置くモデルをトレーニングするには、apply_recency_bias を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、apply_recency_bias を false に設定します。 デフォルト値: true 範囲: true または false 値の型: ブール値 HPO 調整可能: いいえ

Personalized-Ranking レシピ

Personalized-Ranking レシピは、アイテムのパーソナライズされたランキングを生成します。パーソナライズされたランキングは、特定のユーザー向けに再ランク付けされた推奨アイテムのリストです。これは、検索結果、プロモーション、厳選されたリストなど、順序付けされたアイテムのコレクションがあり、ユーザーごとにパーソナライズされた再ランク付けを提供する場合に有益です。例えば、Personalized-Ranking を使用すると、Amazon Personalize は で生成した検索結果を再ランク付けできますOpenSearch。

モデルをトレーニングするために、Personalized-Ranking レシピは、アイテムインタラクションデータセットのデータを使用し、ユーザーがそれらを作成した場合は、データセットグループのアイテムデータセットと Users データセットを使用します (これらのデータセットはオプションです)。Personalized-Ranking を使用すると、Items データセットに 非構造化テキストメタデータ を含めることができ、アイテムインタラクションデータセットに コンテキストメタデータ を含めることができます。パーソナライズされたランキングを取得するには、GetPersonalizedRanking API を使用します。

ソリューションバージョンを作成した後は、必ずソリューションバージョンとデータを最新の状態に保ってください。Personalized-Ranking では、Amazon Personalize の新しいソリューションバージョンを手動で作成 (モデルを再トレーニング) して、レコメンデーション用に新しいアイテムを検討し、ユーザーの最新の動作でモデルを更新する必要があります。次に、ソリューションバージョンを使用してキャンペーンを更新する必要があります。詳細については、「レコメンデーションの関連性の維持」を参照してください。

> **Note**
>
> ランク付けのためにインタラクションデータなしでアイテムを提供すると、Amazon Personalize は GetPersonalizedRanking API レスポンスでレコメンデーションスコアなしでこれらのアイテムを返します。

このレシピには以下のプロパティがあります。

- 名前 – `aws-personalized-ranking`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-personalized-ranking`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-personalized-ranking`

- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/JSON-percentile-filtering`
- レシピタイプ – `PERSONALIZED_RANKING`

ハイパーパラメータ

以下の表では、Personalize-Ranking レシピのハイパーパラメータについて説明します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータがハイパーパラメータ最適化 (HPO) に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	
`hidden_dimension`	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。アイテムインタラクションデータセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、データセットが大きくなり、処理時間が長くなります。最適な値を決定するには、HPO を使用します。HPO を使用するには、CreateSolution オペレーション CreateSolutionVersion とオペレーションを呼び出すときに `performHPO` を `true` に設定します。 デフォルト値: 149

名前	説明
	範囲: [32, 256] 値の型: 整数 HPO 調整可能: はい
bptt	通し時間のバックプロパゲーションの手法を使用するかどうかを決定します。通し時間のバックプロパゲーションは、再帰的なニューラルネットワークベースのアルゴリズムの重みを更新する手法です。遅延報酬を早期イベントに接続するには、長期クレジットに bptt を使用します。例えば、遅延報酬には、数回のクリック後に行われる購入を指定できます。早期イベントは、最初のクリックにすることができます。クリックなどの同じイベントタイプ内であっても、長期的な効果を考慮し、合計報酬を最大化することをお勧めします。長期的な効果を考慮するには、より大きい bptt 値を使用します。大きな bptt 値を使用するには、より大きいデータセットと長い処理時間が必要です。 デフォルト値: 32 範囲: [2, 32] 値の型: 整数 HPO 調整可能: はい

名前	説明
recency_mask	モデルがアイテムインタラクションデータセットの最新の人気傾向を考慮する必要があるかどうかを決定します。最新の人気トレンドには、インタラクションイベントの基盤となるパターンの突然の変化が含まれる可能性があります。最近のイベントにより多くの重みを置くモデルをトレーニングするには、recency_mask を true に設定します。過去のすべてのインタラクションを均等に重み付けするモデルをトレーニングするには、recency_mask を false に設定します。同じ重みを使用して適切なレコメンデーションを取得するには、より大きなトレーニングデータセットが必要になる場合があります。 デフォルト値: True 範囲: True または False 値の型: ブール値 HPO 調整可能: はい
特徴化のハイパーパラメータ	

名前	説明
min_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最小パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。履歴の長さが短いある割合のユーザーを除外するには、min_user_history_length_percentile を使用します。履歴が短いユーザーは、多くの場合、ユーザーの個人的なニーズや希望ではなく、アイテムの人気に基づくパターンを表示します。それらを削除すると、データの基盤となるパターンに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.0 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最大パーセンタイル。履歴の長さ は、ユーザーに関するデータの合計量です。max_user_history_length_percentile を使用して、ある割合の履歴の長さが長いユーザーを除外します。これらのユーザーのデータにはノイズが含まれる傾向があるためです。例えば、ロボットに自動化されたインタラクションの長いリストがあるとします。これらのユーザーを削除することで、トレーニングのノイズが制限されます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min__user_history_length_percentile to 0.05 と max_user_history_length_percentile to 0.95 を設定すると、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.99 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

Personalized-Ranking サンプルノートブック

Personalized-Ranking レシピの使用方法を示すサンプル Jupyter ノートブックについては、「パーソナライズされたランキングの例」を参照してください。

RELATED_ITEMS

> **Note**
>
> RELATED_ITEMS レシピはすべてインタラクションデータを使用します。商品のメタデータも用意していて、Amazon Personalize が類似商品の検索に使用できるようにしたい場合は、Similar-Items レシピを選択してください。または、モデルにさらにハイパーパラメータを設定したい場合は SIMS レシピを選択してください。

RELATED_ITEMS レシピは、レコメンデーションを取得するときに指定したアイテムに類似するアイテムを返します。RELATED_ITEMS レシピは次のとおりです。

- Similar-Items レシピ
- SIMS の recipe

Similar-Items

Similar-Items レシピは、指定したアイテムに類似するアイテムのレコメンデーションを生成します。インタラクションデータとアイテムメタデータの両方に基づいて類似性は、インタラクションデータとアイテムメタデータの両方に基づいて計算されます。Amazon Personalize がレコメンデーションリクエストで指定したアイテム ID を見つけられない場合、レシピは人気のあるアイテムをレコメンデーションとして返します。Similar-Items は、レコメンデーションを生成する際にユーザーデータセットのデータを使用しません。ただし、ユーザーデータセットのデータに基づいてレコメンデーションをフィルタリングすることはできます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

SIMS

アイテム間の類似度 (SIMS) レシピは、アイテムインタラクションデータセットのユーザー履歴内のアイテムの共起に基づいて、特定のアイテムに類似したアイテムを生成します。アイテムに十分なユーザー行動データがない場合、または指定されたアイテム ID が見つからない場合、レシピは人気アイテムをレコメンデーションとして返します。

Similar-Items レシピ

> **Note**
>
> RELATED_ITEMS レシピはすべてインタラクションデータを使用します。商品のメタデータもあり、Amazon Personalize で類似商品を検索できるようにしたい場合は、[Similar-Items] を選択してください。または、モデルにさらにハイパーパラメータを設定したい場合は SIMS の recipe を選択してください。

Similar-Items (aws-similar-items) レシピは、指定したアイテムに類似するアイテムのレコメンデーションを生成します。Similar-Items を使用すると、顧客が以前の行動や商品メタデータに基づいてカタログ内の新しい商品を見つけやすくなります。類似アイテムを推奨することで、アプリケーションのユーザーエンゲージメント、クリックスルー率、およびコンバージョン率を高めることができます。

Similar-Items は、インタラクションデータとユーザーが提供するアイテムメタデータに基づいて類似度を計算します。インタラクションデータセットのユーザー履歴でアイテムの共起と、アイテムメタデータの類似性が考慮されます。例えば、Similar-Items を使用すると、Amazon Personalize は、顧客が類似のスタイル (カテゴリ別メタデータ) で頻繁に併せて購入するアイテムや、別のユーザーも視聴し、かつ、説明が類似する映画をレコメンドできます。

Similar-Items を使用して、GetRecommendations 操作 (または Amazon Personalize コンソール) でアイテム ID を指定すると、Amazon Personalize は類似アイテムのリストを返します。または、バッチワークフローを使用して、在庫内のすべてのアイテムについて類似アイテムを取得できます (「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照)。類似のアイテムが見つかったら、リクエストで指定したアイテムの属性に基づいてアイテムをフィルタリングできます。そのためには、`CurrentItem.attribute` 要素をフィルターに追加します。例については、item data filter examplesを参照してください。

Similar-Items を使用するには、少なくとも 1000 の一意の履歴インタラクションとイベントインタラクション (組み合わせ) を含むアイテムインタラクションデータセットを作成する必要があります。より正確な予測を行うには、アイテムデータセットを作成し、カタログ内のアイテムに関するメタデータをインポートすることもお勧めします。Similar-Items は、レコメンデーションを生成する際にユーザーデータセットのデータを使用しません。ユーザーデータセットのデータに基づいてレコメンデーションをフィルタリングすることはできます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

テキストデータとアイテムタイトルデータを含むアイテムデータセットがある場合は、関連するアイテムのテーマをバッチレコメンデーションで生成できます。詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

コールドアイテム (インタラクションが 5 個未満のアイテム) に類似したアイテムについてのレコメンデーションを取得できます。Amazon Personalize がレコメンデーションリクエストまたはバッチ入力ファイルで指定したアイテム ID を見つけられない場合、レシピは人気のあるアイテムをレコメンデーションとして返します。

ソリューションバージョンを作成した後は、必ずソリューションバージョンとデータを最新の状態に保ってください。Similar-Items では、Amazon Personalize の新しいソリューションバージョンを手動で作成 (モデルを再トレーニング) して、レコメンデーション用に新しいアイテムを検討し、ユーザーの最新の動作でモデルを更新する必要があります。次に、ソリューションバージョンを使用してキャンペーンを更新する必要があります。詳細については、「レコメンデーションの関連性の維持」を参照してください。

プロパティおよびハイパーパラメータ

Similar-Items レシピには、次のプロパティがあります。

- 名前 – `aws-similar-items`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-similar-items`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-similar-items`

詳細については、「レシピの選択」を参照してください。

以下の表では、Similar-Items レシピのハイパーパラメータを示しています。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータが HPO に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	
popularity_discount_factor	人気がレコメンデーションにどのように影響するかを設定します。人気の高い商品を多く含めるには、0 に近い値を指定します。1 に近い値を指定すると、人気度があまり重視されなくなります。 デフォルト値: 0.0 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ
item_id_hidden_dim	インタラクションデータに基づいてアイテム ID の埋め込みをモデル化するために Amazon Personalize が使用する隠れた変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。item_id_hidden_dim を使用するには、HPO を使用し、最小範囲と最大範囲の値を指定する必要があります。Amazon Personalize は、HPO を使用して、指定した範囲内で最適な値を見つけます。アイテムインタラクションデータセットが大きい場合は、より大きな最大値を指定します。最大値を大きくすると、処理に時間がかかります。 HPO を使用するには、CreateSolution 操作を呼び出すときにに performHPO および true を設定します。 デフォルト値: 100 範囲: [30, 200] 値の型: 整数 HPO 調整可能: はい

名前	説明
item_metadata_hidden_dim	Amazon Personalize がアイテムメタデータをモデル化するために使用する隠れた変数の数。item_metadata_hidden_dim を使用するには、HPO を使用し、最小範囲と最大範囲の値を指定する必要があります。Amazon Personalize は、HPO を使用して、指定した範囲内で最適な値を見つけます。アイテムインタラクションデータセットが大きい場合は、より大きな最大値を指定します。最大値を大きくすると、処理に時間がかかります。 HPO を使用するには、CreateSolution 操作を呼び出すときにに performHPO および true を設定します。 デフォルト値: 100 範囲: [30, 200] 値の型: 整数 HPO 調整可能: はい

SIMS の recipe

> **Note**
>
> RELATED_ITEMS レシピはすべてインタラクションデータを使用します。モデルにさらに多くのハイパーパラメータを設定したい場合は、SIMS を選択してください。商品のメタデータがあり、Amazon Personalize に類似商品の検索に使用させたい場合は、を選択してください。Similar-Items レシピ

I tem-to-item 類似度 (SIMS) レシピは、共同フィルタリングを使用して、レコメンデーションを取得するときに指定したアイテムに最も類似したアイテムをレコメンデーションします。SIMS は、色や料金などのアイテムメタデータではなく、アイテムインタラクションデータセットを使用して類似性を判断します。SIMS は、Interactions データセットのユーザー履歴でアイテムの共起を識別して、

類似アイテムを推奨します。例えば、SIMS を使用すると、Amazon Personalize は、顧客が頻繁に併せて購入するコーヒーショップの商品や、別のユーザーも視聴した映画を推奨できます。

類似商品のレコメンデーションを受け取ったら、リクエストで指定した商品の属性に基づいて商品をフィルタリングできます。そのためには、`CurrentItem.attribute` 要素をフィルターに追加します。例については、item data filter examplesを参照してください。

SIMS を使用するには、履歴とイベントのユニークなインタラクションを (合わせて) 1000 件以上含むアイテムインタラクションデータセットを作成する必要があります。SIMS は、レコメンデーションを生成する際に Users または Items のデータセットのデータを使用しません。これらのデータセットのデータに基づいてレコメンデーションをフィルタリングすることはできます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

アイテムに関する十分なユーザー行動データがない場合、または指定されたアイテム ID が見つからない場合、SIMS は人気のあるアイテムを推奨します。ソリューションバージョンを作成した後は、必ずソリューションバージョンとデータを最新の状態に保ってください。SIMS では、Amazon Personalize の新しいソリューションバージョンを手動で作成 (モデルを再トレーニング) して、レコメンデーション用に新しいアイテムを検討し、ユーザーの最新の動作でモデルを更新する必要があります。次に、ソリューションバージョンを使用してキャンペーンを更新する必要があります。詳細については、「レコメンデーションの関連性の維持」を参照してください。

SIMS レシピには以下のプロパティがあります。

- 名前 – `aws-sims`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-sims`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-sims`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/sims`
- レシピタイプ – `RELATED_ITEMS`

以下の表では、SIMS レシピのハイパーパラメータについて説明します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。ハイパーパラメータに最適な値を選択するプロセスは、ハイパーパラメータの最適化 (HPO) と呼ばれます。詳細については、「ハイパーパラメータおよび HPO」を参照してください。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータがハイパーパラメータ最適化 (HPO) に参加できますか?

名前	説明
アルゴリズムのハイパーパラメータ	
popularity_discount_factor	人気がレコメンデーションにどのように影響するかを設定します。人気の高い商品を多く含めるには、0 に近い値を指定します。1 に近い値を指定すると、人気度があまり重視されなくなります。 デフォルト値: 0.5 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: はい
min_cointeraction_count	アイテムのペア間の類似度を計算するために必要な同時インタラクションの最小数。例えば、値が 3 の場合、アルゴリズムで類似度を計算するためには、両方のアイテムとインタラクションを行った 3 人以上のユーザーが必要であることを意味します。 デフォルト値： 3 範囲: [0, 10] 値の型: 整数 HPO 調整可能: はい
特徴化のハイパーパラメータ	
min_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最小パーセンタイル。履歴の長さは、ユーザーに関し

名前	説明
	て利用可能なデータの合計量です。履歴の長さが短いある割合のユーザーを除外するには、min_user_history_length_percentile を使用します。履歴が短いユーザーは、多くの場合、ユーザーの個人的なニーズや希望ではなく、アイテムの人気に基づくパターンを表示します。それらを削除すると、データの基盤となるパターンに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 デフォルト値: 0.005 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_user_history_length_percentile	モデルのトレーニングに含めるユーザー履歴の長さの最大パーセンタイル。履歴の長さは、ユーザーに関して利用可能なデータの合計量です。履歴の長さが長いある割合のユーザーを除外するには、max_user_history_length_percentile を使用します。長い歴史を持つユーザーは、ノイズを含む傾向があります。例えば、ロボットに自動化されたインタラクションの長いリストがあるとします。これらのユーザーを削除することで、トレーニングのノイズが制限されます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のユーザーを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min_hist_length_percentile = 0.05 と max_hist_length_percentile = 0.95 には、履歴の長さが下位または上位 5% であるユーザーを除くすべてのユーザーが含まれます。 デフォルト値: 0.995 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
min_item_interaction_count_percentile	モデルのトレーニングに含めるアイテムのやり取りの最小パーセンタイル数。インタラクション履歴が短いアイテムの割合を除外するには、min_item_interaction_count_percentile を使用します。履歴が短いアイテムは、多くの場合、新しいアイテムです。それらを削除すると、既知の履歴を持つアイテムに重点を置いてモデルをトレーニングできます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のアイテムを保持し、エッジケースを削除する値を設定することをお勧めします。 デフォルト値: 0.01 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

名前	説明
max_item_interaction_count_percentile	モデルのトレーニングに含めるアイテムのやり取りの最大パーセンタイル数。インタラクション履歴が長いアイテムの割合を除外するには、max_item_interaction_count_percentile を使用します。履歴が長いアイテムは古い傾向があり、最新のものではなくなっている可能性があります。例えば、絶版になっている映画のリリースがあります。これらのアイテムを削除すると、より関連性の高いアイテムに重点を置くことができます。ヒストグラムまたは同様のツールを使用して、ユーザー履歴の長さを確認した後で適切な値を選択します。大部分のアイテムを保持し、エッジケースを削除する値を設定することをお勧めします。 例えば、min_item_interaction_count_percentile = 0.05 と max_item_interaction_count_percentile = 0.95 には、インタラクションカウントが下位または上位 5% であるアイテムを除くすべてのアイテムが含まれます。 デフォルト値: 0.9 範囲: [0.0, 1.0] 値の型: 浮動小数点 HPO 調整可能: いいえ

SIMS サンプルノートブック

SIMS レシピの使用方法を示す Jupyter ノートブックのサンプルについては、「Finding similar items + HPO」を参照してください。

PERSONALIZED_ACTIONS

ロイヤルティプログラムへの登録、アプリのダウンロード、クレジットカードの申請など、次に最適なアクションをユーザーにリアルタイムでレコメンドするには、PERSONALIZED_ACTIONS レシピを使用してください。次に最適なアクションをレコメンドすることにより、顧客ロイヤルティを高め、収益を増やし、ユーザーエクスペリエンスを向上させることができます。

PERSONALIZED_ACTIONS レシピは次のとおりです。

- Next-Best-Action レシピ

Next-Best-Action レシピ

Next-Best-Action (aws-next-best-action) レシピは、ユーザーにとって次善のアクションを提案するリアルタイムのレコメンデーションを生成します。ユーザーにとって次善のアクションは、ユーザーが実行する可能性が最も高いアクションです。例えば、ロイヤルティプログラムへの登録、アプリのダウンロード、クレジットカードの申請などです。

Next-Best-Action を使用すると、ユーザーがアプリケーションを使用する際に、パーソナライズされたアクションのレコメンデーションを提供できます。ユーザーに適切なアクションを提案することにより、より多くのユーザーがアクションを実行できるようになります。推奨するアクションによっては、顧客の ロイヤルティを高め、収益を増やし、アプリケーションのユーザーエクスペリエンスを向上させることができます。パーソナライズしたアクションのレコメンデーションが e コマースアプリケーションでどのように役立つかを説明するユースケース例については、「ユースケースの例」を参照してください。

Amazon Personalize は、Actions データセットにインポートしたアクションから次善のアクションを予測します。アクションやアイテムとのやり取りに基づいて、ユーザーが実行する可能性が最も高いアクションを特定します。アクションデータにアクションの値が含まれている場合、Amazon Personalize はアクションの値を説明します。ユーザーが 2 つの異なるアクションを実行する可能性が同程度である場合、Amazon Personalize は値の大きい方のアクションを高くランク付けします。

ユーザーに対するリアルタイムのアクションレコメンデーションを取得すると、Amazon Personalize は、設定可能な期間 (`action optimization period`) 内にユーザーが実行する可能性が最も高いアクションのリストを返します。例えば、今後 14 日間にユーザーが実行する可能性が最も高いアクションなどです。リストは傾向スコアの降順でソートされます。このスコアは、ユーザーがアクションを実行する可能性を表します。

アクションインタラクションデータをインポートするまで、Amazon Personalize はパーソナライズなしでアクションを推奨し、傾向スコアは 0.0 です。アクションのスコアは、アクションが以下の値になった後に付けられます。

- TAKEN イベントタイプのアクションインタラクションが 50 回以上。
- NOT_TAKEN または VIEWED イベントタイプのアクションインタラクションが 50 回以上。

これらのアクションインタラクションは、最新のソリューションバージョントレーニング時に存在し、アクションインタラクションデータセットの最新のインタラクションタイムスタンプから 6 週間以内に発生している必要があります。

Next-Best-Action レシピが使用するデータの詳細については、「必須およびオプションのデータセット」を参照してください。

Next-Best-Action レシピを使用してソリューションを作成する場合、`action optimization period` 特微量化ハイパーパラメータを使用して Amazon Personalize がアクションを予測するときに使用する時間枠を設定できます。詳細については、「プロパティおよびハイパーパラメータ」を参照してください。

トピック

- ユースケースの例
- レシピ機能
- 必須およびオプションのデータセット
- プロパティおよびハイパーパラメータ

ユースケースの例

ユーザーに適切なアクションを提案することにより、より多くのユーザーがアクションを実行できるようになります。推奨するアクションによっては、顧客の口イヤルティを高め、収益を増やし、アプリケーションのユーザーエクスペリエンスを向上させることができる可能性があります。

例えば、次のさまざまなアクションを提案する e コマースアプリケーションがあるとします。

- ロイヤルティプログラムのサブスクライブ
- モバイルアプリのダウンロード
- ジュエリーカテゴリで購入
- ビューティーおよびグルーミングカテゴリで購入

あなたのサイトで頻繁に買い物していて、ジュエリー、ビューティーおよびグルーミングの購入アクションを繰り返し行っているユーザーがいるかもしれません。このユーザーの場合、Amazon Personalize のアクションのレコメンデーションとそのスコアには以下が含まれる可能性があります。

- ロイヤルティプログラムのサブスクライブ

  傾向スコア - 1.00
- ジュエリーカテゴリで購入

  傾向スコア - 0.86
- ビューティーおよびグルーミングカテゴリで購入

  傾向スコア - 0.85

これらのアクションのレコメンデーションがあれば、ロイヤルティプログラムの登録をユーザーに促すことができます。このアクションは傾向スコアが最も高く、ユーザーが実行する可能性が最も高いアクションです。これは、ユーザーが頻繁にあなたの店舗で買い物し、ロイヤルティプログラムの特典を利用する可能性が高いためです。

レシピ機能

Next-Best-Action レシピでは、アクションレコメンデーションを生成する際に以下の Amazon Personalize レシピ機能を使用します。

- リアルタイムのパーソナライゼーション: Amazon Personalize はリアルタイムのパーソナライゼーションを使用して、ユーザーの関心の高まりに応じてアクションのレコメンデーションを更新および調整します。詳細については、「リアルタイムパーソナライゼーション」を参照してください。
- 探索: 探索では、レコメンデーションには新しいアクションやインタラクションデータの少ないアクションが含まれます。探索の詳細については、「探査」を参照してください。
- 自動更新: Amazon Personalize は自動更新を使用して、2 時間ごとに最新モデル (ソリューションバージョン) を自動的に更新し、探索を介し新しいアクションをレコメンデーションに含めます。詳細については、「自動更新」を参照してください。

必須およびオプションのデータセット

Next-Best-Action レシピを使用するには、以下のデータセットを作成する必要があります。

- Actions: アクションに関するデータ (値など) を Amazon Personalize Actions データセットにインポートします。

  アクションデータでは、アクションごとに EXPIRATION_TIMESTAMP を指定できます。アクションの有効期限が切れている場合、Amazon Personalize はそのアクションをレコメンデーションに含めません。アクションごとに REPEAT_FREQUENCY を指定することもできます。これは、ユーザーがアクションを操作した後、Amazon Personalize がアクションを再度推奨するまでの待ち時間を示しています。Actions データセットに保存できるデータについては、「Actions データセット」を参照してください。
- アイテムインタラクション:アイテムインタラクションデータセットには、少なくとも 1000 件のアイテムインタラクションが必要です。Amazon Personalize はアイテムインタラクションを使用して、ユーザーの現在の状態と関心を把握します。アイテムインタラクションデータの詳細については、「アイテムインタラクションデータセット」を参照してください。

以下のデータセットはオプションです。

- アクションインタラクションデータセット: アクションインタラクションは Actions データセット内のユーザーとアクションが関与するインタラクションです。実行済み、未実行、表示済みのアクションインタラクションをインポートできます。このデータはオプションですが、高品質のレコメンデーションをするためにアクションインタラクションデータをインポートすることをお勧めします。アクションインタラクションデータがない場合は、PutActionInteractions API オペレーションを使用して空のアクションインタラクションデータセットを作成し、顧客のアクションとのインタラクションを記録できます。

  アクションインタラクションデータをインポートするまで、Amazon Personalize はパーソナライズなしでアクションを推奨し、傾向スコアは 0.0 です。アクションのスコアは、アクションが以下の値になった後に付けられます。

  - TAKEN イベントタイプのアクションインタラクションが 50 回以上。
  - NOT_TAKEN または VIEWED イベントタイプのアクションインタラクションが 50 回以上。

  これらのアクションインタラクションは、最新のソリューションバージョントレーニング時に存在し、アクションインタラクションデータセットの最新のインタラクションタイムスタンプから 6 週間以内に発生している必要があります。

  インポートできるアクションインタラクションデータについては、「Action インタラクションデータセット」を参照してください。アクションインタラクションイベントの記録については、「アクションインタラクションイベントの記録」を参照してください。

> **Note**
>
> Next-Best-Action では、Amazon Personalize は Action インタラクションデータセットのインプレッションデータやコンテキストメタデータを使用しません。

- Users: Amazon Personalize は、ユーザーとその関心についてよりよく理解するために、Users データセット内のあらゆるデータを使用します。Users データセットのデータを使用して、アクションのレコメンデーションをフィルタリングすることもできます。インポートできるユーザーデータについては、「ユーザーデータセット」を参照してください。
- Items: Amazon Personalize は、アイテムデータセット内のデータをアイテムインタラクションデータセットと共に使用して、アイテムの動作の関連性とパターンを識別します。これにより、Amazon Personalize はユーザーとその関心について理解しやすくなります。インポートできるアイテムデータについては、「製品データセット」を参照してください。

プロパティおよびハイパーパラメータ

Next-Best-Action レシピはハイパーパラメータ最適化をサポートしていません。Next-Best-Action レシピには、次のプロパティがあります。

- 名前 – `aws-next-best-action`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-next-best-action`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-next-best-action`

次の表では、レシピの機能化ハイパーパラメータについて説明しています。 aws-next-best-action ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。特徴化のハイパーパラメータは、トレーニングで使用するデータのフィルタリング方法を制御します。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)
- HPO 調整可能: パラメータが HPO に参加できるかどうか

名前	説明
特徴化のハイパーパラメータ	
action_optimization_period	Amazon Personalize がユーザーの次善のアクションを予測する際に使用する時間帯。例えば、今後 14 日間にユーザーが実行する可能性が最も高いアクションなどです。 アクションインタラクションデータがあまりない場合は、大きい値を指定します。どの値を指定すればよいかわからない場合は、デフォルトを使用してください。 デフォルト値: 14 範囲: [7, 28] 値の型: 整数 HPO 調整可能: いいえ

USER_SEGMENTATION

USER_SEGMENTATION レシピは、アイテム入力データに基づいてユーザーのセグメントを生成します。各ユーザーセグメントは、各ユーザーがインベントリ内のアイテムを操作する蓋然性に基づいて、降順にソートされます。USER_SEGMENTATION レシピを使用して、アイテムまたはアイテム属性の詳細設定に基づいてカタログを操作する可能性が最も高いユーザーのセグメントを作成します。例えば、特定の映画を視聴したり、ブランドごとに特定の製品を購入したりする可能性が最も高いユーザーをターゲットとしたマーケティングキャンペーンを作成したい場合があります。

Item-Affinity

Item-Affinity (aws-item-affinity) レシピは、指定した各アイテムのユーザーセグメントを作成する USER_SEGMENTATION レシピです。

モデルをトレーニングするために、Item-Affinity レシピは、データセットグループ内の Interactions および Items データセットを使用します。ユーザーセグメントを作成するには、Item-Affinity レシピ

を使用してソリューションバージョンをトレーニングしてから、バッチセグメントジョブを作成します。

Item-Attribute-Affinity

Item-Attribute-Affinity (aws-item-attribute-affinity) レシピは、指定した各アイテム属性のユーザーセグメントを作成する USER_SEGMENTATION レシピです。

モデルをトレーニングするために、Item-Attribute-Affinity レシピは、データセットグループの Interactions データセットと Items データセットを使用します。ユーザーセグメントを作成するには、Item-Attribute-Affinity レシピを使用してソリューションバージョンをトレーニングしてから、バッチセグメントジョブを作成します。

Item-Affinity レシピ

Item-Affinity (aws-item-affinity) レシピは、指定した各アイテムのユーザーセグメント (ユーザーグループ) を作成する USER_SEGMENTATION レシピです。これらは、Amazon Personalize が各アイテムとやり取りする可能性が最も高いと予測しているユーザーです。Item-Affinity を使用して、ユーザーの詳細を確認し、それぞれのユーザーセグメントに基づいてアクションを実行します。

例えば、カタログ内のアイテムに関するユーザーの嗜好に基づいて、小売アプリケーションのマーケティングキャンペーンを作成したい場合があります。Item-Affinity は、Interactions と Items のデータセットのデータに基づいて、各アイテムについてユーザーセグメントを作成します。これを使用して、アクション (アイテムのクリックやアイテムの購入など) を実行する可能性に基づいて、さまざまなアイテムをさまざまなユーザーセグメントにプロモーションできます。さまざまなユーザーの組み合わせに対する製品のクロスセリングや、採用可能性のある求人応募者の特定なども、他の用途の例として挙げることができます。

アイテムに基づいてユーザーセグメントを取得するには、Item-Affinity レシピを使用してソリューションとソリューションバージョンを作成し、JSON 形式のアイテムのリストを Amazon S3 バケットに追加して、バッチセグメントジョブを作成します。Amazon Personalize は、各アイテムのユーザーセグメントを Amazon S3 の出力場所に出力します。入力データに対して、ユーザーセグメントを取得する最大 500 アイテムを設定できます。バッチセグメントジョブの入力データの準備については、「バッチレコメンデーション用の入力データを準備します。」を参照してください。

Item-Affinity を使用するには、アイテムインタラクションデータセットが必要です。アイテムデータセットとユーザーデータセットはオプションです。バッチセグメントジョブでユーザーセグメントを取得できます。詳細については、「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

ソリューションバージョンを作成した後は、必ずソリューションバージョンとデータを最新の状態に保ってください。Item-Affinity では、Amazon Personalize の新しいソリューションバージョンを作成して、ユーザーセグメントの新規ユーザーを考慮し、ユーザーの最新の行動に合わせてモデルを更新する必要があります。アイテムのユーザーセグメントを取得するには、ソリューションバージョンを作成したときにそのアイテムが存在している必要があります。

Item-Affinity レシピには、次のプロパティがあります。

- 名前 – `aws-item-affinity`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-item-affinity`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-item-affinity`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/item-affinity`
- レシピタイプ – `USER_SEGMENTATION`

以下の表では、Item-Affinity レシピのハイパーパラメータを示しています。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整するアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。Item-Affinity レシピでハイパーパラメータ最適化 (HPO) を使用することはできません。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)

名前	説明
アルゴリズムのハイパーパラメータ	
`hidden_dimension`	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。インタラクションデータセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、

名前	説明
	データセットが大きくなり、処理時間が長くなります。 デフォルト値: 149 範囲: [32, 256] 値の型: 整数

Item-Attribute-Affinity レシピ

Item-Attribute-Affinity (aws-item-attribute-affinity) レシピは、指定した各アイテム属性のユーザーセグメント (ユーザーグループ) を作成する USER_SEGMENTATION レシピです。これらは、Amazon Personalize が特定の属性を持つ商品を操作する可能性が最も高いと予測しているユーザーです。Item-Attribute-Affinity を使用して、ユーザーの詳細を確認し、それぞれのユーザーセグメントに基づいてアクションを実行します。

例えば、カタログ内の靴の種類に関するユーザーの嗜好に基づいて、小売アプリケーションのマーケティングキャンペーンを作成したい場合があります。Item-Attribute-Affinity は、Interactions と Items のデータセットに含まれる靴の種類に基づく各データについてユーザーセグメントを作成します。これを使用して、アクション (靴のクリックや靴の購入など) を実行する可能性に基づいて、さまざまな靴をさまざまなユーザーセグメントにプロモーションできます。他の用途には、さまざまな映画ジャンルをさまざまなユーザーに宣伝したり、職種に基づいて採用可能性のある求人応募者を特定することなども、他の用途の例として挙げることができます。

アイテム属性に基づいてユーザーセグメントを取得するには、Item-Attribute-Affinity レシピを使用してソリューションとソリューションバージョンを作成し、JSON 形式のアイテム属性のリストを Amazon S3 バケットに追加して、バッチセグメントジョブを作成します。Amazon Personalize は、各アイテムのユーザーセグメントを Amazon S3 の出力場所に出力します。入力データには最大 10 個のクエリを含めることができ、各クエリは 1 つ以上のアイテム属性です。バッチセグメントジョブの入力データの準備については、「バッチレコメンデーション用の入力データを準備します。」を参照してください。

Item-Attribute-Affinity を使用するには、アイテムインタラクションデータセットとアイテムデータセットが必要です。Items データセットには、テキストではなく、かつ、予約されていないメタデータ列である列が少なくとも 1 つ必要です。バッチセグメントジョブでユーザーセグメントを取得で

きます。詳細については、「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

ソリューションバージョンを作成した後は、必ずソリューションバージョンとデータを最新の状態に保ってください。Item-Attribute-Affinity では、Amazon Personalize の新しいソリューションバージョンを作成して、ユーザーセグメントの新規ユーザーを考慮し、ユーザーの最新の行動に合わせてモデルを更新する必要があります。アイテム属性のユーザーセグメントを取得するには、ソリューションバージョンを作成したときにアイテム属性が存在している必要があります。

Item-Attribute-Affinity レシピには、次のプロパティがあります。

- 名前 – `aws-item-attribute-affinity`
- レシピ Amazon リソースネーム (ARN) – `arn:aws:personalize:::recipe/aws-item-attribute-affinity`
- アルゴリズム ARN – `arn:aws:personalize:::algorithm/aws-item-attribute-affinity`
- 機能変換 ARN – `arn:aws:personalize:::feature-transformation/item-attribute-affinity`
- レシピタイプ – `USER_SEGMENTATION`

次の表で、Item-Attribute-Affinity レシピのハイパーパラメータを示します。ハイパーパラメータは、モデルパフォーマンスを向上させるために調整できるアルゴリズムパラメータです。アルゴリズムのハイパーパラメータは、モデルの実行方法を制御します。Item-Attribute-Affinity レシピでハイパーパラメータ最適化 (HPO) を使用することはできません。

このテーブルには、各ハイパーパラメータに関する以下の情報も含まれています。

- 範囲: [下限、上限]
- 値のタイプ: 整数、連続 (浮動小数点)、カテゴリ別 (ブール値、リスト、文字列)

名前	説明
アルゴリズムのハイパーパラメータ	
`hidden_dimension`	モデルで使用される非表示変数の数。非表示の変数は、ユーザーの購入履歴と商品統計を再作成して、ランキングスコアを生成します。インタラクション

名前	説明
	データセットにより複雑なパターンが含まれている場合は、より多くの非表示ディメンションを指定します。使用する非表示のディメンションが多くなると、データセットが大きくなり、処理時間が長くなります。 デフォルト値: 149 範囲: [32, 256] 値の型: 整数

準備チェックリスト

Amazon Personalize の仕組みを確認し、入門演習を完了したら、自身のデータで Amazon Personalize を使用する準備を始めることができます。このチェックリストには、Amazon Personalize の機能、要件、およびデータガイダンスのリストが記載されています。計画を立てるのに役立ちますし、Amazon Personalize でリソースを作成する際の参照として使用することもできます。

トピック

ユースケースを Amazon Personalize リソースと一致させましたか?

Amazon Personalize のレコメンデーションは、次のユースケースに対応できます。

- ユーザー向けにパーソナライズされたレコメンデーションの生成
- 類似商品や関連商品のレコメンド
- トレンド商品や人気商品のレコメンド
- ユーザーへの次善アクションのレコメンド
- 関連性による並べ替え (カスタムリソースのみ)
- ユーザーセグメントの生成 (カスタムリソースのみ)

Amazon Personalize は、これらのユースケースに合わせて設定されたドメインベースのリソースとカスタムリソースを備えています。ドメインデータセットグループまたはカスタムデータセットグループを作成します。

- ドメインデータセットグループでは、VIDEO_ON_DEMAND または ECOMMERCE ドメイン向けに事前設定され最適化されたリソースを作成します。

  ストリーミング動画または eコマースアプリケーションがある場合は、ドメインデータセットグループから開始することをおすすめします。カスタムユースケース向けにトレーニングされたソリューションやソリューションバージョンなどのカスタムリソースを追加できます。また、カスタムリソースを使用してバッチレコメンデーションを取得することもできます。ドメインデータセットグループには、アクションやアクションインタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。

- カスタムデータセットグループでは、ユースケースに合ったレシピを選択します。設定可能なソリューションとソリューションバージョン (トレーニングされた Amazon Personalize のレコメンデーションモデル) をトレーニングし、デプロイします。準備ができたら、ソリューションバージョンをキャンペーンに展開して、リアルタイムのレコメンデーションを行うことができます。または、キャンペーンなしで一括レコメンデーションを取得することもできます。

  ストリーミング動画または eコマースアプリケーションがない場合は、カスタムデータセットグループを作成します。それ以外の場合は、ドメインデータセットグループから始めて、必要に応じてカスタムリソースを追加してください。

Amazon Personalize で使用できるユースケースとカスタムレシピについては、「ドメインのユースケースとカスタムレシピ」を参照してください。

アイテムインタラクションデータは十分ですか?

すべてのユースケースとレシピにおいて、25 人のユニークユーザーに対し、それぞれ 2 回以上のインタラクションがあり、1,000 回以上のアイテムインタラクションが必要です。質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

十分なデータがあるかどうかわからない場合は、Amazon Personalize コンソールでデータをインポートして分析できます。詳細については、「データセット内のデータの品質と量を分析する」を参照してください。

リアルタイムのイベントストリーミングアーキテクチャは導入されていますか?

アイテムインタラクションデータが足りない場合は、Amazon Personalize を使用して追加のリアルタイムイベントデータを収集できます。いくつかのレシピとユースケースを使用することで、Amazon Personalize はユーザーの最新のアクティビティから学習し、ユーザーがアプリケーションを使用するたびにレコメンデーションを更新することができます。

イベントがレコメンデーションに与える影響、サードパーティのイベント追跡サービスのリスト、実装例など、イベントの記録に関する情報については、「イベントの記録」を参照してください。

データは Amazon Personalize 用に最適化されていますか?

データ内の以下の点を確認することをお勧めします。

- 欠落している値を確認します。レコードの少なくとも 70% にすべての属性のデータを含めることをおすすめします。NULL 値を許容する列は 70% 以上記入しておくことをお勧めします。
- 命名規則の不一致、アイテムのカテゴリの重複、データセット間の ID の不一致、ID の重複など、データ内の不正確さや問題を修正します。これらの問題は、レコメンデーションに悪影響を及ぼしたり、予期しない動作につながる可能性があります。例えば、データには「N/A」と「該当なし」の両方が含まれていても、「N/A」のみに基づいてレコメンデーションを除外している場合があります。「該当なし」とマークされたアイテムはフィルターでは削除されません。
- 複数のジャンルの映画など、アイテム、ユーザー、またはアクションが複数のカテゴリを持つことができる場合は、カテゴリ値を 1 つの属性にまとめ、それぞれの値を | 演算子で区切ります。例えば、映画のジャンルデータは、アクション | アドベンチャー | スリラーなどです。
- 1 つの列に使用できるカテゴリの数が 1000 を超えることは避けてください (列にフィルタリングのみを目的としたデータが含まれている場合を除く)。

データに関する推奨事項の完全なリストと、Amazon Personalize を使用して問題を特定する方法については、「データセット内のデータの品質と量を分析する」を参照してください。

レコメンデーションを改善するためのオプションデータを収集していますか?

以下のデータは、レコメンデーションの関連性を高めるのに役立ちます。

- イベントタイプ (すべてのドメインデータセットグループのユースケースで必須)
- イベント値
- コンテキストメタデータ
- アイテムとユーザーのメタデータ
- アクションインタラクションデータ (PERSONALIZED_ACTIONS レシピでのみ使用)

Amazon Personalize が使用できるデータの種類についての詳細は、「Amazon Personalize が使用できるデータの種類」を参照してください。

レコメンデーションをテストする予定はありますか?

A/B テストを使用すると、さまざまなユーザーグループがさまざまなモデルのレコメンデーションとインタラクションを行った結果を比較できます。A/B テストは、さまざまなレコメンデーション戦略を比較し、レコメンデーションがビジネス目標の達成に役立っているかどうかを確認するのに役立ちます。詳細については、「A/B テストを使用したレコメンデーションの影響の測定」を参照してください。

他にビジネス目標はありますか?

場合によっては、ユーザーに関連するレコメンデーションの生成に加えて、目標を付与していることがあります。例えば、収益を最大化したり、特定のカテゴリの特定の種類のアイテムを宣伝する場合があります。次の Amazon Personalize 機能が役立ちます。

- プロモーション: プロモーションを利用して、特定の割合のアイテムがビジネス要件を満たしていることを確認できます。詳細については、「レコメンデーション内のアイテムのプロモーション」を参照してください。
- ビジネス目標に合わせた最適化: カスタムデータセットグループレシピの中には、ストリーミング時間の最大化や収益の増加など、カスタム目的に合わせてソリューションを最適化できるものもあります。詳細については、「追加の目的のためのソリューションの最適化」を参照してください。
- レコメンデーションのフィルタリング フィルターを使用してビジネスルールをレコメンデーションに適用します。フィルターを使用して、特定の種類のアイテムをレコメンデーションに含めたり、レコメンデーションに含めたり、レコメンデーションから除外したりできます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

Amazon Personalize のワークフロー

準備チェックリスト を確認したら、Amazon Personalize ワークフローを完了する準備が整いました。

1. データセットグループの作成

   データセットグループは、Amazon Personalize のリソースのコンテナです。作成するデータセットグループのタイプによって、Amazon Personalize ワークフローのステップ 3 で作成できるリソースが決まります。

   - ドメインデータセットグループを使用すると、VIDEO_ON_DEMAND または eコマースドメインのユースケースに合わせて設定されたレコメンダーを作成できます。レコメンダーを使用して、レコメンデーションを取得します。Amazon Personalize は設定、トレーニング、更新を管理します。ドメインデータセットグループから始める場合でも、カスタムリソースを追加できます。ドメインデータセットグループには、アクションデータセットやアクションインタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。
   - または、カスタムリソースを使用してカスタムデータセットグループを作成できます。これらには、ソリューション、ソリューションバージョン、キャンペーンが含まれます。これらのリソースでは、構成、更新、再トレーニングをより細かく制御できます。

2. データを準備してインポートする

   アイテムインタラクション、アクションインタラクション、アイテム、ユーザー、およびアクションのレコードをデータセットにインポートします (データ用の Amazon Personalize のコンテナ)。レコードは一括でインポートすることも、個別にインポートすることもできます。一括データをインポートする場合、Amazon SageMaker Data Wrangler を使用して 40 以上のソースからデータをインポートし、Amazon Personalize 用に準備できます。詳細については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。

   Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

3. ドメインレコメンダーまたはカスタムリソースを作成する

   データをインポートしたら、ドメインレコメンダー (ドメインデータセットグループ用) またはカスタムリソース (カスタムデータセットグループ用) を作成して、データに基づいてモデルをトレーニングします。これらのリソースを使用して、レコメンデーションを生成します。

4. レコメンデーションの取得

   レコメンダーやカスタムキャンペーンを使ってレコメンデーションを取得しましょう。カスタムデータセットグループでは、バッチレコメンデーションやユーザーセグメントも取得できます。

Amazon Personalize ワークフローを初めて完了した後は、データを最新の状態に保ち、カスタムソリューションを定期的に再トレーニングしてください。これにより、モデルはユーザーの最新のアクティビティから学習し、レコメンデーションの関連性を維持および改善できます。詳細については、「レコメンデーションの関連性の維持」を参照してください。

ステップ 1: データセットグループを作成する

Amazon Personalize の使用を開始する際に、データセットグループを作成します。データセットグループは、データセット、ドメインレコメンダー、カスタムリソースなど、Amazon Personalize リソース用のコンテナです。データセットグループは、リソースを独立したコレクションにまとめるため、あるデータセットグループのリソースが他のデータセットグループのリソースに影響を与えることはできません。

ビジネスドメインごとにデータセットグループを作成します。例えば、ストリーミング動画のレコメンデーションを提供するアプリケーションと、オーディオブックのレコメンデーションを提供する別のアプリケーションがあるとします。Amazon Personalize では、各アプリケーションに独自のデータセットグループを作成します。この方法では、あるアプリケーションのデータが、Amazon Personalize が別のアプリケーション用に生成するレコメンデーションには影響しません。

ドメインデータセットグループまたはカスタムデータセットグループを作成できます。

- ドメインデータセットグループを使用して、さまざまなユースケース向けに事前設定され、最適化されたリソースを作成できます。データセットグループを作成する際には、VIDEO_ON_DEMAND または ECOMMERCE のドメインを指定してドメインデータセットグループにします。

  ストリーミング動画または eコマースアプリケーションがある場合は、ドメインデータセットグループを作成することをおすすめします。カスタムユースケース向けにトレーニングされたソリューションやソリューションバージョンなどのカスタムリソースを追加できます。ドメインデータセットグループには、アクションデータセットやアクションインタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。
- カスタムデータセットグループには、ユースケースに応じて設定したカスタムリソースのみが含まれます。カスタムリソースを使用して、ビジネスニーズに基づき、設定可能なソリューションとソリューションバージョン (トレーニングされた Amazon Personalize のレコメンデーションモデル)

をトレーニングし、デプロイします。VIDEO_ON_DEMAND や ECOMMERCE アプリケーションがない場合は、カスタムデータセットグループを作成することをお勧めします。それ以外の場合は、ドメインデータセットグループから始めて、必要に応じてカスタムリソースを追加することをおすすめします。

Amazon Personalize コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDK を使用してデータセットグループを作成できます。

トピック

データセットグループの作成 (コンソール)

Amazon Personalize コンソールでデータセットグループ名を指定して、データセットグループを作成します。

データセットグループを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループの作成] を選択します。
3. Amazon Personalize を初めて使用する場合は、[Create dataset group] (データセットグループを作成) のページの [New dataset group] (新しいデータセットグループ) で、[Get started] (使用を開始) を選択します。
4. [データセットグループの詳細] で [Dataset group name (データセットグループ名)] にデータセットグループの名前を指定します。
5. [ドメイン] を選択します。
   - [Eコマース] を選択して ECOMMERCE ドメインデータセットグループを作成します。
   - VIDEO_ON_DEMAND ドメインデータセットグループを作成するには、[ビデオオンデマンド] を選択します。
   - ソリューション、キャンペーン、バッチ推論ジョブなどのカスタムリソースのみを含むカスタムデータセットグループを作成するには、[カスタム] を選択します。

6. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
7. [データセットグループの作成] を選択します。[概要] ページが表示されます。これで、データをインポートする準備ができました。ステップ 2: データの準備とインポート を参照してください。

データセットグループの作成 (AWS CLI)

その後、create-dataset-group 操作を使用してデータセットを作成します。ドメインに ECOMMERCE または VIDEO_ON_DEMAND を指定して、ドメインデータセットグループを作成します。ドメインを指定せずに、カスタムデータセットグループを作成します。Tags パラメータを使用して、オプションで Amazon Personalize のリソースにタグを付けることができます。サンプルについては、「タグの追加 (AWS CLI)」を参照してください。

次のコードは、VIDEO_ON_DEMAND ドメインのドメインデータセットグループを作成します。

```
aws personalize create-dataset-group \
--name dataset-group-name \
--domain VIDEO_ON_DEMAND
```

成功した場合、データセットグループの Amazon リソースネーム (ARN) は次のように表示されます。

```
{
  "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/
DatasetGroupName"
}
```

後で使用するためにこの値を記録します。作成したデータセットグループを表示するには、describe-dataset-group コマンドを使用して、返されたデータセットグループの ARN を指定します。

```
aws personalize describe-dataset-group \
--dataset-group-arn dataset group arn
```

データセットグループとそのプロパティは次のように表示されます。

```
{
```

```
    "datasetGroup": {
        "name": "DatasetGroupName",
        "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/
DatasetGroupName",
        "status": "ACTIVE",
        "creationDateTime": 1542392161.262,
        "lastUpdatedDateTime": 1542396513.377
    }
}
```

データセットグループの status が ACTIVE になると、データをインポートする準備が整います。詳細については、「ステップ 2: データの準備とインポート」を参照してください。

データセットグループ (AWS SDK) の作成

次のコードを使用して、ドメインデータセットグループを作成します。ドメインデータセットグループに名前を付け、domain には ECOMMERCE または VIDEO_ON_DEMAND を指定します。カスタムデータセットグループを作成するには、コードを変更してドメインパラメータを削除します。

API 操作の詳細については、API リファレンスのセクションの「CreateDatasetGroup」を参照してください。Tags パラメータを使用して、オプションで Amazon Personalize のリソースにタグを付けることができます。サンプルについては、「タグの追加 (AWS SDKs)」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_group(
  name = 'dataset group name',
  domain = 'business domain'
)
dsg_arn = response['datasetGroupArn']

description = personalize.describe_dataset_group(datasetGroupArn = dsg_arn)
['datasetGroup']

print('Name: ' + description['name'])
print('ARN: ' + description['datasetGroupArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

```
public static String createDomainDatasetGroup(PersonalizeClient personalizeClient,
                                               String datasetGroupName,
                                               String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
 CreateDatasetGroupRequest.builder()
                .name(datasetGroupName)
                .domain(domain)
                .build();
        return
 personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import {  CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: 'NAME',  /* required */
  domain: 'DOMAIN'   /* required for a domain dsg, specify ECOMMERCE or
 VIDEO_ON_DEMAND */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetGroupCommand(domainDatasetGroupParams));
    console.log("Success", response);
    return response; // For unit tests.
```

```
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

DescribeDatasetGroup 操作は、datasetGroupArn と操作のステータスを返します。データセットグループの status が ACTIVE になると、データをインポートする準備が整います。詳細については、「ステップ 2: データの準備とインポート」を参照してください。

ステップ 2: データの準備とインポート

Amazon Personalize は、データを使用して、ユーザーおよびユーザーセグメント向けのレコメンデーションを生成します。Amazon Personalize は、ユーザーがデータセットを削除するまで、データをデータセットに保存します。すべてのユースケース (ドメインデータセットグループ) とレシピ (カスタムリソース) で、インタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

データをインポートする場合、レコードを一括でインポートするか、増分的にインポートするか、その両方とするかを選択できます。

- 一括インポートでは、大量の履歴レコードを一度にインポートする必要があります。SageMaker Data Wrangler と複数のデータソースを使用して、アイテムインタラクション、ユーザー、アイテムのバルクデータを準備してインポートできます。または、ユーザーがバルクデータを準備し、Amazon S3 の CSV ファイルから Amazon Personalize に直接インポートすることもできます。Amazon Personalize用にバルクデータをフォーマットする方法については、「データ形式ガイドライン」を参照してください。

- 個別インポートでは、Amazon Personalize コンソールと API オペレーションを使用してレコードを個別にインポートします。または、ライブイベントからインタラクションデータをリアルタイムでインポートすることもできます。

Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

カタログが大きくなってきたら、一括または増分的なデータインポート操作で履歴データを更新するようにします。リアルタイムのレコメンデーションについては、アイテムインタラクションデータセットをユーザーの動作で最新の状態に保ちます。そのためには、イベントトラッカーと PutEvents オペレーションを使用してリアルタイムのインタラクションイベントを記録します。詳細については、イベントの記録 を参照してください。

トピック

- バルクデータの準備とインポート
- 個々のレコードのインポート

バルクデータの準備とインポート

データセットを作成すると、バルク履歴データの Amazon Personalize へのインポートを開始する準備が整います。バルクレコードのインポートには、次の 2 つの選択肢があります。

- アイテムインタラクション、ユーザー、アイテムデータセットでは、Amazon SageMaker Data Wrangler を使用して 40 以上のソースからデータをインポートし、視覚化と Amazon Personalize 固有のインサイトを生成し、Amazon Personalize の要件を満たすように変換できます。
- すべてのデータセットタイプで、バルクデータをデータセットに直接インポートできます。直接インポートする場合は、Amazon Personalize の要件を満たすようにデータを手動でフォーマットし、Amazon S3 にアップロードします。次に、スキーマとデータセットを作成し、データセットインポートジョブを使用してデータをデータセットに直接インポートします。

次のガイドラインは、バルクデータが正しくフォーマットされていることを確認するのに役立ちます。

- 入力データはコンマ区切り値 (CSV) ファイルである必要があります。

- CSV ファイルの最初の行には、列ヘッダーが含まれている必要があります。ヘッダーを引用符 (") で囲まないでください。
- データセットタイプに必要なフィールドがあることを確認し、その名前が Amazon Personalize の要件と一致していることを確認してください。例えば、アイテムデータには、各アイテムの ID を持つ ITEM_IDENTIFICATION_NUMBER と呼ばれる列を含むことがあります。この列を ITEM_ID フィールドとして使用するには、列の名前を ITEM_ID に変更します。Data Wrangler を使用してデータをフォーマットする場合は、Amazon Personalize Data Wrangler のマップ列変換を使用して、列に正しい名前が付けられていることを確認できます。

  必要なフィールドについての詳細は、「スキーマ」を参照してください。Data Wrangler を使用してデータを準備する方法については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。
- CSV ファイルの列ヘッダー名は、スキーマに対応している必要があります。
- CSV ファイルの各レコードは 1 行にする必要があります。
- 各列のデータ型はスキーマに対応している必要があります。Data Wrangler を使用してデータをフォーマットする場合、Data Wrangler 変換の Parse Value as Type を使用してデータ型を変換できます。
- TIMESTAMP および CREATION_TIMESTAMP データは UNIX エポック時間形式である必要があります。詳細については、「タイムスタンプのデータ」を参照してください。
- 項目 ID、ユーザー ID、およびアクション ID データに文字"や特殊文字を含めないでください。
- データに ASCII でエンコードされていない文字が含まれている場合は、CSV ファイルを UTF-8 形式でエンコードする必要があります。
- テキストデータは必ず、「非構造化テキストメタデータ」で説明されているとおりにフォーマットしてください。
- インプレッションデータとカテゴリ別データは、「明示的なインプレッションのフォーマット」と「カテゴリ別データのフォーマット」で説明されているとおりにフォーマットしてください。

Amazon Personalize の一括データフォーマット要件の詳細については、「データ形式ガイドライン」を参照してください。

Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

レコメンダーをすでに作成しているか、キャンペーンを含むカスタムソリューションバージョンをデプロイしている場合、新しい一括レコードがレコメンデーションにどのように影響するかは、使用す

るドメインのユースケースまたはレシピによって異なります。詳細については、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。

一括レコードのフィルター更新

一括インポートの完了から 20 分以内に、Amazon Personalize は、データセットグループで作成したすべてのフィルターを、新しいアイテムおよびユーザーデータで更新します。この更新により、Amazon Personalize はユーザーのレコメンデーションをフィルタリングする際に最新のデータを使用できるようになります。

トピック

- Amazon SageMaker Data Wrangler を使用したデータの準備とインポート
- Amazon Personalize データセットへのデータの直接インポート

Amazon SageMaker Data Wrangler を使用したデータの準備とインポート

> **Important**
>
> Data Wrangler を使用すると、 SageMaker コストが発生します。料金と料金の完全なリストについては、Amazon SageMaker 料金表の Data Wrangler タブを参照してください。追加料金が発生しないように、使用が終了したら Data Wrangler インスタンスをシャットダウンしてください。詳細については、「Data Wrangler をシャットダウンする」を参照してください。

データセットグループを作成したら、Amazon SageMaker Data Wrangler (Data Wrangler) を使用して、40 以上のソースから Amazon Personalize データセットにデータをインポートできます。Data Wrangler は、データをインポート、準備、変換、分析するための end-to-end ソリューションを提供する Amazon SageMaker Studio Classic の機能です。Data Wrangler を使用してデータを準備し、アクションデータセットまたはアクションインタラクションデータセットにインポートすることはできません。

Data Wrangler を使用してデータを準備してインポートする場合は、データフローを使用します。データフローは、データのインポートから始まる一連の機械学習データ準備ステップを定義します。フローにステップを追加するたびに、Data Wrangler はデータの変換やビジュアライゼーションの生成などのアクションをデータに対して実行します。

Amazon Personalize のデータを準備するためにフローに追加できるいくつかのステップを以下に示します。

- インサイト: Amazon Personalize 固有のインサイトステップをフローに追加できます。これらのインサイトは、データについて、またデータを改善するために実行できるアクションについて知るのに役立ちます。
- ビジュアライゼーション: ビジュアライゼーションステップを追加して、ヒストグラムや散布図などのグラフを生成できます。グラフは、外れ値や欠損値など、データ内の問題を発見するのに役立ちます。
- 変換: Amazon Personalize 固有の変換ステップと一般的な変換ステップを使用して、データが Amazon Personalize の要件を満たしていることを確認できます。Amazon Personalize 変換は、Amazon Personalize データセットタイプに応じてデータ列を必要な列にマッピングするのに役立ちます。

Amazon Personalize にデータをインポートする前に Data Wrangler を終了する必要がある場合は、Amazon Personalize コンソールから Data Wrangler を起動するときに同じデータセットタイプを選択することで、中断したところに戻ることができます。または、 SageMaker Studio Classic から Data Wrangler に直接アクセスできます。

以下のように、Data Wrangler から Amazon Personalize にデータをインポートすることをお勧めします。変換、視覚化、分析のステップはオプションで繰り返し可能で、どの順序でも実行できます。

1. アクセス許可の設定 - Amazon Personalize と SageMaker サービスロールのアクセス許可を設定します。そして、ユーザーの許可を設定します。
2. Amazon Personalize コンソールから SageMaker Studio Classic で Data Wrangler を起動する - Amazon Personalize コンソールを使用して SageMaker ドメインを設定し、 SageMaker Studio Classic で Data Wrangler を起動します。
3. Data Wrangler へのデータのインポート - 40 以上のソースから Data Wrangler にデータをインポートします。ソースには、Amazon Redshift、Amazon EMR、Amazon Athena などの AWS サービスや、Snowflake や などのサードパーティーが含まれます DataBricks。
4. データの変換 - Data Wrangler を使用して、Amazon Personalize の要件を満たすようにデータを変換します。
5. データの視覚化と分析 - Data Wrangler を使用してデータを視覚化し、Amazon Personalize 固有のインサイトを通じて分析します。
6. Amazon Personalize へのデータの処理とインポート - SageMaker Studio Classic Jupyter Notebook を使用して、処理されたデータを Amazon Personalize にインポートします。

追加情報

以下のリソースは、Amazon SageMaker Data Wrangler と Amazon Personalize の使用に関する追加情報を提供します。

- サンプルデータセットの処理と変換のチュートリアルについては、Amazon SageMaker デベロッパーガイドの「デモ: Data Wrangler Titanic Dataset チュートリアル」を参照してください。このチュートリアルでは、Data Wrangler のフィールドと機能を紹介します。
- Amazon SageMaker ドメインへのオンボーディングの詳細については、「Amazon SageMaker デベロッパーガイド」の「Amazon SageMaker ドメインへのクイックオンボード」を参照してください。
- Amazon Personalize のデータセットとスキーマの要件については、「データ形式ガイドライン」および「スキーマ」を参照してください。

アクセス許可のセットアップ

Data Wrangler を使用してデータを準備するには、以下の権限を設定する必要があります。

- Amazon Personalize のサービスロールを作成する: まだ作成していない場合は、「Amazon Personalize の設定」の「Amazon Personalize 用の IAM サービスロールを作成する」の手順を完了してください。このロールには、処理されたデータを保存する Amazon S3 バケットの `GetObject` および `ListBucket` アクセス権限が必要です。また、任意の AWS KMS キーを使用するためのアクセス許可が必要です。

  Amazon S3 バケットへのアクセス権を Amazon Personalize に付与する方法の詳細については、「Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与」を参照してください。Amazon Personalize に AWS KMS キーへのアクセスを許可する方法については、「」を参照してくださいAWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する。
- アクセス SageMaker 許可を持つ管理ユーザーを作成する： 管理者は へのフルアクセス SageMaker を持ち、 SageMaker ドメインを作成できる必要があります。詳細については、「Amazon SageMaker デベロッパーガイド」の「管理ユーザーとグループの作成」を参照してください。
- SageMaker 実行ロールの作成： SageMaker リソースと Amazon Personalize データインポートオペレーションへのアクセス権を持つ SageMaker 実行ロールを作成します。 SageMaker 実行ロールにはポリシーが`AmazonSageMakerFullAccess`アタッチされている必要があります。より詳細な Data Wrangler アクセス許可が必要な場合は、「Amazon SageMaker デベロッパーガイド」

の「Data Wrangler のセキュリティとアクセス許可」を参照してください。 SageMaker ロールの詳細については、SageMaker 「 ロール」を参照してください。

Amazon Personalize のデータインポートオペレーションへのアクセスを許可するには、次の IAM ポリシーを実行 SageMaker ロールにアタッチします。このポリシーは、Amazon Personalize にデータをインポートし、Amazon S3 バケットにポリシーをアタッチするために必要なアクセス権限を付与します。また、サービスが Amazon Personalize の場合は、PassRole アクセス権限が付与されます。Data Wrangler でデータを準備したら、Amazon S3 bucket-name をフォーマットされたデータの送信先として使用したい Amazon S3 バケットの名前に更新します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:Create*",
                "personalize:List*",
                "personalize:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "personalize.amazonaws.com"
                }
```

```
            }
        }
    ]
}
```

IAM ポリシーの作成については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。IAM ポリシーのロールへのアタッチについての詳細は、「IAM ユーザーガイド」の「IAM ID のアクセス許可の追加および削除」を参照してください。

Amazon Personalize から Data Wrangler を起動する

Amazon Personalize から Data Wrangler を起動するには、Amazon Personalize コンソールを使用して SageMaker ドメインを設定し、Data Wrangler を起動します。

Amazon Personalize から Data Wrangler を起動するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。
3. [データセットの設定] で [データセットの作成] を選択し、作成するデータセットのタイプを選択します。Data Wrangler を使用してアクションデータセットまたはアクションインタラクションデータセットを準備することはできません。
4. [Data Wrangler を使用してデータをインポート] を選択し、[次へ] を選択します。
5. SageMaker ドメイン では、既存のドメインを使用するか、新しいドメインを作成するかを選択します。 SageMaker Studio Classic で SageMaker Data Wrangler にアクセスするにはドメインが必要です。ドメインとユーザープロファイルの詳細については、「Amazon SageMaker デベロッパーガイド」のSageMaker「ドメイン」を参照してください。
6. 既存のドメインを使用するには、SageMaker ドメインとユーザープロファイルを選択してドメインを設定します。
7. 新しいドメインを作成するには

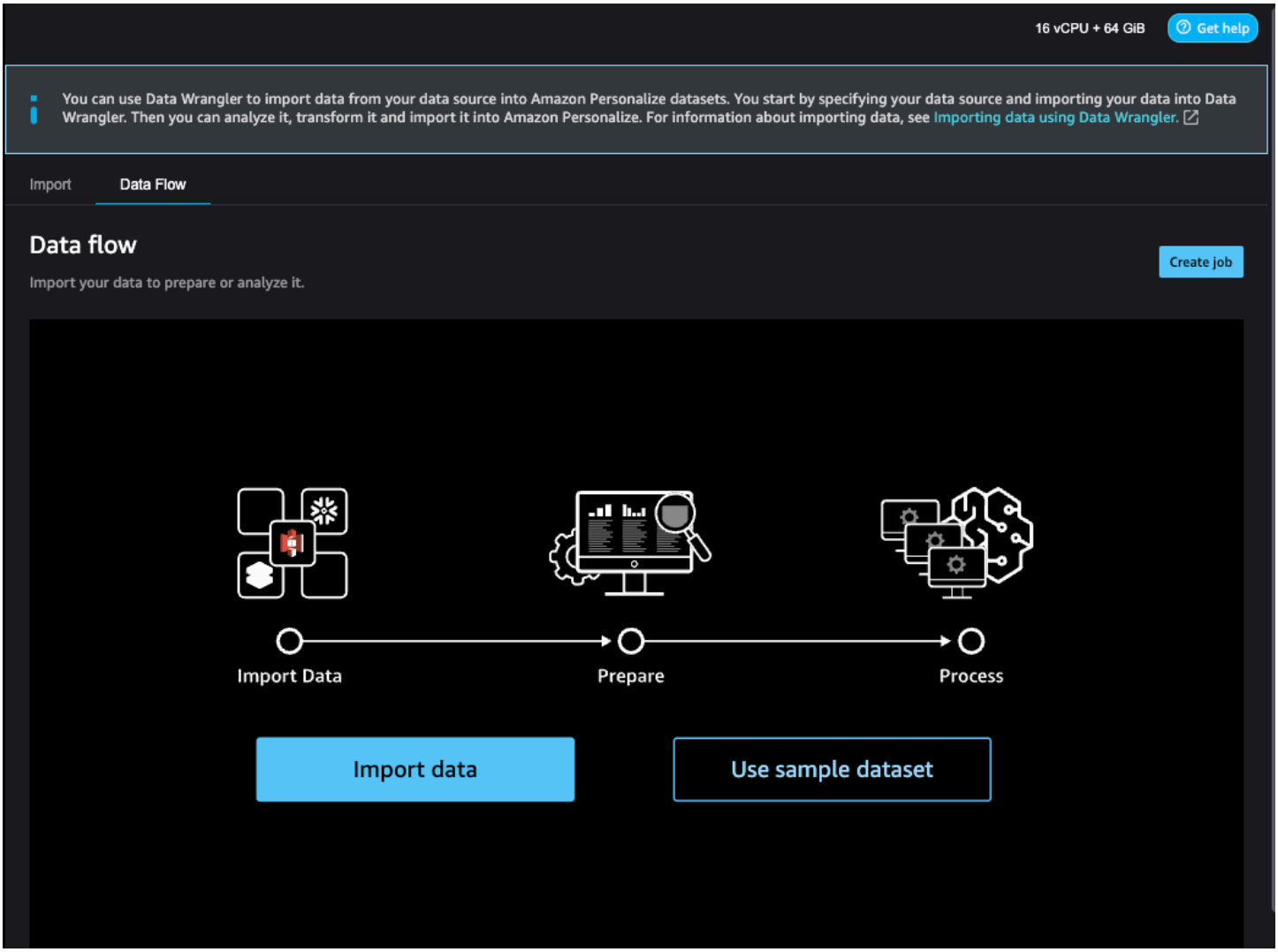
   - 新しいドメインに名前を付けます。
   - ユーザープロファイル名を選択します。
   - [実行ロール] で、「アクセス許可のセットアップ」で作成したロールを選択します。または、CreateRole アクセス許可がある場合は、ロール作成ウィザードを使用して新しいロールを作成します。使用するロールには、`AmazonSageMakerFullAccess` ポリシーがアタッチされている必要があります。

8. [次へ] をクリックします。新しいドメインを作成する場合、 はドメインの作成 SageMaker を開始します。これには最大 10 分かかることがあります。
9. SageMaker ドメインの詳細を確認します。
10. Data Wrangler でデータをインポートを選択します。 SageMaker Studio Classic が環境の作成を開始し、完了すると、 SageMaker Studio Classic の Data Wrangler のデータフローページが新しいタブで開きます。 SageMaker Studio Classic が環境の作成を完了するまでに、最大 5 分かかる場合があります。作成が完了すると、Data Wrangler へのデータのインポートを開始する準備が整います。詳細については、「Data Wrangler へのデータのインポート」を参照してください。

Data Wrangler へのデータのインポート

SageMaker ドメインを設定し、新しいタブで Data Wrangler を起動すると、ソースから Data Wrangler にデータをインポートする準備が整います。Data Wrangler を使用して Amazon Personalize 用のデータを準備する場合、一度に 1 つのデータセットをインポートします。アイテムインタラクションデータセットから始めることをお勧めします。Data Wrangler を使用してアクションデータセットまたはアクションインタラクションデータセットを準備することはできません。

データフローページから開始します。このページは次の図のように表示されます。



データのインポートを開始するには、[データのインポート] を選択し、データソースを指定します。Data Wrangler は 40 以上のソースをサポートしています。これには、Amazon Redshift、Amazon EMR、Amazon Athena などの AWS サービスや、Snowflake や などのサードパーティーが含まれます DataBricks。 Amazon Athena データソースが異なれば、データの接続とインポートの手順も異なります。

利用可能なソースの完全なリストとデータのインポート step-by-step 手順については、Amazon SageMaker デベロッパーガイドの「インポート」を参照してください。

Data Wrangler にデータをインポートしたら、データを変換する準備が整います。データ変換については、「データ変換」を参照してください。

データ変換

Data Wrangler でデータを変換するには、データフローに変換ステップを追加します。Data Wrangler には、Amazon Personalize マップ列変換など、データの準備に使用できる 300 を超える変換が含まれています。また、一般的な Data Wrangler 変換を使用して、外れ値、型の問題、欠損値などの問題を修正できます。

データの変換が完了したら、Data Wrangler で分析できます。または、Data Wrangler でデータを準備し終えたら、そのデータを処理して Amazon Personalize にインポートできます。データの分析については、「ビジュアライゼーションとデータインサイトの生成」を参照してください。データの処理とインポートについては、「データを処理して Amazon Personalize にインポートする」を参照してください。

トピック

- Amazon Personalize マッピング列
- 一般的な Data Wrangler 変換

Amazon Personalize マッピング列

Amazon Personalize の要件を満たすようにデータを変換するには、Amazon Personalize のマップ列変換を追加し、その列を Amazon Personalize の必須フィールドとオプションフィールドにマッピングします。

Amazon Personalize のマップ列変換を使用するには

1. 最新の変換に [+] を選択し、[変換を追加] を選択します。変換をまだ追加していない場合は、データ型変換に [+] を選択してください。Data Wrangler は、この変換をフローに自動的に追加します。
2. [Add step] (ステップを追加) を選択します。
3. [Amazon Personalize の変換] を選択してください。Amazon Personalize のマップ列変換はデフォルトで選択されています。
4. 変換フィールドを使用して、データを必要な Amazon Personalize 属性にマッピングします。
    1. データに一致するデータセットタイプ (インタラクション、アイテム、ユーザー) を選択します。
    2. ドメイン (Eコマース、VIDEO_ON_DEMAND、またはカスタム) を選択します。選択するドメインは、データセットグループを作成したときに指定したドメインと一致する必要があります。

3. Amazon Personalize の必須フィールドとオプションフィールドに一致する列を選択してください。例えば、item_ID 列では、各商品の固有識別情報を格納するデータ内の列を選択します。

   各列フィールドはデータタイプによってフィルタ処理されます。Amazon Personalize のデータ型要件を満たすデータ内の列のみを使用できます。データが必要なタイプでない場合は、Parse Value as Type Data Wrangler 変換を使用して変換できます。

一般的な Data Wrangler 変換

以下の一般的な Data Wrangler 変換は、Amazon Personalize 用のデータを準備するのに役立ちます。

- データ型変換: Amazon Personalizeのマップ列変換に使用可能なオプションとしてフィールドが表示されていない場合は、そのデータ型を変換する必要がある場合があります。Data Wrangler 変換の Parse Value as Type は、データの変換に役立ちます。また、フローの作成時に Data Wrangler がデフォルトで追加するデータ型変換を使用することもできます。この変換を使用するには、[タイプ] ドロップダウンリストからデータタイプを選択し、[プレビュー]を選択してから[更新]を選択します。

  フィールドに必要なデータ型については、「スキーマ」でドメインとデータセットタイプのセクションを参照してください。
- 欠損値と外れ値の処理:欠損値や外れ値のインサイトを生成した場合は、Data Wrangler 変換「外れ値の処理と欠損値の処理を使用してこれらの問題を解決できます。
- カスタム変換: Data Wrangler では、Python (ユーザー定義関数) 、pandas PySpark、または PySpark (SQL) を使用して独自の変換を作成できます。カスタム変換を使用して、重複する列の削除や列によるグループ化などのタスクを実行できます。詳細については、「Amazon SageMaker デベロッパーガイド」の「カスタム変換」を参照してください。

ビジュアライゼーションとデータインサイトの生成

データを Data Wrangler にインポートすると、そのデータを使用してビジュアライゼーションやデータインサイトを生成できます。

- ビジュアライゼーション: Data Wrangler は、ヒストグラムや散布図など、さまざまなタイプのグラフを生成できます。例えば、ヒストグラムを生成して、データの外れ値を特定できます。

- データインサイト: Amazon Personalize のデータ品質およびインサイトレポートを使用すると、データインサイトと列と行の統計を通じてデータについて知ることができます。このレポートにより、データに何らかのタイプの問題があるかどうかがわかります。また、データを改善するためにどのようなアクションを取ればよいのかを知ることができます。これらのアクションは、モデルのトレーニングの要件などの Amazon Personalize のリソース要件を満たすのに役立つ場合や、レコメンデーションの改善につながる場合があります。

ビジュアライゼーションとインサイトを通じてデータについて学習したら、その情報を活用してさらに変換を適用してデータを改善することができます。または、データの準備が完了したら、データを処理して Amazon Personalize にインポートできます。データ変換については、「データ変換」を参照してください。データの処理とインポートについては、「データを処理して Amazon Personalize にインポートする」を参照してください。

ビジュアライゼーションの生成

Data Wrangler を使用して、ヒストグラムや散布図など、さまざまなタイプのグラフを作成できます。例えば、ヒストグラムを生成して、データの外れ値を特定できます。データビジュアライゼーションを生成するには、フローに分析ステップを追加し、[分析タイプ] から作成したいビジュアライゼーションを選択します。

Data Wrangler でのビジュアライゼーションの作成の詳細については、「Amazon SageMaker デベロッパーガイド」の「分析とビジュアライゼーション」を参照してください。

データ分析情報の生成

Data Wrangler を使用して、データセットタイプに固有の Amazon Personalize のデータ品質およびインサイトレポートを生成できます。レポートを生成する前に、Amazon Personalize の要件を満たすようにデータを変換することをお勧めします。これにより、より関連性の高いインサイトが得られます。詳細については、「データ変換」を参照してください。

トピック

- レポートの内容
- レポートの生成

レポートの内容

Amazon Personalize のデータ品質およびインサイトレポートには、以下のセクションが含まれます。

- 概要: レポートの概要には、データセットの統計と優先度の高い警告が含まれます。
  - データセット統計: これらには、インタラクションデータ内のユニークユーザー数などの Amazon Personalize 固有の統計と、欠損値や外れ値の数などの一般的な統計が含まれます。
  - 優先度の高い警告: これらは、トレーニングや推奨事項に最も影響する Amazon Personalize 固有のインサイトです。各警告には、問題を解決するために実行できる推奨アクションが含まれています。
- 重複行と不完全な行: これらのセクションには、データ内のどの行に欠損値があるか、どの行が重複しているかに関する情報が含まれます。
- 機能の概要: このセクションには、各列のデータタイプ、無効または欠落しているデータの情報、警告数が含まれます。
- 機能の詳細: このセクションには、各データ列の詳細情報が記載されたサブセクションがあります。各サブセクションには、カテゴリ値の数や欠損値情報など、列の統計情報が含まれています。また、各サブセクションには、Amazon Personalize 固有のインサイトとデータ列に関する推奨アクションが含まれています。例えば、ある列に 30 を超えるカテゴリがあるというインサイトがある場合があります。

データ型の問題

レポートでは、データ型が正しくない列を特定し、必要なデータ型を指定します。これらの機能に関するインサイトを得るには、列のデータ型を変換してレポートを再生成する必要があります。タイプを変換するには、Data Wrangler 変換の Parse Value as Type を使用できます。

Amazon Personalize のインサイト

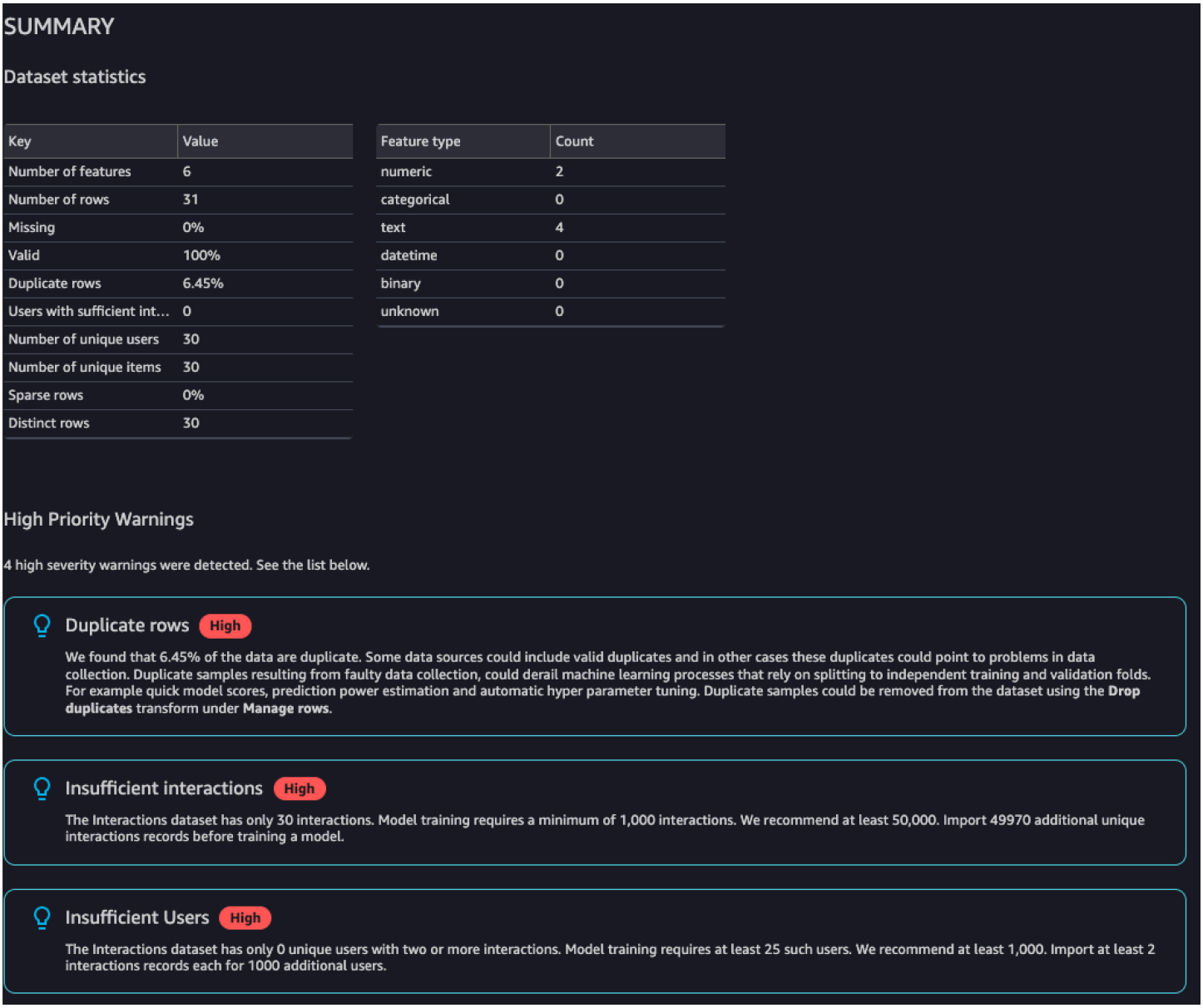
Amazon Personalize のインサイトには、結果と推奨アクションが含まれます。このアクションはオプションです。例えば、レポートには、カテゴリデータ列のカテゴリ数に関するインサイトとアクションが含まれる場合があります。列がカテゴリに分類されていないと思われる場合は、このインサイトは無視して何も起こさないでください。

表現のわずかな違いを除いて、Amazon Personalize 固有のインサイトは、Amazon Personalize でデータを分析したときに生成される 1 つのデータセットのインサイトと同じです。例えば、Data Wrangler のインサイトレポートには、「アイテムインタラクションデータセットには、インタラクションが 2 回以上あるユニークユーザーが X 人しかいない」などのインサイトが含まれています。ただし、「アイテムデータセットの X% のアイテムで、アイテムインタラクションデータセット内でインタラクションが発生していない」などのインサイトは含まれていません。

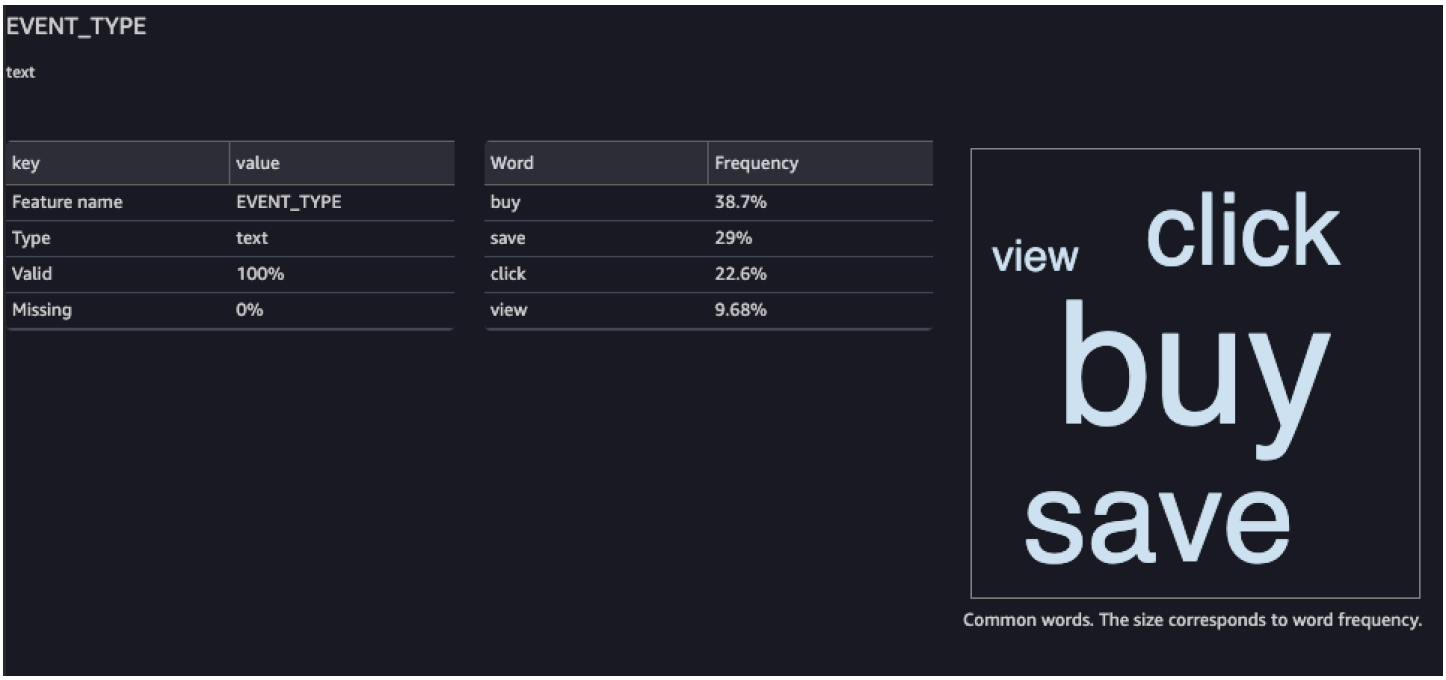
Amazon Personalize 固有のインサイトのリストについては、「データインサイト」の複数のデータセットを参照していないインサイトを参照してください。

レポートの例

Amazon Personalize レポートのルックアンドフィールは、Data Wrangler の一般的なインサイトレポートと同じです。一般的なインサイトレポートの例については、「Amazon SageMaker デベロッパーガイド」の「Get Insights On Data and Data Quality」を参照してください。次の例は、アイテムインタラクションデータセットのレポートの概要セクションがどのように表示されるかを示しています。これには、データセットの統計情報と、優先度が高いと思われるアイテムインタラクションデータセットの警告が含まれています。



次の例は、アイテムインタラクションデータセットの EVENT_TYPE 列の機能の詳細セクションがレポートにどのように表示されるかを示しています。



レポートの生成

Amazon Personalize のデータ品質およびインサイトレポートを生成するには、[変換に必要なデータインサイトを取得] を選択し、分析を作成します。

Amazon Personalize のデータ品質およびインサイトレポートを生成するには

1. 分析しているトランスフォームの [+] オプションを選択します。トランスフォームをまだ追加していない場合は、データ型トランスフォームに [+] を選択してください。Data Wrangler は、このトランスフォームをフローに自動的に追加します。
2. [データインサイトを取得 を選択します。[分析の作成] パネルが表示されます。
3. 分析タイプには、Amazon Personalize のデータ品質およびインサイトレポートを選択します。
4. [データセットタイプ] では、分析する Amazon Personalize データセットのタイプを選択します。
5. オプションで [フルデータで実行] を選択します。デフォルトでは、Data Wrangler はデータのサンプルについてのみインサイトを生成します。
6. [作成] を選択します。分析が完了すると、レポートが表示されます。

データを処理して Amazon Personalize にインポートする

データの分析と変換が完了したら、データを処理して Amazon Personalize にインポートする準備が整います。

- データの処理 — データを処理すると、変換がデータセット全体に適用され、指定した宛先に出力されます。この場合は、Amazon S3 バケットを指定します。
- Amazon Personalize へのデータのインポート – 処理されたデータを Amazon Personalize にインポートするには、 SageMaker Studio Classic で提供されている Jupyter Notebook を実行します。このノートブックは Amazon Personalize データセットを作成し、そのデータセットにデータをインポートします。

データの処理

Amazon Personalize にデータをインポートする前に、データセット全体に変換を適用して、Amazon S3 バケットに出力する必要があります。これを行うには、送信先を Amazon S3 バケットに設定して宛先ノードを作成し、変換の処理ジョブを起動します。

送信先を指定し、プロセスジョブを起動する step-by-step 手順については、「Amazon SageMaker Data Wrangler を使用して数回クリックするだけで処理ジョブを起動する」を参照してください。デスティネーションを追加するときは、[Amazon S3] を選択します。この場所は、処理されたデータを Amazon Personalize にインポートするときに使用します。

データの処理が終了すると、Amazon S3 バケットから Amazon Personalize にデータをインポートする準備が整います。

Amazon Personalize にデータをインポートする

データを処理したら、Amazon Personalize にインポートする準備が整います。処理されたデータを Amazon Personalize にインポートするには、 SageMaker Studio Classic で提供されている Jupyter Notebook を実行します。このノートブックは Amazon Personalize データセットを作成し、そのデータセットにデータをインポートします。

処理されたデータを Amazon Personalize にインポートするには

1. エクスポートしたいトランスフォーメーションについては、[エクスポート先] を選択し、[Amazon Personalize (Jupyter Notebook 経由)] を選択します。
2. ノートブックを変更して、処理ジョブのデータ送信先として使用した Amazon S3 バケットを指定します。オプションで、データセットグループのドメインを指定します。デフォルトでは、ノートブックはカスタムデータセットグループを作成します。

3. スキーマを作成しているノートブックのセルを確認してください。セルを実行する前に、スキーマのフィールドが期待どおりのタイプと属性であることを確認してください。

   - NULL データをサポートするフィールドがタイプのリストに null がリストされていることを確認してください。次の例は、フィールドに null を追加する方法を示しています。

     ```
     {
       "name": "GENDER",
       "type": [
         "null",
         "string"
       ],
       "categorical": true
     }
     ```

   - カテゴリフィールドのカテゴリ属性が true に設定されていることを確認します。次の例は、フィールドをカテゴリとしてマークする方法を示しています。

     ```
     {
               "name": "SUBSCRIPTION_MODEL",
               "type": "string",
               "categorical": true
     }
     ```

   - テキストフィールドのテキスト属性が true に設定されていることを確認します。次の例は、フィールドをテキストとしてマークする方法を示しています。

     ```
     {
           "name": "DESCRIPTION",
           "type": [
             "null",
             "string"
           ],
           "textual": true
     }
     ```

4. ノートブックを実行してスキーマを作成し、データセットを作成し、データを Amazon Personalize データセットにインポートします。 SageMaker Studio Classic の外部にあるノートブックと同じようにノートブックを実行します。Jupyter Notebook の実行について詳しくは、「コードの実行」を参照してください。 SageMaker Studio Classic のノートブックの詳細につ

いては、「Amazon デベロッパーガイド」の「Amazon SageMaker Notebooks の使用」を参照してください。 SageMaker

ノートブックが完成したら、インタラクションデータをインポートすれば、レコメンダーやカスタムリソースを作成できます。または、アイテムデータセットまたはユーザーデータセットでこのプロセスを繰り返すこともできます。レコメンダーまたはカスタムリソースの作成の詳細については、「ステップ 3: レコメンダーまたはカスタムリソースを作成する」を参照してください。

Amazon Personalize データセットへのデータの直接インポート

データセットを作成したら、大きな CSV ファイルなどのバルクレコードを Amazon Personalize のデータセットにインポートできます。

データを直接Amazon Personalize にデータセットにインポートするには、次の操作を実行します。

1. データに基づいてスキーマ JSON ファイルを作成します。スキーマの要件と例については、「スキーマ」を参照してください。
2. データが正しいフォーマットであることを確認してください。列名はスキーマと一致している必要があります。データは CSV ファイルである必要があります。データ形式のガイドラインについては、「データ形式ガイドライン」を参照してください。
3. CSV ファイルを Amazon Simple Storage Service (Amazon S3) バケットにアップロードし、Amazon S3 リソースへのアクセス権を Amazon Personalize に付与します。
4. ステップ 1 の JSON ファイルを使用して Amazon Personalize スキーマを作成します。そして、Amazon Personalize のデータセットを作成します。
5. Amazon S3 バケットからのデータをデータセットに取り込むデータセットのインポートジョブを作成します。インタラクションデータセットのデータセットインポートジョブを作成するには、CSV ファイルに 1,000 件以上のインタラクションレコードが含まれている必要があります。

Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

トピック

- Amazon S3 バケットへのアップロード
- データセットとスキーマの作成

- データセットのインポートジョブを使用したバルクレコードのインポート

Amazon S3 バケットへのアップロード

履歴入力データをフォーマットした後 (「データ形式ガイドライン」を参照)、CSV ファイルを Amazon S3 バケットにアップロードし、Amazon Personalize に Amazon S3 リソースへの許可を付与する必要があります。

1. まだ行っていない場合は、アクセス許可のセットアップ の手順に従ってアクセス許可を設定して、Amazon Personalize がユーザーに代わって、Amazon Personalize リソースにアクセスできるようにします。
2. Amazon Simple Storage Service (Amazon S3) バケットに CSV ファイルをアップロードします。これは、Amazon Personalize によるデータのインポート元です。詳細については、Amazon Simple Storage Service ユーザーガイドの「ドラッグアンドドロップを使用したファイルとフォルダのアップロード」を参照してください。
3. アクセスポリシーを Amazon S3 バケットと Amazon Personalize のサービスロールにアタッチすることにより、Amazon S3 リソースへのアクセス権を Amazon Personalize に付与します。Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与 を参照してください。

   暗号化に AWS Key Management Service (AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize IAM サービスロールにキーを使用するアクセス権限を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

Amazon S3 バケットにデータをアップロードし、Amazon Personalize に Amazon S3 へのアクセス権を付与したら、Amazon Personalize スキーマとデータセットを作成する準備が整います。「データセットとスキーマの作成」を参照してください。

データセットとスキーマの作成

ステップ 1: データセットグループを作成する を完了したら、データセットを作成する準備が整います。データセットは、データ用の Amazon Personalize のコンテナです。データセットを作成するときは、データセットのスキーマも作成します。スキーマは、Amazon Personalize にデータの構造を知らせ、Amazon Personalize がデータを解析できるようにします。

Amazon Personalize コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDK を使用してデータセットを作成します。ドメインデータセットグループには、アクションやアクション

インタラクションデータセットなどのネクストベストアクションリソースを作成することはできません。さまざまなタイプのデータセット、およびデータセットとスキーマの要件については、「データセットとスキーマ」を参照してください。

トピック

- データセットとスキーマの作成 (コンソール)
- データセットとスキーマの作成 (AWS CLI)
- データセットとスキーマ (AWS SDK) の作成

データセットとスキーマの作成 (コンソール)

これがデータセットグループの最初のデータセットである場合、最初のデータセットタイプはアイテムインタラクションデータセットになります。コンソールでアイテムインタラクションデータセットを作成するには、データセット名を指定してから、Avro 形式の JSON スキーマを指定します。このデータセットグループの最初のデータセットでない場合は、データセットのタイプを選択してから、名前とスキーマを指定します。

Amazon Personalize のデータセットとスキーマの要件については、「データセットとスキーマ」を参照してください。

> **Note**
>
> 「ステップ 1: データセットグループを作成する」を完了したばかりで、既にデータセットを作成している場合は、この手順のステップ 4 にスキップしてください。

データセットとスキーマを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のページで、ステップ 1: データセットグループを作成する で作成したデータセットグループを選択します。
3. [データセットの設定] で [データセットの作成] を選択し、作成するデータセットのタイプを選択します。
4. [Amazon Personalize データセットにデータを直接インポートする] を選択し、[次へ] を選択します。

5. [Dataset details] (データセットの詳細) の [Dataset name] (データセット名) で、データセットの名前を指定します。
6. [Schema details] (スキーマの詳細) の [Schema selection] (スキーマを選択) で、既存のスキーマを選択するか、[Create new schema] (新しいスキーマを作成) を選択します。
7. 新しいスキーマを作成する場合は、[Schema definition] (スキーマの定義) で、データに一致するスキーマ JSON を貼り付けます。スキーマ の例をガイドとして使用してください。スキーマを作成した後は、スキーマに変更を加えることはできなくなります。
8. [New schema name (新しいスキーマ名)] で、新しいスキーマの名前を指定します。
9. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
10. [Next] (次へ) を選択し、バルクデータの準備とインポート の指示に従ってデータをインポートします。

データセットとスキーマの作成 (AWS CLI)

を使用してデータセットとスキーマを作成するには AWS CLI、まず Avro 形式でスキーマを定義し、オペレーションを使用して Amazon Personalize に追加します。CreateSchemaその後、CreateDataset 操作を使用してデータセットを作成します。Amazon Personalize のデータセットとスキーマの要件については、「データセットとスキーマ」を参照してください。

スキーマとデータセットを作成するには

1. Avro 形式のスキーマファイルを作成し、JSON ファイルとして保存します。このファイルは、作成している Interactions などのデータセットのタイプに基づいている必要があります。

   スキーマはデータの列と一致する必要があり、スキーマ `name` は Amazon Personalize で認識されるデータセットのタイプのいずれかと一致する必要があります。最小限のアイテムインタラクションデータセットスキーマの例を次に示します。その他の例については、「 スキーマ」を参照してください。

   ```
   {
     "type": "record",
     "name": "Interactions",
     "namespace": "com.amazonaws.personalize.schema",
     "fields": [
         {
             "name": "USER_ID",
             "type": "string"
   ```

```
        },
        {
            "name": "ITEM_ID",
            "type": "string"
        },
        {
            "name": "TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}
```

2. 次のコマンドを実行して Amazon Personalize でスキーマを作成します。スキーマを作成した後は、スキーマに変更を加えることはできなくなります。schemaName をスキーマの名前に、file://SchemaName.json を前のステップで作成した JSON ファイルの場所に、それぞれ置き換えます。次の例は、現在のフォルダに属するファイルを示しています。

   ドメインデータセットグループ内のデータセットのスキーマを作成する場合は、domain パラメータを追加して ECOMMERCE または VIDEO_ON_DEMAND に設定します。API の詳細については、「CreateSchema」を参照してください。

```
aws personalize create-schema \
  --name SchemaName \
  --schema file://SchemaName.json
```

   次の例に示すように、スキーマの Amazon リソースネーム (ARN) が表示されます。

```
{
  "schemaArn": "arn:aws:personalize:us-west-2:acct-id:schema/SchemaName"
}
```

3. 次のコマンドを実行して空のデータセットを作成します。データセットグループの作成 (AWS CLI) のデータセットグループの Amazon リソースネーム (ARN) と前の手順のスキーマ ARN を指定します。データセットタイプの値は、Interactions、Users、Items、Actions、または Action_Interactions です。API の詳細については、「CreateDataset」を参照してください。

```
aws personalize create-dataset \
  --name Dataset Name \
  --dataset-group-arn Dataset Group ARN \
```

```
  --dataset-type Dataset Type \
  --schema-arn Schema Arn
```

次の例に示すように、データセット ARN が表示されます。

```
{
  "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/DatasetName/
INTERACTIONS"
}
```

4. 後で使用するためにデータセット ARN を記録します。データセットを作成したら、トレーニングデータをインポートする準備が整います。バルクデータの準備とインポート を参照してください。

データセットとスキーマ (AWS SDK) の作成

AWS SDK を使用してデータセットとスキーマを作成するには、まず Avro 形式でスキーマを定義し、オペレーションを使用して Amazon Personalize に追加します。CreateSchemaスキーマを作成した後は、スキーマに変更を加えることはできなくなります。その後、CreateDataset 操作を使用してデータセットを作成します。Amazon Personalize のデータセットとスキーマの要件については、「データセットとスキーマ」を参照してください。

スキーマとデータセットを作成するには

1. Avro 形式のスキーマファイルを作成し、JSON ファイルとして作業ディレクトリに保存します。

   スキーマはデータの列と一致する必要があり、スキーマ name は Amazon Personalize で認識されるデータセットの種類と一致する必要があります。最小限のアイテムインタラクションデータセットスキーマの例を次に示します。その他の例については、「 スキーマ」を参照してください。

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
```

```
        },
        {
            "name": "ITEM_ID",
            "type": "string"
        },
        {
            "name": "TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}
```

2. 次のコードを使用してスキーマを作成します。スキーマの名前とスキーマ JSON ファイルのファイルパスを指定します。

   ドメインデータセットグループ内のデータセットのスキーマを作成する場合は、domain パラメータを追加して ECOMMERCE または VIDEO_ON_DEMAND に設定します。API の詳細については、「CreateSchema」を参照してください。

   SDK for Python (Boto3)

   ```
   import boto3

   personalize = boto3.client('personalize')

   with open('schemaFile.json') as f:
       createSchemaResponse = personalize.create_schema(
           name = 'schema name',
           schema = f.read()
       )

   schema_arn = createSchemaResponse['schemaArn']

   print('Schema ARN:' + schema_arn )
   ```

   SDK for Java 2.x

   ```
   public static String createSchema(PersonalizeClient personalizeClient, String
    schemaName, String filePath) {

       String schema = null;
   ```

```
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                .name(schemaName)
                .schema(schema)
                .build();

        String schemaArn =
 personalizeClient.createSchema(createSchemaRequest).schemaArn();
        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
```

```
} catch (err) {
  mySchema = 'TEST' // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSchemaCommand(createSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Amazon Personalize は、新しいスキーマの ARN を返します。次のステップで必要になるため、これを記録します。

3. CreateDataset 操作を使用してデータセットを作成します。以下のコードは、データセットを作成する方法を示しています。データセットグループの Amazon リソースネーム (ARN)、前のステップのスキーマ ARN を指定し、データセットタイプを指定します。データセットタイプの値は、Interactions、Users、Items、Actions、または Action_Interactions です。データセットのタイプ別の詳細については、「データセットとスキーマ」を参照してください。

   SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset(
    name = 'dataset_name',
    schemaArn = 'schema_arn',
    datasetGroupArn = 'dataset_group_arn',
    datasetType = 'dataset_type'
)
```

```
print ('Dataset Arn: ' + response['datasetArn'])
```

SDK for Java 2.x

```
public static String createDataset(PersonalizeClient personalizeClient,
                                   String datasetName,
                                   String datasetGroupArn,
                                   String datasetType,
                                   String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
                .name(datasetName)
                .datasetGroupArn(datasetGroupArn)
                .datasetType(datasetType)
                .schemaArn(schemaArn).build();

        String datasetArn =
 personalizeClient.createDataset(request).datasetArn();
        System.out.println("Dataset " + datasetName + " created. Dataset ARN: "
 + datasetArn);

        return datasetArn;

    } catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
```

```
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  datasetType: 'DATASET_TYPE', /* required */
  name: 'NAME', /* required */
  schemaArn: 'SCHEMA_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetCommand(createDatasetParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

データセットを作成したら、トレーニングデータをインポートする準備が整います。「バルクデータの準備とインポート」を参照してください。

データセットのインポートジョブを使用したバルクレコードのインポート

入力データをフォーマットし (「データ形式ガイドライン」を参照)、Amazon Simple Storage Service (Amazon S3) バケットにアップロードして (「Amazon S3 バケットへのアップロード」を参照)、データセットとスキーマの作成 を完了したら、データセットのインポートジョブを作成することによってバルクレコードをインポートします。

データセットインポートジョブは、Amazon S3 バケットからのデータをデータセットに取り込む一括インポートツールです。データセットのインポートジョブは、Amazon Personalize コンソール、(AWS CLI)、 AWS Command Line Interface または AWS SDKsを使用して作成できます。

以前にデータセットのデータセットインポートジョブを作成したことがある場合は、新しいデータセットインポートジョブを使用して、既存のバルクデータを追加または置き換えることができます。詳細については、既存のバルクレコードの更新を参照してください。

トピック

- インポートモード
- バルクレコードのインポート (コンソール)

- バルクレコードのインポート (AWS CLI)
- バルクレコードのインポート (AWS SDK)

インポートモード

データセットのインポートジョブを既に作成している場合は、Amazon Personalize が新しいレコードを追加する方法を設定できます。これを行うには、データセットのインポートジョブのインポートモードを指定します。一括レコードをインポートしていない場合、インポートモードフィールドはコンソールでは使用できず、 `CreateDatasetImportJob` API オペレーションFULLでのみ指定できます。デフォルトは完全な置き換えモードです。

- データセット内の既存のバルクデータをすべて上書きするには、Amazon Personalize コンソールで [既存のデータを置換] を選択するか、CreateDatasetImportJob API オペレーションで FULL を指定します。これにより、リアルタイムで記録されたイベントを含め、個別にインポートしたデータが置き換えられることはありません。
- データセット内の既存のデータにレコードを追加するには、[既存のデータに追加] を選択するか、`CreateDatasetImportJob` API オペレーションで `INCREMENTAL` を指定します。Amazon Personalize は、同じ ID のレコードをすべて新しいレコードに置き換えます。

> **Note**
>
> データセットのインポートジョブでアイテムインタラクションデータセットまたはアクションインタラクションデータセットにデータを追加するには、少なくとも 1,000 件の新しいインタラクションレコードまたはアクションインタラクションレコードが必要です。

バルクレコードのインポート (コンソール)

> **Important**
>
> デフォルトでは、データセットのインポートジョブは、一括でインポートしたデータセット内の既存のデータを置き換えます。バルクデータを既にインポートしている場合は、ジョブのインポートモード を変更してデータを追加できます。

Amazon Personalize コンソールを使用してデータセットにバルクレコードをインポートするには、名前、IAM サービスロール、およびデータの場所を使用してデータセットのインポートジョブを作成します。

データセットとスキーマの作成 でデータセットを作成したばかりの場合は、ステップ 5 に進んでください。

バルクレコードをインポートするには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。データセットグループの 概要 が表示されます。
3. ナビゲーションペインで、[データセット] を選択し、バルクデータのインポート先となるデータセットを選択します。
4. [データセットインポートジョブ] で [データセットインポートジョブの作成] を選択します。
5. これが最初のデータセットのインポートジョブである場合は、「データインポートソース」でS3 からデータをインポートする」を選択します。
6. [データセットのインポートジョブ名] で、インポートジョブの名前を指定します。
7. バルクデータを既にインポートしている場合は、インポートモード でデータセットの更新方法を選択します。[既存のデータを置換] または [既存のデータに追加] を選択します。このオプションは、データセットの最初のジョブである場合は表示されません。詳細については、既存のバルクレコードの更新を参照してください。
8. 「データインポートソース」の「データロケーション」で、Amazon S3 にデータファイルを保存する場所を指定します。次の構文を使用します。

   ```
   s3://<name of your S3 bucket>/<folder path>/<CSV filename>
   ```

   CSV ファイルが Amazon S3 バケット内のフォルダにあり、1 つのデータセットインポートジョブで複数の CSV ファイルをデータセットにアップロードしたい場合、フォルダへのパスを指定できます。Amazon Personalize はフォルダの最初のレベルにあるファイルのみを使用し、サブフォルダのデータは一切使用しません。フォルダ名の後に / を付けて次の構文を使用します。

   ```
   s3://<name of your S3 bucket>/<folder path>/
   ```

9. [IAM ロール] で、新しいロールを作成するか、既存のロールを使用するかを選択します。前提条件を満たしたら、[既存のサービスロールを使用する] を選択し、Amazon Personalize 向けの IAM ロールの作成 で作成したロールを指定します。

10. メトリクス属性を作成し、このジョブに関連するメトリクスをAmazon S3 に公開する場合は、[イベントメトリクスを S3 に公開] で [このインポートジョブのメトリクスを公開] を選択します。

    メトリクスがまだ作成されておらず、このジョブのメトリクスを公開したい場合は、[メトリクス属性の作成] を選択して別のタブに新しいメトリクスを作成します。メトリクス属性を作成したら、この画面に戻ってインポートジョブの作成を完了できます。

    メトリクス属性の詳細については、「レコメンデーションの影響の測定」を参照してください。
11. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けについての詳細は、「Amazon Personalize リソースのタグ付け」を参照してください。
12. [Start import (インポートの開始)] を選択します。データインポートジョブが開始され、[Dashboard Overview (ダッシュボード概要)] ページが表示されます。ステータスが ACTIVE と表示されると、データセットのインポートが完了します。Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

    データをインポートしたら、ソリューションを作成する準備が整います。詳細については、「ソリューションとソリューションバージョンの作成」を参照してください。

バルクレコードのインポート (AWS CLI)

> **Important**
>
> デフォルトでは、データセットのインポートジョブは、一括でインポートしたデータセット内の既存のデータを置き換えます。バルクデータを既にインポートしている場合は、ジョブのインポートモード を変更してデータを追加できます。

を使用してバルクレコードをインポートするには AWS CLI、 CreateDatasetImportJob コマンドを使用してデータセットのインポートジョブを作成します。以前にデータセットのデータセットインポートジョブを作成したことがある場合は、インポートモードパラメーターを使用して新しいデータの追加方法を指定できます。既存のバルクデータの更新の詳細については、「既存のバルクレコードの更新」を参照してください。

バルクレコードをインポートする (AWS CLI)

1. 次のコマンドを実行してデータセットのインポートジョブを作成します。データセットに Amazon リソースネーム (ARN) を入力し、トレーニングデータを保存した Amazon S3 バケットへのパスを指定します。パスには次の構文を使用します。

   s3://<name of your S3 bucket>/<folder path>/<CSV filename>

   CSV ファイルが Amazon S3 バケット内のフォルダにあり、1 つのデータセットインポートジョブで複数の CSV ファイルをデータセットにアップロードしたい場合、フォルダへのパスを指定できます。Amazon Personalize はフォルダの最初のレベルにあるファイルのみを使用し、サブフォルダのデータは一切使用しません。フォルダ名の後に / を付けて次の構文を使用します。

   s3://<name of your S3 bucket>/<folder path>/

   で作成した AWS Identity and Access Management (IAM) ロールの Amazon リソースネーム (ARN) を指定しますAmazon Personalize 向けの IAM ロールの作成。`import-mode` のデフォルト値は FULL です。詳細については、既存のバルクレコードの更新を参照してください。オペレーションの詳細については、「CreateDatasetImportJob」を参照してください。

   ```
   aws personalize create-dataset-import-job \
   --job-name dataset import job name \
   --dataset-arn dataset arn \
   --data-source dataLocation=s3://bucketname/filename \
   --role-arn roleArn \
   --import-mode FULL
   ```

   次の例に示すように、データセットのインポートジョブの ARN が表示されます。

   ```
   {
     "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-job/
   DatasetImportJobName"
   }
   ```

2. `describe-dataset-import-job` コマンドを使用してステータスを確認します。前のステップで返されたデータセットのインポートジョブの ARN を指定します。オペレーションの詳細については、「DescribeDatasetImportJob」を参照してください。

   ```
   aws personalize describe-dataset-import-job \
   --dataset-import-job-arn dataset import job arn
   ```

データセットのインポートジョブのプロパティとそのステータスが表示されます。最初、status は [CREATE PENDING] と表示されます。

```
{
  "datasetImportJob": {
      "jobName": "Dataset Import job name",
      "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-
job/DatasetImportJobArn",
      "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/
DatasetGroupName/INTERACTIONS",
      "dataSource": {
          "dataLocation": "s3://<bucketname>/ratings.csv"
      },
      "importMode": "FULL",
      "roleArn": "role-arn",
      "status": "CREATE PENDING",
      "creationDateTime": 1542392161.837,
      "lastUpdatedDateTime": 1542393013.377
  }
}
```

ステータスが ACTIVE と表示されると、データセットのインポートが完了します。Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

データをデータセットグループの関連するデータセットにインポートしたら、ソリューションバージョンを作成します (トレーニング済みモデル)。詳細については、「ソリューションとソリューションバージョンの作成」を参照してください。

バルクレコードのインポート (AWS SDK)

> **Important**
>
> デフォルトでは、データセットのインポートジョブは、一括でインポートしたデータセット内の既存のデータを置き換えます。バルクデータを既にインポートしている場合は、ジョブのインポートモード を変更してデータを追加できます。

データをインポートするには、CreateDatasetImportJob オペレーションを使用してデータセットのインポートジョブを作成します。次のコードは、データセットのインポートジョブを作成する方法を示しています。

ジョブ名を入力し、datasetArn をデータセットの Amazon リソースネーム (ARN) に設定して、dataLocation をトレーニングデータを保存した Amazon S3 バケットへのパスに設定します。パスには次の構文を使用します。

s3://<name of your S3 bucket>/<folder path>/<CSV filename>.csv

CSV ファイルが Amazon S3 バケット内のフォルダにあり、1 つのデータセットインポートジョブで複数の CSV ファイルをデータセットにアップロードしたい場合、フォルダへのパスを指定できます。Amazon Personalize はフォルダの最初のレベルにあるファイルのみを使用し、サブフォルダのデータは一切使用しません。フォルダ名の後に / を付けて次の構文を使用します。

s3://<name of your S3 bucket>/<folder path>/

にはroleArn、S3 バケットへのアクセス許可を Amazon Personalize に付与する AWS Identity and Access Management (IAM) ロールを指定します。Amazon Personalize 向けの IAM ロールの作成 を参照してください。importMode のデフォルト値は FULL です。これにより、データセット内のすべてのバルクデータが置き換えられます。データを追加するには、 に設定しますINCREMENTAL。既存のバルクデータの更新の詳細については、「既存のバルクレコードの更新」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_import_job(
    jobName = 'YourImportJob',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn',
    importMode = 'FULL'
)

dsij_arn = response['datasetImportJobArn']

print ('Dataset Import Job arn: ' + dsij_arn)

description = personalize.describe_dataset_import_job(
```

```
    datasetImportJobArn = dsij_arn)['datasetImportJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['datasetImportJobArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
 personalizeClient,
                                                        String jobName,
                                                        String datasetArn,
                                                        String s3BucketPath,
                                                        String roleArn,
                                                        ImportMode importMode) {

  long waitInMilliseconds = 60 * 1000;
  String status;
  String datasetImportJobArn;

  try {
      DataSource importDataSource = DataSource.builder()
              .dataLocation(s3BucketPath)
              .build();

      CreateDatasetImportJobRequest createDatasetImportJobRequest =
 CreateDatasetImportJobRequest.builder()
              .datasetArn(datasetArn)
              .dataSource(importDataSource)
              .jobName(jobName)
              .roleArn(roleArn)
              .importMode(importMode)
              .build();

      datasetImportJobArn =
 personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
              .datasetImportJobArn();

      DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
 DescribeDatasetImportJobRequest.builder()
              .datasetImportJobArn(datasetImportJobArn)
              .build();
```

```
        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetImportJob datasetImportJob = personalizeClient
                    .describeDatasetImportJob(describeDatasetImportJobRequest)
                    .datasetImportJob();

            status = datasetImportJob.status();
            System.out.println("Dataset import job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return datasetImportJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateDatasetImportJobCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: {
```

```
    dataLocation: 's3://<name of your S3 bucket>/<folderName>/<CSVfilename>.csv'  /*
 required */
  },
  jobName: 'NAME',           /* required */
  roleArn: 'ROLE_ARN',       /* required */
  importMode: "FULL"         /* optional, default is FULL */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

DescribeDatasetImportJob 操作のレスポンスには、操作のステータスが含まれます。

ステータスが ACTIVE に変わるまで待ってから、データを使用してモデルをトレーニングしてください。

ステータスが ACTIVE と表示されると、データセットのインポートが完了します。Amazon Personalize データセットにデータをインポートしたら、分析、Amazon S3 バケットへのエクスポート、更新、またはデータセットの削除による削除を行うことができます。詳細については、「データセット内のトレーニングデータの管理」を参照してください。

データをデータセットグループの関連するデータセットにインポートしたら、ソリューションバージョンを作成します (トレーニング済みモデル)。詳細については、「ソリューションとソリューションバージョンの作成」を参照してください。

個々のレコードのインポート

データセットとスキーマの作成 を完了したら、アイテムインタラクション、ユーザー、アイテム、アクション、またはアクションインタラクションを含む個々のレコードを、既存のデータセットにインポートできます。データを個別にインポートすると、カタログの拡大に合わせ Amazon Personalize のデータセットにレコードの小さなバッチを追加できます。1 回のインポート操作で最大 10 個のレコードをインポートできます。

Apache Kafka を使用している場合は、Amazon Personalize 用 Kafka コネクタを使用して、Amazon Personalize にデータをリアルタイムでストリーミングできます。詳細については、personalize-kafka-connector の Github リポジトリにある「Amazon Personalize 用 Kafka コネクタ」を参照してください。

履歴レコードが大量にある場合は、最初にデータを一括でインポートしてから、必要に応じてデータを個別にインポートすることをお勧めします。「Amazon Personalize データセットへのデータの直接インポート」を参照してください。

レコードを個別にインポートする場合のフィルター更新

Amazon Personalize は、データセットグループで作成したすべてのフィルターを、最後の個別のインポートから 20 分以内に、新しいインタラクション、アイテム、およびユーザーデータで更新します。この更新により、キャンペーンでユーザーのレコメンデーションをフィルタリングするときに最新のデータを使用できるようになります。

レコメンダーをすでに作成しているか、キャンペーンでカスタムソリューションバージョンをデプロイしている場合、新しい個々のレコードがレコメンデーションにどのように影響するかは、使用するドメインのユースケースまたはレシピによって異なります。詳細については、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。

トピック

- インタラクションの個別のインポート
- ユーザーの個別インポート
- アイテムの個別のインポート
- アクションの個別のインポート

インタラクションの個別のインポート

データセットとスキーマの作成 を完了してアイテムインタラクションデータセットを作成したら、1 つ以上の新しいイベントをデータセットに個別にインポートできます。インタラクションイベントを個別にインポートするには、イベントトラッカーを作成してから、1 つ以上のイベントをアイテムインタラクションデータセットにインポートします。Amazon Personalize コンソールを使用して個々の履歴インタラクションイベントをインポートしたり、AWS Command Line Interface (AWS CLI) または AWS SDK を使用して履歴またはリアルタイムのイベントをインポートしたりできます。

このセクションには、Amazon Personalize コンソールを使用したイベントのインポートに関する情報が含まれています。Amazon Personalize コンソールを使用して、履歴イベントのみをインポート

することをお勧めします。AWS CLI または AWS SDK を使用してイベントをリアルタイムで記録する方法については、「イベントの記録」を参照してください。

Amazon Personalize が新しいレコードのフィルターを更新する方法、および新しいレコードがレコメンデーションにどのように影響するかについては、「個々のレコードのインポート」を参照してください。

トピック

- イベントトラッカーの作成 (コンソール)
- イベントの個別のインポート (コンソール)

イベントトラッカーの作成 (コンソール)

> **Note**
>
> イベントトラッカーを作成した場合は、イベントの個別のインポート (コンソール) にスキップできます。

イベントをインタラクションデータセットにインポートする前に、データセットグループのイベントトラッカーを作成する必要があります。

イベントトラッカーを作成するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] ページで、イベントのインポート先とするアイテムインタラクションデータセットを含むデータセットグループを選択します。
3. データセットグループの [Dashboard] (ダッシュボード) の [Install event ingestion SDK] (イベント取り込み SDK をインストール) で、[Start] (開始) を選択します。
4. [Configure tracker] (トラッカーを設定) のページの [Tracker configurations] (トラッカーの設定) で、[Tracker name] (トラッカー名) にイベントトラッカーの名前を入力し、[Next] (次へ) を選択します。
5. [Install the SDK] (SDKをインストール) のページには、新しいイベントトラッカーの [Tracking ID] (追跡 ID) と、イベントデータをストリーミングするために AWS Amplify または AWS Lambda を使用する手順が表示されます。

Amazon Personalize コンソールを使用してイベントデータをアップロードしているため、この情報は無視してかまいません。AWS Amplify または AWS Lambda を使用して、将来的にイベントデータをストリーミングする場合は、[Event trackers] (イベントトラッカー) のページでイベントトラッカーを選択することで、この情報を表示できます。

6. [終了] を選択します。コンソールを使用してイベントをインポートできるようになりました (「イベントの個別のインポート (コンソール)」を参照してください。または、PutEvents 操作を使用してリアルタイムでイベントを記録します (「イベントの記録」を参照)。

イベントの個別のインポート (コンソール)

イベントトラッカーを作成した後、アイテムインタラクションデータセットにイベントを個別にインポートできます。この手順は、アイテムインタラクションデータセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

イベントを個別にインポートするには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] ページで、イベントのインポート先とするアイテムインタラクションデータセットを含むデータセットグループを選択します。
3. ナビゲーションペインで、[datasets] (データセット) を選択します。
4. [Datasets] (データセット) のページで、Interactions データセットを選択します。
5. データセットの詳細のページの右上で、[Modify dataset] (データセットを変更) を選択し、[Create record] (レコードを作成) を選択します。
6. [Create user-item interaction record(s)] (ユーザーとアイテムのインタラクションレコードを作成) のページの [Record input] (レコードの入力) で、イベントの詳細を JSON 形式で入力します。イベントのフィールドの名前と値は、アイテムインタラクションデータセットを作成したときに使用したスキーマと一致する必要があります。Amazon Personalize は、このスキーマのフィールド名とデータ型を含む JSON テンプレートを提供します。一度に最大 10 個のイベントをインポートできます。
7. [Create record(s)] (レコードを作成) を選択します。[レスポンス] では、インポートの結果が一覧表示され、成功または失敗のメッセージが表示されます。

ユーザーの個別インポート

データセットとスキーマの作成 を完了してユーザーデータセットを作成したら、1 名以上の新しいユーザーをデータセットに個別的にインポートできます。ユーザーを個別インポートすると、カタログの拡大に合わせてユーザーデータセットを小さなバッチインポートで最新の状態に保つことができます。一度に最大 10 名のユーザーをインポートできます。新しいユーザーが大量にある場合は、最初にデータを一括でインポートしてから、必要に応じてユーザーデータを個別にインポートすることをお勧めします。Amazon Personalize データセットへのデータの直接インポート を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用してユーザーをインポートできます。Users データセットに既に存在するユーザーと同じ `userId` でユーザーをインポートすると、Amazon Personalize はそのユーザーを新しいユーザーに置き換えます。一度に最大 10 名のユーザーをインポートできます。

Amazon Personalize が新しいレコードのフィルターを更新する方法、および新しいレコードがレコメンデーションにどのように影響するかについては、「個々のレコードのインポート」を参照してください。

トピック

- ユーザーの個別インポート (コンソール)
- ユーザーを個別にインポートする (AWS CLI)
- ユーザーを個別にインポートする (AWS SDKs)

ユーザーの個別インポート (コンソール)

一度に最大 10 名のユーザーをインポートできます。この手順は、ユーザーデータセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

ユーザーを個別にインポートするには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のページで、ユーザーのインポート先とする Users データセットを含むデータセットグループを選択します。
3. ナビゲーションペインで、[Datasets] (データセット) を選択します。

4. [Datasets] (データセット) のページで、[Users dataset] (ユーザーデータセット) を選択します。
5. データセットの詳細のページの右上にある [Modify dataset] (データセットを変更) を選択し、[Create record] (レコードを作成) を選択します。
6. [Create user record(s)] (ユーザーレコードを作成) のページのレコードの入力で、ユーザーの詳細を JSON 形式で入力します。ユーザーのフィールドの名前と値は、Users データセットを作成したときに使用したスキーマと一致する必要があります。Amazon Personalize は、このスキーマのフィールド名とデータ型を含む JSON テンプレートを提供します。
7. [Create record(s)] (レコードを作成) を選択します。[レスポンス] では、インポートの結果が一覧表示され、成功または失敗のメッセージが表示されます。

ユーザーを個別にインポートする (AWS CLI)

PutUsers 操作を使用して、1 名以上のユーザーを Users データセットに追加します。1 回の PutUsers コールで最大 10 名のユーザーをインポートできます。このセクションでは、Users データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

次の put-users コマンドを使用して、 AWS CLI で 1 名以上のユーザーを追加します。dataset arn をデータセットの Amazon リソースネーム (ARN) に、user Id をユーザーの ID に、それぞれ置き換えます。同じ userId を持つユーザーが既に Users データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。

properties の場合、Users データセットの各フィールドについて、propertyName を、スキーマのフィールド名 (キャメルケース) に置き換えます。例えば、GENDER は gender になり、MEMBERSHIP_TYPE は membershipType になります。user data をユーザーのデータに置き換えます。カテゴリ文字列データについて、単一のプロパティに複数のカテゴリを含めるには、各カテゴリをパイプ (|) で区切ります。例: \"Premium Class|Legacy Member\"。

```
aws personalize-events put-users \
  --dataset-arn dataset arn \
  --users '[{
      "userId": "user Id",
      "properties": "{\"propertyName\": "\user data\"}"
    },
    {
      "userId": "user Id",
      "properties": "{\"propertyName\": "\user data\"}"
    }]'
```

ユーザーを個別にインポートする (AWS SDKs)

PutUsers 操作を使用して、1 名以上のユーザーを Users データセットに追加します。同じ `userId` を持つユーザーが既に Users データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。1 回の `PutUsers` コールで最大 10 名のユーザーをインポートできます。このセクションでは、Users データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

次のコードは、1 名以上のユーザーをユーザーデータセットに追加する方法を示しています。各プロパティ名パラメータについて、スキーマのフィールド名 (キャメルケース) を渡します。例えば、GENDER は `gender` になり、MEMBERSHIP_TYPE は `membershipType` になります。各プロパティ値パラメータについて、ユーザーにデータを渡します。

カテゴリ文字列データについて、単一のプロパティに複数のカテゴリを含めるには、各カテゴリをパイプ (|) で区切ります。例: `"Premium class|Legacy Member"`。

SDK for Python (Boto3)

```
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_users(
    datasetArn = 'dataset arn',
    users = [{
      'userId': 'user ID',
      'properties': "{\"propertyName\": \"user data\"}"
      },
      {
      'userId': 'user ID',
      'properties': "{\"propertyName\": \"user data\"}"
      }]
)
```

SDK for Java 2.x

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String user1Id,
                           String user1PropertyName,
                           String user1PropertyValue,
                           String user2Id,
```

```
                        String user2PropertyName,
                        String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
          .userId(user1Id)
          .properties(String.format("{\"%1$s\": \"%2$s\"}", user1PropertyName,
 user1PropertyValue))
          .build();

        users.add(user1);

        User user2 = User.builder()
          .userId(user2Id)
          .properties(String.format("{\"%1$s\": \"%2$s\"}", user2PropertyName,
 user2PropertyValue))
          .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
          .datasetArn(datasetArn)
          .build();

        responseCode =
 personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

SDK for JavaScript v3

```
import {
  PutUsersCommand,
  PersonalizeEventsClient,
```

```
} from "@aws-sdk/client-personalize-events";

const personalizeEventsClient = new PersonalizeEventsClient({
  region: "REGION",
});

// set the put users parameters
var putUsersParam = {
  datasetArn:
    "DATASET ARN",
  users: [
    {
      userId: "userId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
    {
      userId: "userId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutUsersCommand(putUsersParam)
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

アイテムの個別のインポート

データセットとスキーマの作成 を完了してアイテムデータセットを作成したら、1 つ以上の新しいアイテムをデータセットに個別にインポートできます。アイテムを個別にインポートすると、カタログの拡大に合わせてアイテムデータセットを最新の状態に保つことができます。一度に最大 10 個のアイテムをインポートできます。新しいアイテムが大量にある場合は、最初にデータを一

括でインポートしてから、必要に応じてアイテムデータを個別にインポートすることをお勧めします。Amazon Personalize データセットへのデータの直接インポート を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用してアイテムをインポートできます。Items データセットに既に存在するアイテムと同じ `itemId` でアイテムをインポートすると、Amazon Personalize はそのアイテムを新しいアイテムに置き換えます。

Amazon Personalize が新しいレコードのフィルターを更新する方法、および新しいレコードがレコメンデーションにどのように影響するかについては、「個々のレコードのインポート」を参照してください。

トピック

- アイテムの個別インポート (コンソール)
- アイテムを個別にインポートする (AWS CLI)
- 項目を個別にインポートする (AWS SDKs)

アイテムの個別インポート (コンソール)

一度に最大 10 個のアイテムを Items データセットにインポートできます。この手順は、アイテムデータセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

アイテムを個別にインポートするには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のページで、アイテムのインポート先とする Items データセットを含むデータセットグループを選択します。
3. ナビゲーションペインで、[Datasets] (データセット) を選択します。
4. [Datasets] (データセット) のページで、[Items dataset] (アイテムデータセット) を選択します。
5. データセットの詳細のページの右上で、[Modify dataset] (データセットを変更) を選択し、[Create record] (レコードを作成) を選択します。
6. [Create item record(s)] (アイテムレコードを作成) のページの [Record input] (レコードの入力) で、アイテムの詳細を JSON 形式で入力します。アイテムのフィールドの名前と値は、Items データセットを作成したときに使用したスキーマと一致する必要があります。Amazon

Personalize は、このスキーマのフィールド名とデータ型を含む JSON テンプレートを提供します。

7. [Create record(s)] (レコードを作成) を選択します。[レスポンス] では、インポートの結果が一覧表示され、成功または失敗のメッセージが表示されます。

アイテムを個別にインポートする (AWS CLI)

PutItems 操作を使用して、1 つ以上のアイテムを Items データセットに追加します。1 回の PutItems コールで最大 10 個のアイテムをインポートできます。このセクションでは、Items データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

次の `put-items` コマンドを使用して、 AWS CLI で 1 つ以上のアイテムを追加します。`dataset arn` をデータセットの Amazon リソースネーム (ARN) に、`item Id` をアイテムの ID に、それぞれ置き換えます。同じ `itemId` を持つアイテムが既に Items データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。

`properties` の場合、Items データセットの各フィールドについて、`propertyName` を、スキーマのフィールド名 (キャメルケース) に置き換えます。例えば、GENRES は `genres` になり、CREATION_TIMESTAMP は creationTimestamp になります。`item data` をアイテムのデータに置き換えます。`CREATION_TIMESTAMP` データは Unix エポック時間形式で、かつ、秒単位である必要があります。カテゴリ文字列データについて、単一のプロパティに複数のカテゴリを含めるには、各カテゴリをパイプ (|) で区切ります。例: `\"Horror|Action\"`。

```
aws personalize-events put-items \
  --dataset-arn dataset arn \
  --items '[{
      "itemId": "item Id",
      "properties": "{\"propertyName\": "\item data\"}"
    },
    {
      "itemId": "item Id",
      "properties": "{\"propertyName\": "\item data\"}"
    }]'
```

項目を個別にインポートする (AWS SDKs)

PutItems 操作を使用して、1 つ以上のアイテムを Items データセットに追加します。1 回の PutItems コールで最大 10 個のアイテムをインポートできます。同じ `itemId` を持つアイテムが既

に Items データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。このセクションでは、Items データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

次のコードは、1 つ以上のアイテムをアイテムデータセットに追加する方法を示しています。各プロパティ名パラメータについて、スキーマのフィールド名 (キャメルケース) を渡します。例えば、GENRES は genres になり、CREATION_TIMESTAMP は creationTimestamp になります。各プロパティ値パラメータについて、アイテムのデータを渡します。CREATION_TIMESTAMP データは Unix エポック時間形式で、かつ、秒単位である必要があります。

カテゴリ文字列データについて、単一のプロパティに複数のカテゴリを含めるには、各カテゴリをパイプ (|) で区切ります。例: "Horror|Action"。

SDK for Python (Boto3)

```
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_items(
    datasetArn = 'dataset arn',
    items = [{
      'itemId': 'item ID',
      'properties': "{\"propertyName\": \"item data\"}"
      },
      {
      'itemId': 'item ID',
      'properties': "{\"propertyName\": \"item data\"}"
      }]
)
```

SDK for Java 2.x

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String item1Id,
                           String item1PropertyName,
                           String item1PropertyValue,
                           String item2Id,
                           String item2PropertyName,
                           String item2PropertyValue) {
```

```
    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
                .itemId(item1Id)
                .properties(String.format("{\"%1$s\": \"%2$s\"}",
                        item1PropertyName, item1PropertyValue))
                .build();

        items.add(item1);

        Item item2 = Item.builder()
                .itemId(item2Id)
                .properties(String.format("{\"%1$s\": \"%2$s\"}",
                        item2PropertyName, item2PropertyValue))
                .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
                .datasetArn(datasetArn)
                .items(items)
                .build();

        responseCode =
 personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
    }
```

SDK for JavaScript v3

```
import {
  PutItemsCommand,
  PersonalizeEventsClient,
} from "@aws-sdk/client-personalize-events";
```

```
const personalizeEventsClient = new PersonalizeEventsClient({
  region: "REGION",
});

// set the put items parameters
var putItemsParam = {
  datasetArn:
    "DATASET ARN",
  items: [
    {
      itemId: "itemId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
    {
      itemId: "itemId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutItemsCommand(putItemsParam)
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

アクションの個別のインポート

データセットとスキーマの作成 を完了して Actions データセットを作成したら、1 つ以上の新しいアクションをデータセットに個別にインポートできます。アクションを個別インポートすると、カタログの拡大に合わせて Actions データセットを小さなバッチインポートで最新の状態に保つことができます。一度に最大 10 のアクションをインポートできます。新しいアクションが大量にある場合は、最初にデータを一括でインポートしてから、必要に応じてアクションデータを個別にインポートする

ことをお勧めします。「Amazon Personalize データセットへのデータの直接インポート」を参照してください。

Amazon Personalize コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用してアクションをインポートできます。Actions データセットに既に存在するアクションと同じ `actionId` でアクションをインポートすると、Amazon Personalize はそのアクションを新しいアクションに置き換えます。

新しいレコードがレコメンデーションにどのように影響するかについては、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。

トピック

- アクションの個別インポート (コンソール)
- アクションの個別のインポート (AWS CLI)
- アクションの個別のインポート (AWS SDK)

アクションの個別インポート (コンソール)

一度に最大 10 個のアクションを Actions データセットにインポートできます。このセクションでは、Actions データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

アクションを個別にインポートするには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] ページで、追加する Actions データセットを含むデータセットグループを選択します。
3. ナビゲーションペインで、[データセット] を選択します。
4. [データセット] ページで、Actions データセットを選択します。
5. データセットの詳細のページの右上で、[Modify dataset] (データセットを変更) を選択し、[Create record] (レコードを作成) を選択します。
6. [アクションレコードを作成] ページの [レコードの入力] で、アクションの詳細を JSON 形式で入力します。アクションのフィールドの名前と値は、Actions データセットを作成したときに使用したスキーマと一致する必要があります。Amazon Personalize は、このスキーマのフィールド名とデータ型を含む JSON テンプレートを提供します。

7. [Create record(s)] (レコードを作成) を選択します。[レスポンス] では、インポートの結果が一覧表示され、成功または失敗のメッセージが表示されます。

アクションの個別のインポート (AWS CLI)

PutActions API 操作を使用して、1 つ以上のアクションを Actions データセットに追加します。一度に最大 10 個のアクションをインポートできます。このセクションでは、Actions データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

次の put-actions コマンドを使用して、AWS CLI で 1 つ以上のアクションを追加します。dataset arn をデータセットの Amazon リソースネーム (ARN) に、actionId をアクションの ID に、それぞれ置き換えます。同じ actionId を持つアクションが既に Actions データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。

properties の場合、Actions データセットの各フィールドについて、propertyName を、スキーマのフィールド名 (キャメルケース) に置き換えます。例えば、ACTION_EXPIRATION_TIMESTAMP は actionExpirationTimestamp になり、CREATION_TIMESTAMP は creationTimestamp になります。property data をプロパティのデータに置き換えます。

```
aws personalize-events put-actions \
  --dataset-arn dataset arn \
  --actions '[{
      "actionId": "actionId",
      "properties": "{\"propertyName\": "\property data\"}"
    },
    {
      "actionId": "actionId",
      "properties": "{\"propertyName\": "\property data\"}"
    }]'
```

アクションの個別のインポート (AWS SDK)

PutActions 操作を使用して、1 つ以上のアクションを Actions データセットに追加します。1 回の PutActions 呼び出しで最大 10 のアクションをインポートできます。同じ actionId を持つアクションが既に Actions データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。このセクションでは、Actions データセットが既に作成されていることを前提としています。データセットの作成については、「データセットとスキーマの作成」を参照してください。

次のコードは、1 つ以上のアクションを Actions データセットに追加する方法を示しています。アクションごとに、actionId を指定します。同じ actionId を持つアクションが既に Actions データセットにある場合、Amazon Personalize はそれを新しいものに置き換えます。properties の場合、Actions データセットの各追加フィールドについて、propertyName を、スキーマのフィールド名 (キャメルケース) に置き換えます。例えば、ACTION_EXPIRATION_TIMESTAMP は actionExpirationTimestamp になり、CREATION_TIMESTAMP は creationTimestamp になります。property data をプロパティのデータに置き換えます。

```
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_actions(
    datasetArn = 'dataset arn',
    actions = [{
      'actionId': 'actionId',
      'properties': "{\"propertyName\": \"property value\"}"
      },
      {
      'actionId': 'actionId',
      'properties': "{\"propertyName\": \"property value\"}"
      }]
)
```

ステップ 3: レコメンダーまたはカスタムリソースを作成する

データをインポートしたら、レコメンダーまたはカスタムリソースを作成する準備が整います。次のリソースを使用してレコメンデーションを取得します。作成するリソースは、データセットグループのタイプによって異なります。

- ドメインデータセットグループについては、ドメインに基づいて事前に定義されたユースケース向けのレコメンダーを作成します。アプリケーションでレコメンダーを使用して、レコメンデーションを取得します。使用可能なユースケースについては、「ユースケースの選択」を参照してください。ドメインデータセットグループにカスタムリソースを追加することもできます。これらには、カスタムユースケース向けにトレーニングされたソリューションやソリューションバージョンが含まれます。
- カスタムデータセットグループについては、レシピを使用してソリューションを設定します。次に、ソリューションバージョンの作成 (モデルのトレーニング) を行います。利用可能なレシピについては、「レシピの選択」を参照してください。

リアルタイムのレコメンデーションについては、ソリューションバージョンのソリューションをキャンペーンにデプロイします。バッチレコメンデーションについては、キャンペーンを作成する必要はありません。

トピック

- ドメインレコメンダーの作成
- カスタムリソースの作成

ドメインレコメンダーの作成

データをインポートしたら、ドメインデータセットグループ内のレコメンダーの作成、評価、管理を開始できます。レコメンダーは、レコメンデーションを生成するドメインデータセットグループリソースです。これをアプリケーションで使用して、GetRecommendations 操作に関するリアルタイムのレコメンデーションを取得します。

トピック

- レコメンダーの作成
- レコメンダーを評価する
- レコメンダーの管理

レコメンダーの作成

ドメインデータセットグループを作成したら、ドメインのユースケースのレコメンデーションを作成できます。レコメンダーは、レコメンデーションを生成するドメインデータセットグループリソースです。アプリケーションのレコメンダーを使用して、GetRecommendations 操作でリアルタイムのレコメンデーションを取得します。

レコメンダーを作成する場合、ユースケースを指定します。そして、Amazon Personalize は、ユースケースに最適な設定でレコメンダーをサポートするモデルをトレーニングします。各ユースケースには、レコメンデーションを取得するための異なる API 要件があります。ドメイン別のレコメンダーのユースケースのリストについては、「ユースケースの選択」を参照してください。リージョンごとに最大 15 のレコメンダーを作成できます。

Amazon Personalize は 7 日ごとにレコメンダーをサポートするモデルを自動的に再トレーニングします。これは、データセット内のデータ全体に基づいてまったく新しいモデルを作成する完全な再

トレーニングです。[上位のおすすめ] と [おすすめ商品] のユースケースでは、Amazon Personalize は、既存のモデルを 2 時間ごとに更新し、探索を伴うレコメンデーションに新しいアイテムを含めます。

レコメンダーを作成すると、レコメンデーションのアイテムメタデータを有効にできます。詳細については、「レコメンデーションのメタデータを有効にする」を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用してレコメンダーを作成できます。

レコメンダーのステータス

レコメンダーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

レコメンダーのステータスを取得するには、Amazon Personalize コンソールの [レコメンダー] ページに移動するか、DescribeRecommender 操作を使用します。

トピック

1 秒あたりの最小レコメンデーションリクエスト数と自動スケーリング

> **Important**
>
> `minRecommendationRequestsPerSecond` を高く設定すると請求額が増加します。最初は `minRecommendationRequestsPerSecond` に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、`minRecommendationRequestsPerSecond`必要に応じて を増やします。

レコメンダーを作成する際、レコメンダーの 1 秒あたりの最小レコメンデーション
リクエスト数を設定できます。1 秒あたりの最小レコメンデーションリクエスト数
(minRecommendationRequestsPerSecond) は、Amazon Personalize によってプロビジョニ
ングされるベースラインレコメンデーションリクエストスループットを指定します。デフォル
トは minRecommendationRequestsPerSecond です1。レコメンデーションリクエストは 1 回の
GetRecommendations 操作です。リクエストスループットは 1 秒あたりのリクエスト数で測定さ
れます。Amazon Personalize は 1 秒あたりのリクエスト数を使用して、1 時間あたりのリクエスト
数とレコメンダーの使用量を算出します。

1 秒あたりのリクエスト数が minRecommendationRequestsPerSecond を超えて増加した場
合、Amazon Personalize はプロビジョンド容量を自動的にスケールアップまたはスケールダウンし
ますが、minRecommendationRequestsPerSecond を下回ることはありません。容量が引き上げ
られている間に短時間の遅延が生じます。これにより、リクエストの損失が生じる可能性があり
す。

請求書は、1 時間あたりの最小リクエスト数 (に基づく minRecommendationRequestsPerSecond)
または実際のリクエスト数のいずれか大きい方です。実際に使用されるリクエストのスループッ
トは、5 分間のウィンドウ内の平均リクエスト数/秒として計算されます。デフォルトの から開始
しminRecommendationRequestsPerSecond、Amazon CloudWatch メトリクスを使用して使用
状況を追跡し、minRecommendationRequestsPerSecond必要に応じて を増やすことをお勧めし
ます。

レコメンデーションのメタデータを有効にする

> **Important**
>
> レコメンデーションのメタデータを有効にすると、追加コストが発生します。詳細について
> は、「Amazon Personalize の料金」を参照してください。

レコメンダーを作成すると、アイテムデータセットのアイテムメタデータをレコメンデーション結果
に含めるオプションを有効にできます。有効にすると、レコメンデーションのリクエストでアイテム
データセットの列を指定できます。Amazon Personalize は、レコメンデーションレスポンス内の各
アイテムについてこのデータを返します。

メタデータを使用して、映画のジャンルをカルーセルに追加するなど、ユーザーインターフェイスの
レコメンデーションを充実させることができます。あるいは、レコメンデーションの質を視覚的に評
価するのにも使えます。アプリで生成 AI を使用している場合は、メタデータを AI プロンプトに組み

込んで、より関連性の高いコンテンツを生成できます。Amazon Personalize の生成 AI の使用に関する詳細については、「Amazon Personalize と生成 AI」を参照してください。

- Amazon Personalize コンソールでメタデータを有効にするには、レコメンダーを作成するときに、[詳細設定] の [レコメンデーション結果にあるアイテムのメタデータを返す] を選択します。詳細については、「レコメンダーの作成 (コンソール)」を参照してください。
- AWS SDKs または でメタデータを有効にするには AWS CLI、 CreateRecommender API オペレーション および を `recommenderConfigenableMetadataWithRecommendations`に設定します`true`。詳細については、レコメンダーの作成 (AWS CLI)またはレコメンダーの作成 (AWS SDKs)を参照してください。

レコメンデーションにメタデータを追加するには、メタデータの列を含むアイテムデータセットが必要です。トレーニングではメタデータを使用する必要はありません。データセットの作成については、「データセットとスキーマの作成」を参照してください。データの管理と更新については、「データセット内のトレーニングデータの管理」を参照してください。

トレーニング時に使用する列の設定

レコメンダーを作成するときに、レコメンダーをサポートするモデルをトレーニングするときに Amazon Personalize が考慮する列を変更できます。

これにより、トレーニングデータのさまざまな組み合わせを試してみることができます。または、意味のあるデータがない列を除外することもできます。例えば、レコメンデーションをフィルタリングするためだけに使用したい列があるとします。この列をトレーニングから除外すると、Amazon Personalize はフィルタリング時にのみこの列を考慮します。

EVENT_TYPE 列を除外することはできません。デフォルトでは、Amazon Personalize はトレーニング時に使用できるすべての列を使用します。次のデータは、常にトレーニングから除外されます。

- ブールデータ型の列
- インプレッションデータ
- カテゴリ別またはテキスト別ではないカスタム文字列フィールド

トレーニングにインプレッションデータを含めることはできませんが、ユースケースやレシピでそのデータが使用されている場合、Amazon Personalize はインプレッションデータを使用してレコメンデーションを取得するときに探索をガイドします。

- Amazon Personalize コンソールでトレーニングするときに使用する列を設定するには、レコメンダーを作成するときに、詳細設定ページで使用する列を選択します。詳細については、「レコメンダーの作成 (コンソール)」を参照してください。
- AWS SDKs または でトレーニングするときに使用する列を設定するには AWS CLI、CreateRecommender API オペレーションを使用し、 の で `excludedDatasetColumns` `recommenderConfig`を指定しますtrainingDataConfig。コードサンプルについては、「トレーニング時に使用する列の設定 (AWS CLI) 」または「トレーニング時に使用する列の設定 (AWS SDKs」を参照してください。

レコメンダーの作成 (コンソール)

> **Important**
>
> `minRecommendationRequestsPerSecond` を高く設定すると請求額が増加します。最初は `minRecommendationRequestsPerSecond` に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、`minRecommendationRequestsPerSecond`必要に応じて を増やします。詳細については、1 秒あたりの最小レコメンデーションリクエスト数と自動スケーリングを参照してください。

次のように、Amazon Personalize コンソールを使用して各ユースケース向けのレコメンダーを作成します。ドメインデータセットグループを作成したばかりで、既に [Overview] (概要) のページが表示されている場合は、ステップ 3 に進みます。

レコメンダーを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、ドメインデータセットグループを選択します。
3. ステップ 3 で、[<domain name> レコメンダーを使用する] を選択し、[レコメンダーの作成] を選択します。
4. [ユースケースの選択] のページで、レコメンダー入を作成するユースケースを選択し、それぞれに [レコメンダー名] を入力します。Amazon Personalize は、選択した各ユースケースについてレコメンダーを作成します。使用可能なユースケースは、ドメインによって異なります。ユースケースの選択については、「ユースケースの選択」を参照してください。

5. [次へ] をクリックします。
6. [詳細設定] のページでは、ビジネスニーズに応じて各レコメンダーを設定します。

   - レコメンダーのユースケースで使用されるデータセットごとに、レコメンダーを裏付けるモデルをトレーニングするときに Amazon Personalize が考慮する列を選択できます。デフォルトでは、Amazon Personalize はトレーニング時に使用できるすべての列を使用します。詳細については、「トレーニング時に使用する列の設定」を参照してください。
   - [1 秒あたりの最小レコメンデーションリクエスト数] を変更して、レコメンダーの新しい最小リクエスト容量を指定できます。`minRecommendationRequestsPerSecond` を高く設定すると請求額が増加します。最初は 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、`minRecommendationRequestsPerSecond`必要に応じて を増やします。詳細については、1 秒あたりの最小レコメンデーションリクエスト数と自動スケーリングを参照してください。
   - アイテムデータセットのメタデータをレコメンデーションに含める場合は、[レコメンデーション結果にアイテムメタデータを返す] を選択します。有効にすると、レコメンデーションのリクエストやパーソナライズされたランキングで、アイテムデータセットから列を指定できます。Amazon Personalize は、レコメンデーションレスポンス内の各アイテムについてこのデータを返します。

     メタデータを有効にするには、メタデータの列を含むアイテムデータセットが必要です。
   - `Top picks for you` または `Recommended for you` のユースケースでは、オプションで探索設定を変更します。探索は、インタラクションデータがないかほとんどない状態で、ユーザーが商品にどのように反応するかを学習するための、さまざまな商品のレコメンデーションをテストに関連します。次のフィールドを使用して、探索を設定します。
     - 関連性の低いアイテムの探索に重点を置く (探索の重み) — 探索する範囲を設定します。0 ~ 1 の小数値を指定します。デフォルトは 0.3 です。値が 1 に近くなるほど、探索が多くなります。探索が増えると、レコメンデーションにはより多くのアイテムが含まれますが、アイテムインタラクションデータや以前の行動に基づく関連性が少なくなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
     - 探索アイテムが存在するようになってからの期間のカットオフ - アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は作成タイムスタンプを基に、あるいは作成タイムスタンプデータがない場合はアイテムインタラクションデータを基に、アイテムが存在するようになってからの期間を決定します。Amazon

Personalize がアイテムが存在するようになってからの期間を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。

Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。その理由は、これらの商品がユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったためです。

- [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けについての詳細は、「Amazon Personalize リソースのタグ付け」を参照してください。

7. 各ユースケースのレコメンダーを作成するには、[レコメンダーの作成] を選択します。

   各レコメンダーのステータスは、[Recommenders] (レコメンダー) のページでモニタリングできます。レコメンダーのステータスが Active の場合、アプリケーションでそれを使用してレコメンデーションを取得できます。

レコメンダーの作成 (AWS CLI)

ドメインデータセットグループを作成したら、ドメインのユースケースのレコメンデーションを作成できます。レコメンダーは、レコメンデーションを生成するドメインデータセットグループリソースです。

`Top picks for you` または `Recommended for you` のユースケースでは、Amazon Personalize は商品をレコメンデーションする際に探索を使用します。詳細については、「探索の設定」を参照してください。

トピック

- レコメンダーの作成
- トレーニング時に使用する列の設定
- 探索の設定
- レコメンデーションのメタデータを有効にする

レコメンダーの作成

次の AWS CLI コードを使用して、ドメインユースケースのレコメンダーを作成します。このコードは、各ドメインユースケースのために実行します。`recipeArn` で、ユースケースの Amazon リソー

スネーム (ARN) を指定します。使用可能なユースケースは、ドメインによって異なります。ユースケースとその ARN のリストについては、「ユースケースの選択」を参照してください。

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN
```

トレーニング時に使用する列の設定

列をトレーニングから除外するには、レコメンダー設定の一部として trainingDataConfig で excludedDatasetColumns オブジェクトを指定します。オブジェクト内の各キーについて、データセットタイプを指定します。値ごとに、除外する列のリストを指定します。詳細については、「トレーニング時に使用する列の設定」を参照してください。

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--recommender-config "{\"trainingDataConfig\": {\"excludedDatasetColumns\":
 { \"datasetType\" : [ \"column1Name\", \"column2Name\"]}}}"
```

探索の設定

Top picks for you または Recommended for you のユースケースでは、Amazon Personalize は商品をレコメンデーションする際に探索を使用します。探索は、インタラクションデータがないかほとんどない状態で、ユーザーが商品にどのように反応するかを学習するための、さまざまな商品のレコメンデーションをテストに関連します。探索は以下のように設定できます。

- 関連性の低いアイテムの探索に重点を置く (探索の重み) — 探索する範囲を設定します。0 ~ 1 の小数値を指定します。デフォルトは 0.3 です。値が 1 に近くなるほど、探索が多くなります。探索が増えると、レコメンデーションにはより多くのアイテムが含まれますが、アイテムインタラクションデータや以前の行動に基づく関連性が少なくなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
- 探索アイテムが存在するようになってからの期間のカットオフ - アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は作成タイムスタンプを基に、あるいは作成タイムスタンプデータがない場合はアイテムインタラクションデータを基に、アイテムが存在する

ようになってからの期間を決定します。Amazon Personalize がアイテムが存在するようになってからの期間を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。

Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。その理由は、これらの商品がユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったためです。

次のコードは、Top picks for you のユースケースのレコメンダーを作成する際に探索を設定する方法を示しています。この例では、デフォルトの値を使用します。

アイテムデータセットがあり、レコメンデーションを取得したときにメタデータを含めるオプションが必要な場合は、recommender-config を更新して enableMetadataWithRecommendations フィールドを追加し、true に設定します。

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group ARN \
--recipe-arn arn:aws:personalize:::recipe/aws-vod-top-picks \
--recommender-config "{\"itemExplorationConfig\":{\"explorationWeight\":\"0.3\",
\"explorationItemAgeCutOff\":\"30\"}}"
```

レコメンデーションのメタデータを有効にする

アイテムデータセットがあり、レコメンデーションを取得したときにメタデータを含めるオプションが必要な場合は、recommender-config で enableMetadataWithRecommendations を true に設定します。

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group \
--recipe-arn recipe ARN \
--recommender-config "{\"enableMetadataWithRecommendations\": "true"}"
```

レコメンダーの作成 (AWS SDKs)

ドメインデータセットグループを作成したら、ドメインのユースケースのレコメンデーションを作成できます。レコメンダーは、レコメンデーションを生成するドメインデータセットグループリソースです。

すべてのユースケースでトレーニング時に使用する列を設定できます。詳細については、「トレーニング時に使用する列の設定」を参照してください。Top picks for you または Recommended for you のユースケースでは、Amazon Personalize は商品をレコメンデーションする際に探索を使用します。詳細については、「探索の設定」を参照してください。

トピック

- レコメンダーの作成
- 探索の設定
- トレーニング時に使用する列の設定
- メタデータの有効化

レコメンダーの作成

次のコードを使用して、ドメインのユースケースのレコメンダーを作成します。レコメンデーションに名前を付け、ドメインデータセットグループの Amazon リソースネーム (ARN) を入力します。recipeArn でユースケースの ARN を指定します。このコードは、各ドメインユースケースのために実行します。使用可能なユースケースは、ドメインによって異なります。ユースケース、その ARN および要件のリストについては、「ユースケースの選択」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN'
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

SDK for Java 2.x

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
```

```
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
```

```
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the recommender's parameters
export const createRecommenderParam = {
  name: "RECOMMENDER_NAME",                    /* required */
  recipeArn: "RECIPE_ARN",                     /* required */
  datasetGroupArn: "DATASET_GROUP_ARN"         /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

探索の設定

`Top picks for your` または `Recommended for you` のユースケースでは、Amazon Personalize は商品をレコメンデーションする際に探索を使用します。探索は、インタラクションデータがないかほとんどない状態で、ユーザーが商品にどのように反応するかを学習するための、さまざまな商品のレコメンデーションをテストに関連します。探索は以下のように設定できます。

- 関連性の低いアイテムの探索に重点を置く (探索の重み) — 探索する範囲を設定します。0 ~ 1 の小数値を指定します。デフォルトは 0.3 です。値が 1 に近くなるほど、探索が多くなります。探索が増えると、レコメンデーションにはより多くのアイテムが含まれますが、アイテムインタラクションデータや以前の行動に基づく関連性が少なくなります。ゼロでは、探索は行われず、レコメンデーションは現在のデータに基づきます (関連性)。
- 探索アイテムが存在するようになってからの期間のカットオフ - アイテムインタラクションデータセット内のすべてのアイテムについて、最新のインタラクションからの日数で、アイテムが存在するようになってからの最長期間を指定します。これにより、アイテムの経過時間に基づいてアイテム探索の範囲が定義されます。Amazon Personalize は作成タイムスタンプを基に、あるいは作成タイムスタンプデータがない場合はアイテムインタラクションデータを基に、アイテムが存在するようになってからの期間を決定します。Amazon Personalize がアイテムが存在するようになってからの期間を決定する方法の詳細については、「作成のタイムスタンプデータ」を参照してください。

  Amazon Personalize が探索中に考慮するアイテムの数を増やすには、より大きな値を入力します。デフォルトは 30 日 で、最短は 1 日 です。レコメンデーションには、指定したアイテムの期間制限より古いアイテムが含まれる場合があります。その理由は、これらの商品がユーザーの興味に関連しており、それらを特定するために探索が必要ではなかったためです。

次のコードは、レコメンダーを作成する際に探索を構成する方法を示しています。この例では、デフォルトの値を使用します。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'arn:aws:personalize:::recipe/aws-vod-top-picks',
  datasetGroupArn = 'dataset group ARN',
```

```
  recommenderConfig = {"itemExplorationConfig": {"explorationWeight": "0.3",
 "explorationItemAgeCutOff": "30"}}
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the recommender's parameters
export const createRecommenderParam = {
  name: "RECOMMENDER_NAME",                    /* required */
  recipeArn: "RECIPE_ARN",                     /* required */
  datasetGroupArn: "DATASET_GROUP_ARN",        /* required */
  recommenderConfig: {
    itemExplorationConfig: {
      explorationWeight: "0.3",
      explorationItemAgeCutOff: "30"
    }
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

トレーニング時に使用する列の設定

列をトレーニングから除外するには、レコメンダー設定の一部として trainingDataConfig で excludedDatasetColumns オブジェクトを指定します。キーごとに、データセットタイプを指定します。値ごとに、除外する列のリストを指定します。次のコードは、レコメンダーを作成する際にトレーニングから列を除外する方法を示しています。詳細については、「トレーニング時に使用する列の設定」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'recipe name',
  datasetGroupArn = 'dataset group ARN',
  recommenderConfig = {
    "trainingDataConfig": {
      "excludedDatasetColumns": {
        "datasetType": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});
```

```
// set the recommender's parameters
export const createRecommenderParam = {
  name: "RECOMMENDER_NAME",             /* required */
  recipeArn: "RECIPE_ARN",              /* required */
  datasetGroupArn: "DATASET_GROUP_ARN", /* required */
  recommenderConfig: {
    trainingDataConfig: {
      excludedDatasetColumns: {
        "DATASET_TYPE": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

メタデータの有効化

アイテムデータセットがあり、レコメンデーションを取得したときにメタデータを含めるオプションが必要な場合は、recommender-config で enableMetadataWithRecommendations を true に設定します。

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'recipe name',
  datasetGroupArn = 'dataset group ARN',
  recommenderConfig = {"enableMetadataWithRecommendations": True}
)
```

```
recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

レコメンダーを評価する

オフラインおよびオンラインのメトリクスを使用して、ソリューションバージョンのパフォーマンスを評価できます。オンラインメトリクスは、リアルタイムのレコメンデーションを使用して、ユーザーのインタラクションで観察される経験的な結果です。例えば、ユーザーがカタログを閲覧するときのユーザーのクリック率を記録できます。オンラインメトリクスの生成および記録は、ユーザーの責任において行われます。

オフラインメトリクスは、レコメンダーを作成するときに Amazon Personalize が生成するメトリクスです。オフラインメトリクスを使用して、レコメンダーの基礎となるモデルのパフォーマンスを評価できます。オフラインメトリクスでは、同じデータでトレーニングされた他のモデルとモデルを比較できます。このセクションの残りの部分では、メトリクスという用語はオフラインメトリクスを意味します。

パフォーマンスメトリクスを取得するために、Amazon Personalize は、入力インタラクションデータをトレーニングセットとテストセットの 2 つのセットに分割します。トレーニングセットは、ユーザーとそのインタラクションデータの 90% で設定されています。テストセットは、ユーザーとそのインタラクションデータの残りの 10% で設定されています。

その後、Amazon Personalize は、トレーニングセットを使用してレコメンダーを作成します。トレーニングが完了すると、Amazon Personalize は、テストセットからの各ユーザーのデータの最も古い 90% を入力として提供します。その後、Amazon Personalize は、レコメンダーが生成するレコメンデーションを、テストセットからの各ユーザーのデータの最新の 10% における実際のインタラクションと比較することにより、メトリクスを計算します。

トピック

- メトリクスの取得
- メトリクスの定義
- 例
- その他のリソース

メトリクスの取得

レコメンダーがアクティブになったら、Amazon Personalize コンソールでレコメンダーのメトリックスを表示したり、DescribeRecommender オペレーションを呼び出してメトリックスを取得したりできます。

トピック

- メトリクスの表示 (コンソール)
- メトリクスの取得 (AWS CLI)
- メトリクス (AWS SDK) の取得

メトリクスの表示 (コンソール)

レコメンダーメトリクスをコンソールに表示するには、レコメンダーの詳細ページに移動します。

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のページで、ドメインデータセットグループを選択します。
3. ナビゲーションペインから、[レコメンダー] を選択します。
4. レコメンダーのリストから、メトリクスを表示するレコメンダーを選択します。

メトリクスの取得 (AWS CLI)

次のコードは、AWS CLI を使用してレコメンダー向けのメトリクスを取得する方法を示しています。

```
aws personalize describe-recommender \
--recommender-arn recommender arn
```

VIDEO_ON_DEMAND ドメインの Top picks for You ユースケース用に作成されたレコメンダーからのメトリクスを出力する例を次に示します。

```
{
    "recommender": {
        "recommenderArn": "arn:aws:personalize:region:acct-id:recommender/
recommenderName",
```

```
        "datasetGroupArn": "arn:aws:personalize:region:acct-id:dataset-group/
dsGroupName",
        "name": "name123",
        "recipeArn": "arn:aws:personalize:::recipe/aws-vod-top-picks",
        "modelMetrics": {
            "coverage": 0.27,
            "mean_reciprocal_rank_at_25": 0.0379,
            "normalized_discounted_cumulative_gain_at_5": 0.0405,
            "normalized_discounted_cumulative_gain_at_10": 0.0513,
            "normalized_discounted_cumulative_gain_at_25": 0.0828,
            "precision_at_5": 0.0136,
            "precision_at_10": 0.0102,
            "precision_at_25": 0.0091,
        }
        "recommenderConfig": {},
        "creationDateTime": "2022-05-06T10:11:24.589000-07:00",
        "lastUpdatedDateTime": "2022-05-06T10:34:33.270000-07:00",
        "status": "ACTIVE",
    }
}
```

メトリクス (AWS SDK) の取得

次のコードは、SDK for Python (Boto3) を使用してレコメンダー用にメトリクスを取得する方法を示しています。

```
import boto3

personalize = boto3.client('personalize')

response = personalize.describe_recommender(
    recommenderArn = 'recommender_arn'
)
print(response['recommender']['modelMetrics'])
```

VIDEO_ON_DEMAND ドメイン用の Top picks for you ユースケース用に作成されたレコメンダーからのメトリクスを出力する例を次に示します。

```
{
    "recommender": {
        "recommenderArn": "arn:aws:personalize:region:acct-id:recommender/
recommenderName",
```

```
        "datasetGroupArn": "arn:aws:personalize:region:acct-id:dataset-group/
dsGroupName",
        "name": "name123",
        "recipeArn": "arn:aws:personalize:::recipe/aws-vod-top-picks",
        "modelMetrics": {
            "coverage": 0.27,
            "mean_reciprocal_rank_at_25": 0.0379,
            "normalized_discounted_cumulative_gain_at_5": 0.0405,
            "normalized_discounted_cumulative_gain_at_10": 0.0513,
            "normalized_discounted_cumulative_gain_at_25": 0.0828,
            "precision_at_5": 0.0136,
            "precision_at_10": 0.0102,
            "precision_at_25": 0.0091,
        }
        "recommenderConfig": {},
        "creationDateTime": "2022-05-06T10:11:24.589000-07:00",
        "lastUpdatedDateTime": "2022-05-06T10:34:33.270000-07:00",
        "status": "ACTIVE",
    }
}
```


メトリクスの定義

Amazon Personalize がレコメンダー向けに生成するメトリックスは、以下の用語を使用して説明されています。

- 関連レコメンデーションとは、ユーザーが実際にインタラクションを行ったアイテムのレコメンデーションです。これらのアイテムは、テストセットからの各ユーザーのインタラクションデータの最新の 10% からのものです。
- ランク は、レコメンデーションリスト内の推奨される事項の順位を示しています。順位 1 (リストの上位) は、ユーザーにとって最も高い関連事項であると推定されます。

各メトリクスでは、数字が大きい (1 に近い) ほど優良です。さらに詳しく調べるには、「その他のリソース」に記載されているリソースを参照してください。

カバレッジ

カバレッジの値は、インタラクションデータセットとアイテムデータセットに含まれる一意のアイテムの総数のうち、Amazon Personalize がお勧めする一意のアイテムの割合を示しています。カバレッジスコアが高いほど、Amazon Personalize は、異なるユーザーに同じ少数のアイテムを繰り返し推奨するのではなく、より多くのアイテムを推奨します。Top picks for you

(VIDEO_ON_DEMAND) や、おすすめ (eコマース) など、アイテム探索を特徴とするユースケースは、そうでないユースケースよりもカバー率が高くなります。

25 での平均逆ランク

このメトリクスは、あるモデルがランキングの上位で関連するレコメンデーションを生成できるかどうかを示します。あるユーザーに関連する検索結果を生成していて、ユーザーがリストの下位にある項目を選択することを期待していない場合、25 での平均逆ランクが高いモデルを選択できます。例えば、ユーザーは検索結果の最初の料理レシピを選ぶことがよくあります。

Amazon Personalize は、レコメンデーションのリクエストの平均相互ランクスコアを使用してこのメトリックスを計算します。それぞれの相互ランクスコアは次のように計算されます。`1 / the rank of the highest item interacted with by the user`、ここで、可能なランキングの合計は 25 です。ユーザーが操作するその他のランクの低いアイテムは無視されます。ユーザーが最初のアイテムを選択した場合、スコアは 1 です。アイテムを何も選択しなかった場合、スコアは 0 です。

例えば、3 人の異なるユーザーに、それぞれ 25 件のレコメンデーションを表示するとします。

- ユーザー 1 がランク 4 のアイテムとランク 10 のアイテムをクリックした場合、両者の相互ランクスコアは 1/4 になります。
- ユーザー 2 がランク 2 のアイテム、ランク 4 のアイテム、ランク 12 のアイテムをクリックした場合、相互のランクスコアは 1/2 になります。
- ユーザー 3 がランク 6 のアイテムを 1 つクリックした場合、相互のランクスコアは 1/6 になります。

すべてのレコメンデーションのリクエスト (この場合は 3) の平均逆ランクは、`(1/4 + 1/2 + 1/6) / 3 = .3056` として計算されます。

K (5、10、または 25) での正規化減損累積利得 (NCDG)

このメトリクスは、モデルがどの程度うまくレコメンデーションをランク付けしているかを教えてくれます。ここで、K は 5、10、25 のレコメンデーションのサンプルサイズです。このメトリクスは、最もランクの高いアイテムだけでなく、レコメンデーションのランキングにもっとも関心がある場合に役立ちます (これについては、「`mean reciprocal rank at 25`」を参照してください)。例えば、1 つのカルーセルに同時に最大 10 本の映画を表示するアプリケーションでは、`NDCG at 10` のスコアが役に立ちます。

Amazon Personalize は、テストセット内の各ユーザーのランキング順位に基づいてレコメンデーションに重みを付けることで NDCG を計算します。各レコメンデーションはそれぞれの順位に

よる要因で割引 (低い分量の付与) されます。最後のメトリックスは、テストセットに含まれる全ユーザーの平均です。K での正規化された割引累積利益は、リストの下位にあるレコメンデーションが、リストの上位にあるレコメンデーションよりも関連性が低いことを想定しています。

Amazon Personalize は、リストの上位が順位 1 である `1/log(1 + position)` の重量要素を使用します。

precision at K

K (5、10、25) の推奨サンプルサイズに基づいて、モデルのレコメンデーションがどの程度関連しているかを示すメトリクス。

Amazon Personalize では、このメトリクスは、上位 K 個のテストセットでの各ユーザーのレコメンデーションのうち、関連するレコメンデーションの数を K で割った数に基づいて計算されます (K は 5、10、または 25 です)。最後のメトリクスは、テストセットに含まれる全ユーザーの平均です。

例えば、あるユーザーに 10 個のアイテムをレコメンドし、ユーザーがそのうちの 3 個とインタラクションした場合、K での精度は、正しく予測された 3 個のアイテムを、合計 10 個のレコメンデーションアイテムで割った値、つまり `3 / 10 = .30` です。

このメトリクスでは、関連事項の正確なレコメンデーションが評されます。スコアが 1 に近いほど、モデルの精度は高くなります。

例

以下は、特定のユーザー向けのレコメンデーションのリストを作成するレコメンダーの簡単な例です。2 番目と 5 番目のレコメンデーションは、このユーザーのテストデータのレコードと一致します。これらは関連するレコメンデーションです。K が 5 に設定されている場合、次のメトリクスがユーザーに生成されます。

reciprocal_rank

計算: 1/2

結果: 0.5000

normalized_discounted_cumulative_gain_at_5

計算: (1/log(1 + 2) + 1/log(1 + 5)) / (1/log(1 + 1) + 1/log(1 + 2))

結果: 0.6241

precision_at_5

計算: 2/5

結果: 0.4000

その他のリソース

レコメンダーシステムのさまざまなメトリクスについて詳しくは、以下の外部リソースを参照してください。

- MRR と MAP と NDCG の比較:ランクに応じた評価メトリクスとそれをいつ使うべきか
- 割引後の累積利益:知っておくべきランキングメトリクス
- レコメンダーシステムの k での再現率と精度
- レコメンダーシステムのランク付け評価メトリクス

レコメンダーの管理

レコメンダーをサポートするモデルを管理する必要はありません。Amazon Personalize は 7 日ごとに自動的に再トレーニングします。これは、データセット内のデータ全体に基づいてまったく新しいモデルを作成する完全な再トレーニングです。[上位のおすすとおすすめのでは、Amazon Personalize は、既存のモデルを 2 時間ごとに更新し、探索を伴うレコメンデーションに新しいアイテムを考慮します。詳細については、「自動更新」を参照してください。

レコメンダーの管理には、次の操作が含まれます。

- レコメンダーの停止と開始 — アクティブなレコメンダーの請求を一時停止したい場合は、レコメンダーを停止して後で再開できます。詳細については、「レコメンダーの停止と開始」を参照してください。
- レコメンダー設定の更新 — レコメンダーがトレーニングで使用する列を更新したり、レコメンダーのリクエスト容量を更新したりできます。詳細については、「レコメンダーの更新」を参照してください。
- レコメンダーの削除 — レコメンダーは DeleteRecommender 操作で削除できます。または、Amazon Personalize コンソールのレコメンダー詳細ページからレコメンダーを削除できます。

トピック

- レコメンダーの更新
- レコメンダーの停止と開始

レコメンダーの更新

レコメンダーを作成したら、レコメンダーの設定を更新できます。

- レコメンダーがトレーニングで使用する列を更新できます。トレーニング時に使用する列を変更すると、Amazon Personalize はレコメンダーを裏付けるモデルの完全な再トレーニングを自動的に開始します。更新が完了しても、レコメンダーからレコメンデーションを受けることができます。レコメンダーは、更新が完了するまで以前の設定を使用します。この更新のステータスを追跡するには、DescribeRecommender 操作で返された `latestRecommenderUpdate` を使用します。レコメンダーを作成したときと同じ列を指定しても、更新は行われません。
- レコメンダーの 1 秒あたりの最小推奨リクエスト数は更新できます。これにより、Amazon Personalize によってプロビジョニングされるベースラインのレコメンデーションリクエストスループットが指定されます。値を高く設定すると請求額が増加します。1 から始めることをお勧めします。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、必要に応じて増やします。詳細については、「1 秒あたりの最小レコメンデーションリクエスト数と自動スケーリング」を参照してください。
- 上位のおすすめやおすすめユースケースについては、関連項目の探索に重点を置くように調整し、探索項目の有効期限を調整することで探索設定を更新できます。探索について詳しくは、「ユースケースの選択」のユースケースのセクションを参照してください。

Amazon Personalize コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して、レコメンデーションを更新できます。

トピック

- レコメンダーの更新 (Amazon Personalize コンソール)
- レコメンダーの更新 (AWS CLI)
- レコメンダーの更新 (AWS SDK)

レコメンダーの更新 (Amazon Personalize コンソール)

レコメンダーの作成後、レコメンダーを更新できます。レコメンダーがトレーニングで使用する列と、レコメンダーの 1 秒あたりの最小推奨リクエスト数を更新できます。上位のおすすめとおすす

めのユースケースについては、探索設定を更新できます。コンソールでレコメンダーを更新するには、次の操作を実行します。

レコメンダーの設定を更新するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、ドメインデータセットグループを選択します。
3. ナビゲーションペインから、[レコメンダー] を選択します。
4. [レコメンダー] のページで、更新するレコメンダーを選択します。
5. [レコメンダー設定] で、[編集] を選択します。
6. レコメンダーの設定に変更を加えて、[更新] を選択します。さまざまな設定オプションについては、「レコメンダーの作成 (コンソール)」を参照してください。

レコメンダーの更新 (AWS CLI)

レコメンダーを AWS CLI で更新するには、update-recommender コマンドを使用します。レコメンダーと更新済みの設定に対して Amazon リソースネーム (ARN) を指定します。次のコードは、レコメンダーがトレーニングに使用する列を更新する方法を示しています。

```
aws personalize update-recommender \
--dataset-group-arn dataset group ARN \
--recommender-config "{\"trainingDataConfig\": {\"excludedDatasetColumns\":
 { \"datasetType\" : [ \"column1Name\", \"column2Name\"]}}}"
```

トレーニングに使用した列を変更すると、Amazon Personalize はレコメンダーを裏付けるモデルの完全な再トレーニングを自動的に開始します。更新が完了しても、レコメンダーからレコメンデーションを取得することができます。レコメンダーは、更新が完了するまで以前の設定を使用します。この更新のステータスを追跡するには、DescribeRecommender 操作で返された latestRecommenderUpdate を使用します。

変更できるさまざまな設定についての詳細は、「RecommenderConfig」を参照してください。

レコメンダーの更新 (AWS SDK)

レコメンダーを AWS で更新するには、UpdateRecommender 操作を使用します。レコメンダーの Amazon リソースネーム (ARN) を指定し、新しい設定を指定します。次のコードは、レコメンダーがトレーニングに使用する列を更新する方法を示しています。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

update_recommender_response = personalize.update_recommender(
  recommenderArn = 'dataset group ARN',
  recommenderConfig = {
    "trainingDataConfig": {
      "excludedDatasetColumns": {
        "datasetType": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
)
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { UpdateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the request's parameters
export const updateRecommenderParam = {
  recommenderArn: "RECOMMENDER_ARN", /* required */
  recommenderConfig: {
    trainingDataConfig: {
      excludedDatasetColumns: {
        "DATASET_TYPE": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
};

export const run = async () => {
  try {
```

```
    const response = await personalizeClient.send(new
 UpdateRecommenderCommand(updateRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

recommenderConfig の excludedDatasetColumns でトレーニングに使用されている列を変更すると、Amazon Personalize はレコメンダーを裏付けるモデルの完全な再トレーニングを自動的に開始します。更新が完了しても、レコメンダーからレコメンデーションを取得することができます。レコメンダーは、更新が完了するまで以前の設定を使用します。この更新のステータスを追跡するには、DescribeRecommender 操作で返された latestRecommenderUpdate を使用します。

変更できるさまざまな設定についての詳細は、「RecommenderConfig」を参照してください。

レコメンダーの停止と開始

レコメンダーがアクティブになったら、レコメンダーを停止して後で開始できます。こうすることで、レコメンダーへの請求を一時停止して、使用したときだけ料金を支払うことができます。例えば、特定の曜日にのみレコメンデーションを取得する必要がある場合があります。必要のない日にレコメンダーを停止し、必要な日にレコメンダーを開始できます。

レコメンダーを停止すると、レコメンダーを使用してレコメンデーションを取得することはできません。レコメンダーを停止すると、レコメンダーへの請求と再トレーニングは停止されます。ただし、レコメンダーを停止してもレコメンダーは削除されません。いつでも再起動して、レコメンデーションの取得を再開できます。レコメンダーを開始しても、データを使用して新しいレコメンダーは作成されません。そうではなく、レコメンダーへの請求と再トレーニングが 7 日ごとに再開されます。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、 AWS SDKs オペレーションを使用して、レコメンダーを停止StartRecommenderおよび開始できます。StopRecommender

レコメンダーの状態

レコメンダーを停止すると、レコメンダーの状態は次の順序でアクティブから非アクティブに変わります。

アクティブ > 保留中止 > 進行中停止 > 非アクティブ

レコメンダーを起動すると、レコメンダーの状態は次の順序でインアクティブからアクティブに変わります。

非アクティブ > 保留中開始 > 進行中 > アクティブ

トピック

- レコメンダーの停止および開始 (コンソール)
- レコメンダーの停止と再開 (AWS CLI)
- レコメンダーの停止と再起動 (AWS SDKs)

レコメンダーの停止および開始 (コンソール)

Amazon Personalize を使用して、レコメンダーの停止および再開ができます。

トピック

- レコメンダーの停止 (コンソール)
- レコメンダーを起動する (コンソール)

レコメンダーの停止 (コンソール)

Amazon Personalize コンソールを使用してアクティブなレコメンダーを停止するには、次のようにします。

レコメンダーを停止するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、ドメインデータセットグループを選択します。
3. ナビゲーションペインから、[レコメンダー] を選択します。
4. [レコメンダー] のページで、更新するレコメンダーを選択します。
5. 右上の [レコメンダーを停止] を選択し、表示されるウィンドウで確認します。

   レコメンダーのステータスが非アクティブになると、レコメンダーは停止しています。これにより、レコメンダーへの請求と再トレーニングは停止されます。レコメンダーは起動するまで使用できません。

レコメンダーを起動する (コンソール)

Amazon Personalize コンソールを使用して、次のように非アクティブなレコメンダーを開始できます。

レコメンダーを開始するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、ドメインデータセットグループを選択します。
3. ナビゲーションペインから、[レコメンダー] を選択します。
4. [レコメンダー] のページで、更新するレコメンダーを選択します。
5. 右上の [レコメンダーを開始] を選択し、表示されるウィンドウでレコメンダーを開始することを確認します。

   レコメンダーの状態がアクティブになったら、レコメンダーからのレコメンデーションの取得を再開できます。レコメンダーへの請求と自動再トレーニングも再開されます。

レコメンダーの停止と再開 (AWS CLI)

でアクティブなレコメンダーを停止するには AWS CLI、 `stop-recommender` コマンドを使用して、次のようにレコメンダーの Amazon リソースネーム (ARN) を指定します。

```
aws personalize stop-recommender --recommender-arn "recommender arn"
```

で非アクティブなレコメンダーを開始するには AWS CLI、 `start-recommender` コマンドを使用して、停止したレコメンダーの ARN を次のように指定します。

```
aws personalize start-recommender --recommender-arn "recommender arn"
```

StartRecommender API オペレーションの詳細については、「」および「StopRecommender」を参照してください。

レコメンダーの停止と再起動 (AWS SDKs)

AWS SDKs を使用して、アクティブなレコメンダーを開始したり、非アクティブなレコメンダーを停止したりできます。StartRecommender API オペレーションの詳細については、「」および「StopRecommender」を参照してください。

トピック

- レコメンダーの停止 (AWS SDKs)
- レコメンダーの開始 (AWS SDKs)

レコメンダーの停止 (AWS SDKs)

次のコードは、 AWS SDKs を使用してアクティブなレコメンダーを停止する方法を示しています。停止すると、レコメンダーへの請求と自動再トレーニングがすべて停止します。レコメンダーは再起動するまで使用できません。

SDK for Python (Boto3)

SDK for Python (Boto3) でアクティブなレコメンダーを停止するには、stop_recommender メソッドを使用し、レコメンダーの Amazon リソースネーム (ARN) を次のように指定します。

```
import boto3
personalize = boto3.client('personalize')

stop_recommender_response = personalize.stop_recommender(
    recommenderArn = "recommenderARN"
)
print(stop_recommender_response)
```

SDK for Java 2.x

SDK for Java 2.x でアクティブなレコメンダーを停止するには、stopRecommender メソッドを使用して、次のようにレコメンダーの ARN を指定します。

```
public static void stopRecommender(PersonalizeClient personalizeClient,
                                           String datasetGroupArn) {

    try {

        StopRecommenderRequest stopRecommenderRequest =
 StopRecommenderRequest.builder()
                .recommenderArn(recommenderArn)
                .build();
        personalizeClient.stopRecommender(stopRecommenderRequest);
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
```

```
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { StopRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the request params
export const stopRecommenderParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new StopRecommenderCommand(stopRecommenderParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

レコメンダーの開始 (AWS SDKs)

次のコードは、 AWS SDKs を使用して非アクティブなレコメンダーを開始する方法を示しています。レコメンダーの状態がアクティブになると、レコメンダーのステータスからのレコメンデーションの取得を再開できます。同時に、レコメンダー請求と自動再トレーニングも再開されます。

SDK for Python (Boto3)

SDK for Python (Boto3) を使用して非アクティブなレコメンダーを開始するには、start_recommender メソッドを使用し、次のようにレコメンダーの Amazon リソースネーム (ARN) を指定します。

```
import boto3
personalize = boto3.client('personalize')

start_recommender_response = personalize.start_recommender(
    recommenderArn = "recommenderARN"
)
print(start_recommender_response)
```

SDK for Java 2.x

SDK for Java 2.x で非アクティブなレコメンダーを起動するには、startRecommender メソッドを使用してレコメンダーの ARN を次のように指定します。

```
public static void startRecommender(PersonalizeClient personalizeClient,
                                            String datasetGroupArn) {

    try {

        StartRecommenderRequest startRecommenderRequest =
 StartRecommenderRequest.builder()
                .recommenderArn(recommenderArn)
                .build();
        personalizeClient.startRecommender(startRecommenderRequest);
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { StartRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
```

```
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the request params
export const startRecommenderParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new StartRecommenderCommand(startRecommenderParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

カスタムリソースの作成

データをインポートしたら、レコメンデーションを取得するために使用するカスタムリソースを作成する準備が整います。レコメンデーションを生成するカスタムリソースを作成するには、以下の作業を行います。

1. ソリューションを設定する: モデルが特定のビジネスニーズを満たすように、ソリューションパラメータとレシピ固有のハイパーパラメータをカスタマイズします。デフォルトでは、新しいソリューションバージョンは自動トレーニングを使用して、設定可能な頻度でソリューションバージョンを作成します。デフォルトの頻度は 7 日ごとです。
2. ソリューションバージョンの作成 (モデルのトレーニング): 自動トレーニングを使用するソリューションでは、ソリューションがアクティブになるとソリューションバージョンの作成が自動的に開始されます。手動トレーニングを使用するソリューションでは、ソリューションバージョンを手動で作成します。ソリューションバージョンでは、Amazon Personalize のレコメンデーションまたはユーザーセグメントが生成されます。
3. キャンペーン付きのソリューションバージョンのデプロイ (リアルタイムレコメンデーションのみ): ソリューションバージョンをデプロイするキャンペーンを作成します。リアルタイムのレコ

メンデーションをリクエストする際には、このキャンペーンを使用します。バッチレコメンデーションを取得しようとしている場合は、キャンペーンを作成する必要はありません。

トピック

- ソリューションとソリューションバージョンの作成
- キャンペーンの作成

ソリューションとソリューションバージョンの作成

データのインポートが完了すると、ソリューションを作成する準備が整います。ソリューションとは、Amazon Personalize のレシピ、カスタマイズされたパラメータ、および 1 つ以上のソリューションバージョン (トレーニング済みモデル) の組み合わせをいいます。

Amazon Personalize でソリューションを作成するには、次のように操作します。

1. ソリューションを作成する – モデルが特定のビジネスニーズを満たすように、ソリューションパラメータとレシピ固有のハイパーパラメータをカスタマイズします。ソリューションの作成と構成 を参照してください。使用可能なレシピのリストを表示するには、「レシピの選択」を参照してください。

   デフォルトでは、新しいソリューションバージョンは自動トレーニングを使用して 7 日ごとにソリューションバージョンを作成します。トレーニングの頻度は構成できます。
2. ソリューションバージョンの作成 (モデルのトレーニング): 自動トレーニングを使用するソリューションでは、ソリューションがアクティブになるとソリューションバージョンの作成が自動的に開始されます。ソリューションバージョンは引き続き手動で作成できます。自動トレーニングがオフになっているソリューションの場合は、ソリューションバージョンを手動で作成します。ソリューションバージョンの作成 を参照してください。
3. ソリューションバージョンを評価する – Amazon Personalize が新しいソリューションバージョンから生成するメトリクスを使用して、モデルのパフォーマンスを評価します。「メトリクスを使用してソリューションバージョンを評価する」を参照してください。

トピック

- ソリューションの作成と構成
- ソリューションバージョンの作成
- メトリクスを使用してソリューションバージョンを評価する

ソリューションの作成と構成

データのインポートが完了すると、ソリューションを作成する準備が整います。ソリューションとは、Amazon Personalize レシピ、カスタマイズされたトレーニングパラメータ、および 1 つ以上のソリューションバージョンの組み合わせを指します。ソリューションバージョンとは、トレーニング済みの機械学習モデルをいいます。

デフォルトでは、すべての新しいソリューションは自動トレーニングを使用して 7 日ごとに新しいソリューションバージョンを作成します。自動トレーニングは、前回のトレーニング以降に一括またはリアルタイムのインタラクションデータをインポートした場合にのみ発生します。これには、アイテムインタラクション、または Next-Best-Action レシピを使用するソリューションの場合はアクションインタラクションデータが含まれます。自動トレーニングは、ソリューションを削除するまで続行されます。詳細については、「自動トレーニングの設定」を参照してください。

既存のソリューションがある場合は、Amazon Personalize コンソールを使用してソリューションを複製できます。ソリューションのクローンを作成するときは、レシピやハイパーパラメータなどの既存のソリューションの設定を開始点として使用し、変更を加えることができます。詳細については、「ソリューションの複製 (コンソール)」を参照してください。

コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用してソリューションを作成および設定できます。ソリューションを作成したら、Amazon Personalize コンソールのソリューションの詳細ページ、または DescribeSolutionオペレーションを使用して、その設定の詳細を表示できます。

ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブである間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

トピック

- ソリューションの作成 (コンソール)
- ソリューションの作成 (AWS CLI)
- ソリューションの作成 (AWS SDKs)
- トレーニング設定
- ソリューションの複製 (コンソール)

ソリューションの作成 (コンソール)

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブである間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

コンソールでソリューションを作成するには、データセットグループを選択し、ソリューション名、レシピ、およびオプションのトレーニング設定を指定します。

ソリューションを設定する (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。
3. 概要ページのステップ 3 で、次のいずれかを実行します。
   - ドメインデータセットグループを作成した場合は、カスタムリソースを使用する を選択し、ソリューションの作成 を選択します。
   - カスタムデータセットグループを作成した場合は、[ソリューションの作成] を選択します。
4. [Solution name (ソリューション名)] で、ソリューションの名前を指定します。
5. ソリューションタイプ で、作成するソリューションのタイプを選択します。選択したタイプによって、利用できるレシピが決まります。
   - [アイテムのレコメンデーション] を選択して、ユーザーのアイテムのレコメンデーションを取得します。例えば、パーソナライズされた映画のレコメンデーションなどです。
   - [アクションのレコメンデーション] を選択して、ユーザーのアクションのレコメンデーションを取得します。例えば、アプリのダウンロードなど、ユーザーにとって次に最適なアクションを生成します。
   - [ユーザーセグメンテーション] を選択してアイテムデータに基づいてユーザーセグメント (ユーザーのグループ) を取得します。
6. [Recipe] (レシピ) で、レシピを選択します (「レシピの選択」を参照)。

7. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
8. [次へ] をクリックします。
9. 「トレーニング設定」ページで、ビジネス要件を満たすようにソリューションをカスタマイズします。

   - 自動トレーニング で、ソリューションが自動トレーニングを使用するかどうかを選択します。自動トレーニングを使用する場合は、 を変更できますAutomatic training frequency。デフォルトのトレーニング頻度は 7 日ごとです。

     自動トレーニングを使用することをお勧めします。これにより、レコメンデーションの関連性を簡単に維持できます。トレーニングの頻度は、ビジネス要件、使用するレシピ、データをインポートする頻度によって異なります。詳細については、「自動トレーニングの設定」を参照してください。関連性の維持については、「」を参照してくださいレコメンデーションの関連性の維持。
   - ハイパーパラメータ設定 で、レシピとビジネスニーズに基づいてハイパーパラメータオプションを設定します。異なるレシピは異なるハイパーパラメータを使用します。使用可能なハイパーパラメータについては、「」の個々のレシピを参照してくださいレシピの選択。
   - トレーニング用の列 で、レシピがアイテムレコメンデーションまたはユーザーセグメントを生成する場合は、オプションで Amazon Personalize がソリューションバージョンを作成するときに考慮する列を選択します。詳細については、「トレーニング時に使用する列の設定」を参照してください。
   - 追加設定 で、アイテムインタラクションデータセットに EVENT_TYPE または EVENT_TYPE 列と EVENT_VALUE 列の両方がある場合、オプションでイベントタイプフィールドとイベント値のしきい値フィールドを使用して、Amazon Personalize がモデルのトレーニング時に使用するアイテムインタラクションデータを選択します。詳細については、「トレーニングに使用するアイテムインタラクションデータの選択」を参照してください。
   - User-Personalization レシピ または Personalized-Ranking レシピ レシピのいずれかを使用する場合は、オプションで [目的] を指定し、[目的の感度] を選択して、関連性に加えて目的に合わせてソリューションを最適化します。目的感度は、Amazon Personalize が目的に基づいてアイテムのレコメンデーションとインタラクションデータを介した関連性のバランスをとる方法を設定します。詳細については、「追加の目的のためのソリューションの最適化」を参照してください。
10. Next を選択し、ソリューションの詳細を確認します。ソリューションの作成後にソリューションの設定を変更することはできません。

11. [Create solution (ソリューションの作成)] を選択します。ソリューションを作成すると、Amazon Personalize は 1 時間以内に最初のソリューションバージョンの作成を開始します。トレーニングが開始されたら、ソリューションの詳細ページのソリューションバージョンセクションでモニタリングできます。自動的に作成されたソリューションバージョンには、トレーニングタイプが AUTOMATIC です。

    ソリューションバージョンが ACTIVE になると、それを使用してレコメンデーションを取得する準備が整います。アクティブなソリューションバージョンの使用方法は、レコメンデーションの取得方法によって異なります。

    - リアルタイムのレコメンデーションについては、Amazon Personalize キャンペーンで ACTIVE ソリューションバージョンをデプロイします。キャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。
    - バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) を参照してください。

ソリューションの作成 (AWS CLI)

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブである間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

を使用してソリューションを作成するには AWS CLI、 `create-solution` コマンドを使用します。このコマンドは CreateSolution API オペレーションを使用します。次のコードは、自動トレーニングを使用するソリューションを作成する方法を示しています。5 日ごとに新しいソリューションバージョンが自動的に作成されます。

コードを使用するには、コードを更新してソリューションに名前を付け、データセットグループの Amazon リソースネーム (ARN) を指定し、オプションでトレーニング頻度を変更し、使用するレシピの ARN を指定します。レシピの詳細については、「レシピの選択」を参照してください。

```
aws personalize create-solution \
```

```
--name solution name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--perform-auto-training \
--solution-config "{\"autoTrainingConfig\": {\"schedulingExpression\": \"rate(5
 days)\"}}"
```

- 自動トレーニングを使用することをお勧めします。これにより、レコメンデーションの関連性の維持と改善が容易になります。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。デフォルトのトレーニング頻度は 7 日ごとです。トレーニングの頻度は、ビジネス要件、使用するレシピ、データをインポートする頻度によって異なります。詳細については、「自動トレーニングの設定」を参照してください。
- レシピに応じて、コードを変更してレシピ固有のプロパティとハイパーパラメータを設定したり (「」を参照ハイパーパラメータおよび HPO)、トレーニングに使用される列を設定したり (「」を参照トレーニング時に使用する列の設定 (AWS CLI))、トレーニングに使用されるアイテムインタラクションデータをフィルタリングしたりできます (「」を参照) トレーニングに使用するアイテムインタラクションデータの選択。
- User-Personalization レシピ または recipe のいずれかを使用する場合はPersonalized-Ranking レシピ、関連性に加えて、目的に合わせてソリューションを最適化できます。詳細については、「追加の目的のためのソリューションの最適化」を参照してください。

ソリューションを作成したら、今後の使用に備えてソリューション ARN を記録します。自動トレーニングでは、ソリューションバージョンの作成は、ソリューションが ACTIVE になった後 1 時間以内に開始されます。1 時間以内にソリューションバージョンを手動で作成した場合、ソリューションは最初の自動トレーニングをスキップします。トレーニングの開始後、 バージョン API オペレーションを使用してソリューションバージョンの Amazon リソースネーム (ARN) ListSolutionを取得できます。ステータスを取得するには、DescribeSolutionバージョン API オペレーションを使用します。

ソリューションバージョンが ACTIVE になると、それを使用してレコメンデーションを取得する準備が整います。アクティブなソリューションバージョンの使用方法は、レコメンデーションの取得方法によって異なります。

- リアルタイムのレコメンデーションについては、Amazon Personalize キャンペーンで ACTIVE ソリューションバージョンをデプロイします。キャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。

- バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) を参照してください。

ソリューションの作成 (AWS SDKs)

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブである間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

AWS SDKsオペレーションを使用します。 CreateSolution次のコードは、自動トレーニングを使用するソリューションを作成する方法を示しています。5 日ごとに新しいソリューションバージョンが自動的に作成されます。

コードを使用するには、コードを更新してソリューションに名前を付け、データセットグループの Amazon リソースネーム (ARN) を指定し、オプションでトレーニング頻度を変更し、使用するレシピの ARN を指定します。レシピの詳細については、「レシピの選択」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
  name = 'solution name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN',
  performAutoTraining = True,
  solutionConfig = {
    "autoTrainingConfig": {
      "schedulingExpression": "rate(5 days)"
    }
  }
)
```

```
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

SDK for JavaScript v3

```
import {
  CreateSolutionCommand,
  PersonalizeClient,
} from "@aws-sdk/client-personalize";

// create client
const personalizeClient = new PersonalizeClient({ region: "REGION" });

// set the solution parameters
export const solutionParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  name: "SOLUTION_NAME" /* required */,
  performAutoTraining: true /* optional, default is true */,
  solutionConfig: {
    autoTrainingConfig: {
      schedulingExpression:
        "rate(5 days)" /* optional, default is every 7 days */,
    },
  },
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionCommand(solutionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 自動トレーニングを使用することをお勧めします。これにより、レコメンデーションの関連性の維持と改善が容易になります。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。デフォルトのトレーニング頻度は 7 日ごとです。トレーニングの頻度は、ビジネス要件、使用するレシピ、データをインポートする頻度によって異なります。詳細については、「自動トレーニングの設定」を参照してください。
- レシピに応じて、コードを変更してレシピ固有のプロパティとハイパーパラメータを設定したり (「」を参照ハイパーパラメータおよび HPO)、トレーニングに使用される列を設定したり (「」を参照トレーニング時に使用される列の設定 (AWS SDKs))、トレーニングに使用されるアイテムインタラクションデータをフィルタリングしたりできます (「」を参照) トレーニングに使用するアイテムインタラクションデータの選択。
- User-Personalization レシピ または recipe のいずれかを使用する場合はPersonalized-Ranking レシピ、関連性に加えて、目的に合わせてソリューションを最適化できます。詳細については、「追加の目的のためのソリューションの最適化」を参照してください。

ソリューションを作成したら、今後の使用に備えてソリューション ARN を記録します。自動トレーニングでは、ソリューションバージョンの作成は、ソリューションが ACTIVE になった後 1 時間以内に開始されます。1 時間以内にソリューションバージョンを手動で作成した場合、ソリューションは最初の自動トレーニングをスキップします。トレーニングの開始後、 バージョン API オペレーションを使用してソリューションバージョンの Amazon リソースネーム (ARN) ListSolutionを取得できます。ステータスを取得するには、DescribeSolutionバージョン API オペレーションを使用します。

ソリューションバージョンが ACTIVE になると、それを使用してレコメンデーションを取得する準備が整います。アクティブなソリューションバージョンの使用方法は、レコメンデーションの取得方法によって異なります。

- リアルタイムのレコメンデーションについては、Amazon Personalize キャンペーンで ACTIVE ソリューションバージョンをデプロイします。キャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。
- バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) を参照してください。

トレーニング設定

ソリューションを作成するときは、特定のビジネスニーズに合わせてトレーニングを設定できます。

- ソリューションが自動トレーニングを使用するかどうかを設定できます。また、トレーニング頻度を設定することもできます。デフォルトでは、すべてのソリューションで自動トレーニングが使用されます。詳細については、「自動トレーニングの設定」を参照してください。
- レシピがアイテムレコメンデーションまたはユーザーセグメントを生成する場合、Amazon Personalize がモデルをトレーニングするときに考慮する列を変更できます (ソリューションバージョンの作成) 。詳細については、「トレーニング時に使用する列の設定」を参照してください。
- ハイパーパラメータを設定して、レシピとビジネスニーズに基づいてモデルを最適化できます。異なるレシピは異なるハイパーパラメータを使用します。ハイパーパラメータの設定については、「」を参照してくださいハイパーパラメータおよび HPO。レシピで使用できるハイパーパラメータについては、「レシピの選択」のレシピのページを参照してください。
- User-Personalization レシピ または Personalized-Ranking レシピ レシピのいずれかを使用する場合は、関連性に加えて、目的に合わせてソリューションを最適化できます。詳細については、追加の目的のためのソリューションの最適化を参照してください。
- イベントタイプとイベント値のデータがある場合は、それらを使用して Amazon Personalize がトレーニング中に考慮するアイテムインタラクションレコードを選択できます。詳細については、「トレーニングに使用するアイテムインタラクションデータの選択」を参照してください。

トピック

- 自動トレーニングの設定
- トレーニング時に使用する列の設定
- 追加の目的のためのソリューションの最適化
- ハイパーパラメータおよび HPO
- トレーニングに使用するアイテムインタラクションデータの選択

自動トレーニングの設定

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブになっている間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

ソリューションを作成するときに、ソリューションが自動トレーニングを使用するかどうかを設定できます。トレーニング頻度を設定することもできます。例えば、5 日ごとに新しいソリューションバージョンを作成するようにソリューションを設定できます。

デフォルトでは、すべての新しいソリューションは自動トレーニングを使用して 7 日ごとに新しいソリューションバージョンを作成します。自動トレーニングは、前回のトレーニング以降に一括またはリアルタイムのインタラクションデータをインポートした場合にのみ発生します。これには、アイテムインタラクション、または Next-Best-Action レシピを使用するソリューションの場合はアクションインタラクションデータが含まれます。自動トレーニングは、ソリューションを削除するまで続行されます。

自動トレーニングを使用することをお勧めします。これにより、ソリューションの維持が容易になります。これにより、ソリューションが最新のデータから学習するために必要な手動トレーニングが削除されます。自動トレーニングを使用しない場合、最新のデータから学習するために、ソリューションの新しいソリューションバージョンを手動で作成する必要があります。これにより、レコメンデーションが古くなり、変換レートが低下する可能性があります。Amazon Personalize のレコメンデーションの管理の詳細については、「」を参照してくださいレコメンデーションの関連性の維持。

自動トレーニングは、Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用して設定できます。コンソールで自動トレーニングを設定する手順については、「」を参照してくださいソリューションの作成 (コンソール)。

ソリューションを作成したら、今後の使用に備えてソリューション ARN を記録します。自動トレーニングでは、ソリューションバージョンの作成は、ソリューションが ACTIVE になった後 1 時間以内に開始されます。1 時間以内にソリューションバージョンを手動で作成した場合、ソリューションは最初の自動トレーニングをスキップします。トレーニングの開始後、 バージョン API オペレーションを使用してソリューションバージョンの Amazon リソースネーム (ARN) ListSolutionを取得できます。ステータスを取得するには、バージョン API DescribeSolutionオペレーションを使用します。

ソリューションバージョンが ACTIVE になると、それを使用してレコメンデーションを取得する準備が整います。アクティブなソリューションバージョンの使用方法は、レコメンデーションの取得方法によって異なります。

- リアルタイムのレコメンデーションについては、Amazon Personalize キャンペーンで ACTIVE ソリューションバージョンをデプロイします。キャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。

- バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) を参照してください。

トピック

- ガイドラインと要件
- 自動トレーニングの設定 (AWS CLI)
- 自動トレーニングの設定 (SDKs)

ガイドラインと要件

自動トレーニングのガイドラインと要件は次のとおりです。

- 自動トレーニングは、前回のトレーニング以降に一括またはリアルタイムのインタラクションデータをインポートした場合にのみ発生します。これには、アイテムインタラクション、または Next-Best-Action レシピを使用するソリューションの場合はアクションインタラクションデータが含まれます。
- 各トレーニングでは、トレーニングに含めるデータセットグループ内のすべてのデータを考慮します。トレーニングで使用される列の設定については、「」を参照してくださいトレーニング時に使用する列の設定。
- ソリューションバージョンは手動で作成できます。
- 自動トレーニングは、ソリューションがアクティブになってから 1 時間以内に開始されます。1 時間以内にソリューションバージョンを手動で作成した場合、ソリューションは最初の自動トレーニングをスキップします。
- トレーニングのスケジュールは、トレーニング開始日に基づいています。例えば、最初のソリューションバージョンが午後 7 時にトレーニングを開始し、毎週のトレーニングを使用する場合、次のソリューションバージョンは 1 週間後の午後 7 時にトレーニングを開始します。
- すべてのレシピについて、少なくとも毎週のトレーニング頻度をお勧めします。トレーニング頻度は 1～30 日の間で指定できます。デフォルトは 7 日ごとです。
  - User-Personalization-v2、User-Personalization、または Next-Best-Action を使用すると、ソリューションが自動的に更新され、レコメンデーションの新しいアイテムまたはアクションが考慮されます。自動更新は自動トレーニングとは異なります。自動更新では、まったく新しいソリューションバージョンは作成されず、モデルは最新のデータから学習しません。ソリューションを維持するには、トレーニング頻度を少なくとも週に 1 回にする必要があります。追加のガイドラインや要件など、自動更新の詳細については、「」を参照してください自動更新。

- Trending-Now を使用すると、Amazon Personalize は、設定可能な時間間隔で、インタラクションデータ内の最もトレンドの高い項目を自動的に識別します。Trending-Now は、バルクまたはストリーミングインタラクションデータを通じて、前回のトレーニング以降に追加されたアイテムをレコメンデーションできます。トレーニング頻度は少なくとも週に 1 回にする必要があります。詳細については、「Trending-Now レシピ」を参照してください。
- 自動更新または Trending-Now レシピでレシピを使用しない場合、Amazon Personalize は次のトレーニング後にのみ新しいアイテムをレコメンデーションと見なします。例えば、Similar-Items レシピを使用し、毎日新しいアイテムを追加する場合、同じ日にレコメンデーションに表示されるには、これらのアイテムの毎日の自動トレーニング頻度を使用する必要があります。

自動トレーニングの設定 (AWS CLI)

次のコードは、5 日ごとにソリューションバージョンを自動的に作成するソリューションを作成する方法を示しています。自動トレーニングを無効にするには、 `perform-auto-training`を に設定しますfalse。

トレーニング頻度を変更するには、 `schedulingExpression`で を変更できますautoTrainingConfig。式は `rate(value unit)`形式である必要があります。値には、1～30 の数値を指定します。単位には、 dayまたは を指定しますdays。

create-solution コマンドの詳細については、「」を参照してくださいソリューションの作成 (AWS CLI)。

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--perform-auto-training \
--solution-config "{\"autoTrainingConfig\": {\"schedulingExpression\": \"rate(5
 days)\"}}"
```

自動トレーニングの設定 (SDKs)

次のコードは、 AWS SDKsで自動トレーニングを使用してソリューションを作成する方法を示しています。このソリューションでは、5 日ごとにソリューションバージョンが自動的に作成されます。自動トレーニングを無効にするには、 `performAutoTraining`を に設定しますfalse。

トレーニング頻度を変更するには、 schedulingExpressionで を変更できますautoTrainingConfig。式は rate(*value unit*)形式である必要があります。値には、1～30の数値を指定します。単位には、 dayまたは を指定しますdays。

CreateSolution API オペレーションの詳細については、「」を参照してくださいソリューションの作成 (AWS SDKs)。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
  name = 'solution name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN',
  performAutoTraining = True,
  solutionConfig = {
    "autoTrainingConfig": {
      "schedulingExpression": "rate(5 days)"
    }
  }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

SDK for JavaScript v3

```
import {
  CreateSolutionCommand,
  PersonalizeClient,
} from "@aws-sdk/client-personalize";

// create client
const personalizeClient = new PersonalizeClient({ region: "REGION" });

// set the solution parameters
export const solutionParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  name: "SOLUTION_NAME" /* required */,
```

```
  performAutoTraining: true /* optional, default is true */,
  solutionConfig: {
    autoTrainingConfig: {
      schedulingExpression:
        "rate(5 days)" /* optional, default is every 7 days */,
    },
  },
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionCommand(solutionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

トレーニング時に使用する列の設定

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブになっている間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

レシピがアイテムのレコメンデーションまたはユーザーセグメントを生成する場合、Amazon Personalize がソリューションバージョンの作成 (モデルのトレーニング) 時に考慮する列を変更できます。

トレーニング時に使用する列を変更して、Amazon Personalize がモデルのトレーニング (ソリューションバージョンの作成) に使用するデータを制御できます。これにより、トレーニングデータのさまざまな組み合わせを試してみることができます。または、意味のあるデータがない列を除外するこ

ともできます。例えば、レコメンデーションのフィルタリングにのみ使用したい列があるとします。この列をトレーニングから除外すると、Amazon Personalize はフィルタリング時にのみこの列を考慮します。

EVENT_TYPE 列を除外することはできません。デフォルトでは、Amazon Personalize はトレーニング時に使用できるすべての列を使用します。次のデータは、常にトレーニングから除外されます。

- ブールデータ型の列
- インプレッションデータ
- カテゴリ別またはテキスト別ではないカスタム文字列フィールド

トレーニングにインプレッションデータを含めることはできませんが、ユースケースやレシピでそのデータが使用されている場合、Amazon Personalize はインプレッションデータを使用してレコメンデーションを取得するときに探索をガイドします。

ソリューションをすでに作成していて、トレーニング時に使用する列を変更したい場合は、ソリューションを複製できます。ソリューションを複製するときは、レシピやハイパーパラメータなど、既存のソリューションの構成を出発点として使用し、必要に応じて変更を加えることができます。詳細については、「ソリューションの複製 (コンソール)」を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDK を使用してトレーニングするときに Amazon Personalize が使用する列を設定できます。Amazon Personalize コンソールで列を選択する方法については、「ソリューションの作成 (コンソール)」の詳細設定ステップを参照してください。ソリューションを作成したら、ソリューションが使用する列を Amazon Personalize コンソールのソリューションの詳細ページに表示するか、DescribeSolution のオペレーションで表示できます。

トピック

- トレーニング時に使用する列の設定 (AWS CLI)
- トレーニング時に使用される列の設定 (AWS SDKs)

トレーニング時に使用する列の設定 (AWS CLI)

列をトレーニングから除外するには、ソリューション構成の一部として `trainingDataConfig` に `excludedDatasetColumns` オブジェクトを指定します。キーごとに、データセットタイプを指定します。値ごとに、除外する列のリストを指定します。次のコードは、 AWS CLIを使用してソリューションを作成するときに列を除外する方法を示しています。

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--solution-config "{\"trainingDataConfig\": {\"excludedDatasetColumns\":
 { \"datasetType\" : [ \"column1Name\", \"column2Name\"]}}}"
```

トレーニング時に使用される列の設定 (AWS SDKs)

列をトレーニングから除外するには、ソリューション構成の一部として trainingDataConfig に excludedDatasetColumns オブジェクトを指定します。キーごとに、データセットタイプを指定します。値ごとに、除外する列のリストを指定します。次のコードは、SDK for Python (Boto3) を使用してソリューションを作成するときに、列を除外する方法を示しています。

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
  name = 'solution name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN',
  solutionConfig = {
    "trainingDataConfig": {
      "excludedDatasetColumns": {
          "datasetType": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

追加の目的のためのソリューションの最適化

> **Important**
>
> ソリューションを作成した後は、その構成を変更することはできません。デフォルトでは、新しいソリューションはすべて自動トレーニングを使用します。自動トレーニングでは、ソリューションがアクティブである間はトレーニングコストが発生します。不要なコストを避

けるため、終了したら必ずソリューションを削除してください。トレーニング費用の詳細については、「Amazon Personalize 料金表」を参照してください。

User-Personalization レシピまたは Personalized-Ranking レシピを使用すると、トレーニングの前に、収益の最大化などの最大の関連性に加えて、目的に合わせて Amazon Personalize ソリューションを最適化できます。

アイテムレコメンデーションレシピを備えた Amazon Personalize の主な目的は、履歴およびリアルタイムのアイテムインタラクションデータに基づいて、ユーザーにとって最も関連性の高いアイテムを予測することです。これらは、ユーザーが操作する可能性が最も高いアイテムです (例えば、ユーザーがクリックする可能性が最も高いアイテム)。ストリーミング時間 (分) の最大化や収益の増加など、追加の目的がある場合は、関連性と目的の両方に基づいてレコメンデーションを生成するソリューションを作成できます。

追加の目的のためにソリューションを最適化するには、User-Personalization レシピまたは Personalized-Ranking レシピを使用して新しいソリューションを作成し、目的に関連する Items データセットの数値メタデータ列を選択します。レコメンデーションを生成する場合、Amazon Personalize は、このデータ列の値がより高いアイテムをより重要視します。例えば、ストリーミング時間 (分) を最大化するために VIDEO_LENGTH 列を選択したり、収益を最大化するために PRICE 列を選択したりできます。

Amazon Personalize コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDK を使用できます。Amazon Personalize コンソールの使用方法については、を参照してくださいソリューションの作成 (コンソール)。

トピック

- ガイドラインと要件
- 目的の重点の置き方と関連性のバランシング
- 最適化パフォーマンスの測定
- ソリューションの最適化 (AWS CLI)
- ソリューション (AWS SDK) の最適化
- サンプル Jupyter ノートブック

ガイドラインと要件

目的の要件は次のとおりです。

- 目的に選択できる列は 1 つだけです。
- 列は、スキーマ内の数値型である必要があります。
- 列にスキーマ内の `null` タイプを含めることはできません。

スキーマとデータ型の詳細については、「スキーマ」を参照してください。

目的の重点の置き方と関連性のバランシング

関連性よりも目的に基づいてアイテムを推奨する場合、トレードオフが生じる可能性があります。例えば、レコメンデーションを通じて収益を増やしたい場合、高額なアイテムのみのレコメンデーションでは、ユーザーとの関係においてアイテムの関連性が低下し、ユーザーのエンゲージメントとコンバージョンが減少する可能性があります。

関連性と目的のバランスを設定するには、ソリューションを作成するときに、次の目的の感度レベルのいずれかを選択します。

- Off (オフ): Amazon Personalize は、主にアイテムインタラクションデータを使用して、ユーザーにとって最も関連性の高いアイテムを予測します。
- Low (低): Amazon Personalize は、目的をあまり重要視しません。アイテムインタラクションデータを介した関連性がより重要視されます。
- Medium (中): Amazon Personalize は、アイテムインタラクションデータを通じて、目的と関連性を同等に重視します。
- High (高): Amazon Personalize は、目的をより重要視します。アイテムインタラクションデータを介した関連性はあまり重要視されません。

最適化パフォーマンスの測定

最適化の目的を持つソリューションのソリューションバージョンを作成する (モデルをトレーニングする) と、Amazon Personalize は `average_rewards_at_k` メトリクスを生成します。`average_rewards_at_k` のスコアは、ソリューションバージョンが目的を達成する上でどれだけよく機能するかを示します。このメトリクスを計算するために、Amazon Personalize は各ユーザーについての報酬を次のように計算します。

```
rewards_per_user = total rewards from the user's interactions with their
top 25 reward generating recommendations / total rewards from the user's
interactions with recommendations
```

最終的な average_rewards_at_k は、1 以下 0 超の 10 進値に正規化されたすべての rewards_per_user の平均です。値が 1 に近いほど、レコメンデーションから期待できるユーザーあたりの平均利益が大きくなります。

例えば、クリックから得られる収益を最大化することが目的の場合、Amazon Personalize は、ユーザーが上位 25 位の最も高額なレコメンデーションからクリックしたアイテムによって生成された合計収益を、ユーザーがクリックしたすべての推奨アイテムからの収益で除することにより、各ユーザースコアを計算します。その後、Amazon Personalize は、すべてのユーザースコアの正規化された平均を返します。average_rewards_at_k が 1 に近いほど、レコメンデーションから得られるユーザーあたりの平均収益が増大することが期待できます。

メトリクスの生成の詳細については、「メトリクスを使用してソリューションバージョンを評価する」を参照してください。

ソリューションの最適化 (AWS CLI)

User-Personalization または Personalized-Ranking レシピを使用する場合にのみ、目的に合わせて最適化できます。を使用して別の目的に合わせてソリューションを最適化するには AWS CLI、新しいソリューションを作成し、optimizationObjectivesolutionConfigオブジェクトのキーを使用して目的の詳細を指定します。optimizationObjective には以下のフィールドがあります。

- itemAttribute: 目的に関連する Items データセットの数値メタデータ列の名前を指定します。
- objectiveSensitivity: レコメンデーションを生成するときに、ソリューションが目的を重要視するレベルを指定します。目的の感度レベルは、Amazon Personalize が、目的と、アイテムインタラクションデータを介した関連性に基づいて推奨されるアイテムのバランスをとる方法を設定します。objectiveSensitivity は、OFF、[LOW] (低)、MEDIUM、または HIGH とすることができます。詳細については、「目的の重点の置き方と関連性のバランシング」を参照してください。

次に create-solution AWS CLI コマンドの例を示します。solution name、dataset group arn、および recipe arn の値を独自の値に置き換えます。

optimizationObjective について、COLUMN_NAME を、目的に関連する Items データセットの数値メタデータ列名に置き換えます。objectiveSensitivity について、[OFF] (オフ)、[LOW] (低)、[MEDIUM] (中)、または [HIGH] (高) を指定します。

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group arn \
```

```
--recipe-arn recipe arn \
--solution-config "{\"optimizationObjective\":{\"itemAttribute\":\"COLUMN_NAME\",
\"objectiveSensitivity\":\"MEDIUM\"}}"
```

ソリューションの準備が整ったら、新しいソリューションバージョンを作成します (コマンドの例については、「ソリューションの作成 (AWS CLI)」を参照してください)。ソリューションバージョンを作成すると、ソリューションバージョンのメトリクスで最適化パフォーマンスを表示できます。最適化パフォーマンスの測定 を参照してください。

ソリューション (AWS SDK) の最適化

User-Personalization または Personalized-Ranking レシピを使用する場合にのみ、目的に合わせて最適化できます。

AWS SDK を使用して別の目的に向けてソリューションを最適化するには、新しいソリューションを作成し、optimizationObjectivesolutionConfigソリューションのオブジェクト内のキーを使用して目的の詳細を指定します。optimizationObjective には以下のフィールドがあります。

- itemAttribute: 目的に関連する、データグループの Items データセットの数値メタデータ列の名前を指定します。
- objectiveSensitivity: レコメンデーションを生成するときに、ソリューションが目的を重要視するレベルを指定します。目的の感度レベルは、Amazon Personalize が、目的と、アイテムインタラクションデータを介した関連性に基づいて推奨されるアイテムのバランスをとる方法を設定します。objectiveSensitivity は、OFF、LOW、MEDIUM、または HIGH にすることができます。詳細については、「目的の重点の置き方と関連性のバランシング」を参照してください。

次のコードを使用して、 AWS SDK for Python (Boto3) または AWS SDK for Java 2.xで追加の目的を持つソリューションを作成します。

ソリューションの準備が整ったら、新しいソリューションバージョンを作成します (コードの例については、「ソリューションバージョン (AWS SDK) の作成」を参照してください)。ソリューションバージョンを作成すると、ソリューションバージョンのメトリクスで最適化パフォーマンスを表示できます。最適化パフォーマンスの測定 を参照してください。

SDK for Python (Boto3)

追加の目的向けに最適化されたソリューションを作成するには、次の create_solution メソッドを使用します。solution name、dataset group arn、および recipe arn の値を独自の値に置き換えます。

optimizationObjective について、COLUMN_NAME を、目的に関連する Items データセットの数値メタデータ列名に置き換えます。objectiveSensitivity について、[OFF] (オフ)、[LOW] (低)、[MEDIUM] (中)、または [HIGH] (高) を指定します。

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
    name= 'solution name',
    recipeArn = 'recipe arn',
    datasetGroupArn = 'dataset group arn',
    solutionConfig = {
        "optimizationObjective": {
            "itemAttribute": "COLUMN_NAME",
            "objectiveSensitivity": "MEDIUM"
        }
    }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

SDK for Java 2.x

追加の目的向けに最適化されたソリューションを作成するには、次 createPersonalizeSolution メソッドを使用し、Amazon Personalize サービスクライアント、データセットグループの Amazon リソースネーム (ARN)、ソリューション名、レシピ ARN、アイテム属性、および目的の感度レベルをパラメータとして渡します。

```
public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                                String datasetGroupArn,
                                                String solutionName,
                                                String recipeArn,
                                                String itemAttribute,
                                                String objectiveSensitivity) {

    try {
        OptimizationObjective optimizationObjective =
 OptimizationObjective.builder()
            .itemAttribute(itemAttribute)
            .objectiveSensitivity(objectiveSensitivity)
            .build();
```

```
        SolutionConfig solutionConfig = SolutionConfig.builder()
            .optimizationObjective(optimizationObjective)
            .build();

        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .solutionConfig(solutionConfig)
            .build();

        CreateSolutionResponse solutionResponse =
 personalizeClient.createSolution(solutionRequest);

        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateSolutionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create the personalizeClient
const personalizeClient = new PersonalizeClient({ region: "REGION"});

// set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN',             /* required */
  recipeArn: 'RECIPE_ARN',                          /* required */
  name: 'NAME',                                     /* required */
  solutionConfig: {
    optimizationObjective: {
      itemAttribute: "COLUMN_NAME",           /* specify the numerical column from
 the Items dataset related to your objective */
      objectiveSensitivity: "MEDIUM"          /* specify OFF, LOW, MEDIUM, or HIGH
 */
```

```
    }
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSolutionCommand(createSolutionParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

サンプル Jupyter ノートブック

追加の目的ベースのアイテムメタデータ用に最適化されたソリューションの作成方法を示すサンプル Jupyter ノートブックについては、Amazon Personalize サンプルリポジトリの objective_optimization フォルダを参照してください。 GitHub

ハイパーパラメータおよび HPO

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブになっている間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

トレーニングの前にハイパーパラメータを指定して、トレーニング済みモデルを特定のユースケースに合わせて最適化します。トレーニングプロセス中に値が決まるモデルパラメータとは対照的です。

ハイパーパラメータは、CreateSolution オペレーションに渡された SolutionConfig オブジェクトの一部である `algorithmHyperParameters` キーを使用して指定します。

CreateSolution リクエストの簡約版は以下のとおりです。この例には、solutionConfig オブジェクトが含まれています。solutionConfig を使用してレシピのデフォルトパラメータを上書きします。

```
{
  "name": "string",
  "recipeArn": "string",
  "eventType": "string",
  "solutionConfig": {
      "optimizationObjective": {
          "itemAttribute": "string",
          "objectiveSensitivity": "string"
      },
      "eventValueThreshold": "string",
      "featureTransformationParameters": {
          "string" : "string"
      },
      "algorithmHyperParameters": {
          "string" : "string"
      },
      "hpoConfig": {
          "algorithmHyperParameterRanges": {
              ...
          },
          "hpoResourceConfig": {
              "maxNumberOfTrainingJobs": "string",
              "maxParallelTrainingJobs": "string"
          }
      },
  },
}
```

異なるレシピは異なるハイパーパラメータを使用します。利用可能なハイパーパラメータについては、「レシピの選択」で個々のレシピを参照してください。

ハイパーパラメータ最適化の有効化

ハイパーパラメータの最適化 (HPO) またはチューニングは、特定の学習目標に最適なハイパーパラメータを選択するタスクです。最適なハイパーパラメータは、指定された可能性の範囲内から異なる値を使用して多くのトレーニングジョブを実行することで決定されます。

User-Personalization-v2 と Personalized-Ranking-v2 では、自動トレーニングを有効にすると、Amazon Personalize は 90 日ごとに自動的に HPO を実行します。自動トレーニングがないと、HPO は発生しません。他のすべてのレシピでは、HPO を有効にする必要があります。HPO を使用するには、performHPO を true に設定し、hpoConfig オブジェクトを含めます。

ハイパーパラメータのタイプは、カテゴリ別、継続的、または整数値のいずれかです。hpoConfig オブジェクトにはこれらの各タイプに該当するキーがあります。ここでは、ハイパーパラメータおよび範囲を指定します。リクエストでは各タイプを指定する必要がありますが、レシピに特定のタイプのパラメーターがない場合は空のままにしておくことができます。例えば、User-Personalization には連続型のチューニング可能なハイパーパラメータはありません。そのため、continousHyperParameterRange には空の配列を渡すことができます。

次のコードは、SDK for Python (Boto3) を使用して HPO を有効にしたソリューションを作成する方法を示しています。この例のソリューションは User-Personalization レシピ レシピを使用し、HPO を true に設定しています。このコードは、hidden_dimension、categoricalHyperParameterRanges、integerHyperParameterRanges に値を与えます。continousHyperParameterRange は空で、hpoResourceConfig は maxNumberOfTrainingJobs と maxParallelTrainingJobs を設定します。

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
    name = "solution name",
    datasetGroupArn = 'arn:aws:personalize:region:accountId:dataset-group/
datasetGroupName',
    recipeArn = 'arn:aws:personalize:::recipe/aws-user-personalization',
    performHPO = True,
    solutionConfig = {
        "algorithmHyperParameters": {
          "hidden_dimension": "55"
        },
        "hpoConfig": {
          "algorithmHyperParameterRanges": {
              "categoricalHyperParameterRanges": [
                  {
                      "name": "recency_mask",
                      "values": [ "true", "false"]
                  }
              ],
```

```
            "integerHyperParameterRanges": [
                {
                    "name": "bptt",
                    "minValue": 2,
                    "maxValue": 22
                }
            ],
            "continuousHyperParameterRanges": [

            ]
        },
        "hpoResourceConfig": {
            "maxNumberOfTrainingJobs": "4",
            "maxParallelTrainingJobs": "2"
        }
      }
    }
)
```

HPO の詳細については、「モデルの自動チューニング」を参照してください。

ハイパーパラメータの表示

DescribeSolution 操作を呼び出すことによって、ソリューションのハイパーパラメータを表示できます。次の例は DescribeSolution 出力を示しています。ソリューションバージョンの作成 (モデルのトレーニング) 後に、DescribeSolutionVersion 操作を使用してハイパーパラメータを表示することもできます。

```
{
  "solution": {
    "name": "hpo_coonfig_solution",
    "solutionArn": "arn:aws:personalize:region:accountId:solution/solutionName",
    "performHPO": true,
    "performAutoML": false,
    "recipeArn": "arn:aws:personalize:::recipe/aws-user-personalization",
    "datasetGroupArn": "arn:aws:personalize:region:accountId:dataset-group/
datasetGroupName",
    "eventType": "click",
    "solutionConfig": {
      "hpoConfig": {
        "hpoResourceConfig": {
          "maxNumberOfTrainingJobs": "4",
          "maxParallelTrainingJobs": "2"
```

```
        },
        "algorithmHyperParameterRanges": {
          "integerHyperParameterRanges": [
            {
              "name": "training.bptt",
              "minValue": 2,
              "maxValue": 22
            }
          ],
          "continuousHyperParameterRanges": [],
          "categoricalHyperParameterRanges": [
            {
              "name": "data.recency_mask",
              "values": [
                "true",
                "false"
              ]
            }
          ]
        }
      },
      "algorithmHyperParameters": {
        "hidden_dimension": "55"
      }
    },
    "status": "ACTIVE",
    "creationDateTime": "2022-07-08T12:12:48.565000-07:00",
    "lastUpdatedDateTime": "2022-07-08T12:12:48.565000-07:00"
  }
}
```

トレーニングに使用するアイテムインタラクションデータの選択

> **Important**
>
> ソリューションを作成した後は、その設定を変更することはできません。デフォルトでは、すべての新しいソリューションで自動トレーニングが使用されます。自動トレーニングでは、ソリューションがアクティブである間にトレーニングコストが発生します。不要なコストを避けるため、作業が終了したらソリューションを削除してください。トレーニングコストの詳細については、「Amazon Personalize の料金」を参照してください。

Amazon Personalize がソリューションバージョンの作成 (モデルのトレーニング) 時に使用するアイテムインタラクションデータセットのイベントを選択できます。トレーニングの前にアイテムインタラクションデータを選択すると、データの関連するサブセットのみをトレーニングに使用したり、ノイズを除去して、より最適化されたモデルをトレーニングしたりできます。アイテムインタラクションデータセットの詳細については、「スキーマ」および「アイテムインタラクションデータセット」を参照してください。

> **Note**
>
> User-Personalization-v2 または Personalized-Ranking-v2 を使用する場合、トレーニングコストは、イベントタイプまたは値でフィルタリングする前にアイテムインタラクションデータに基づきます。料金の詳細については、「Amazon Personalize の料金」を参照してください。

アイテムインタラクションデータは次のように選択できます。

- タイプに基づいてレコードを選択する - ソリューションを設定するときに、アイテムインタラクションデータセットの EVENT_TYPE 列にイベントタイプが含まれている場合、オプションで、トレーニングで使用するイベントタイプを指定できます。例えば、アイテムインタラクションデータセットに [購入]、[クリック]、および [視聴] イベントタイプが含まれており、Amazon Personalize で [視聴] イベントのみを使用してモデルをトレーニングする場合、ソリューションを設定するときに、Amazon Personalize がトレーニングで使用する `event type` として [視聴] を指定します。

  アイテムインタラクションデータセットの EVENT_TYPE 列に複数のイベントタイプがあり、ソリューションを設定するときにイベントタイプを指定しなかった場合、Amazon Personalize は、タイプにかかわらず、すべてのアイテムインタラクションデータを同じ重みでトレーニングに使用します。
- タイプと値に基づいてレコードを選択する - ソリューションを設定するときに、アイテムインタラクションデータセットに EVENT_TYPE フィールドと EVENT_VALUE フィールドが含まれている場合、トレーニングからレコードを除外するためのしきい値として特定の値を設定できます。例えば、EVENT_TYPE が [watch] (視聴) のイベントの EVENT_VALUE データが、ユーザーが視聴した動画のパーセンテージ (%) である場合であって、イベント値のしきい値を 0.5 に、イベントタイプを [watch] (視聴) に設定する場合、Amazon Personalize は、EVENT_VALUE が 0.5 以上の [watch] (視聴) インタラクションイベントのみを使用してモデルをトレーニングします。

次のコードは、SDK for Python (Boto3) を使用して、 がビデオの半分以上を視聴したwatchイベントのみを使用するソリューションを作成する方法を示しています。

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
    name = 'solution name',
    datasetGroupArn = 'arn:aws:personalize:region:accountId:dataset-group/
datasetGroupName',
    recipeArn = 'arn:aws:personalize:::recipe/aws-user-personalization-v2',
    eventType = 'watch',
    solutionConfig = {
        "eventValueThreshold": "0.5"
    }
)

# Store the solution ARN
solution_arn = create_solution_response['solutionArn']

# Use the solution ARN to get the solution status
solution_description = personalize.describe_solution(solutionArn = solution_arn)
['solution']
print('Solution status: ' + solution_description['status'])
```

ソリューションの複製 (コンソール)

新しいソリューションを作成する場合、Amazon Personalize コンソールを使用してソリューションを複製できます。ソリューションを複製するときは、レシピやハイパーパラメータなど、既存のソリューションの設定を出発点として使用し、必要に応じて変更を加えることができます。これは、ソリューションに 1 つの変更を加え、他のすべてのプロパティは変更しないでおきたい場合に便利です。例えば、トレーニングデータの新しい列をデータセットに追加する場合などです。この場合は、ソリューションを複製し、ソリューションに名前を付け、トレーニング時に使用する列を変更し、その他のプロパティはすべて変更しないでおきます。

ソリューションの複製

ソリューションを複製するには、既存のソリューションを選択し、[ソリューションの複製] オプションを選択します。次に、新しいソリューションに名前を付け、関連するフィールドを変更します。

ソリューションを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のページで、データセットグループを選択します。
3. [カスタムリソース] を選択し、[ソリューション] を選択します。
4. 複製するソリューションを選択します。
5. [アクション] を選択し、[ソリューションの複製] を選択します。
6. 新しいソリューションに名前を付けます。
7. ソリューションの詳細と詳細設定に変更を加えます。Amazon Personalize では、これらのフィールドには既存のソリューションの値があらかじめ入力されています。各フィールドの詳細については、「ソリューションの作成と構成」を参照してください。

ソリューションバージョンの作成

完了するとソリューションの作成と構成、トレーニングを開始する準備が整います。

- ソリューションが自動トレーニングを使用している場合、ソリューションは指定したトレーニング頻度でソリューションバージョンを作成します。デフォルトでは、新しいソリューションはすべて自動トレーニングを使用して 7 日ごとに新しいソリューションバージョンを作成します。ソリューションバージョンは引き続き手動で作成できます。詳細については、「自動トレーニングの設定」を参照してください。
- ソリューションのauto トレーニングをオフにした場合や、手動でトレーニングしたい場合は、ソリューションバージョンを手動で作成できます。ソリューションバージョンとは、トレーニング済みの機械学習モデルをいいます。コンソール、 AWS Command Line Interface (AWS CLI)、または AWS SDK を使用してソリューションバージョンを作成できます。ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、この the section called “StopSolutionVersionCreation” 操作を使用してソリューションバージョンの作成プロセスを停止できます。ソリューションバージョンの作成の停止 を参照してください。

トピック

- ソリューションバージョンの作成 (コンソール)
- ソリューションバージョンの作成 (AWS CLI)
- ソリューションバージョン (AWS SDK) の作成
- ソリューションバージョンの作成の停止

ソリューションバージョンの作成 (コンソール)

Amazon Personalize コンソールで最初にソリューションを作成するときは、ソリューションバージョンも作成します。ソリューションの詳細のページでは、[Solution versions] (ソリューションバージョン) のセクションでトレーニングの進捗状況を追跡できます。トレーニングが完了すると、ステータスは Active になり、準備完了です。「キャンペーンの作成」または「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

既存のソリューションに対して追加のソリューションバージョンを作成したい場合は、以下のようにソリューションの概要ページから新しいソリューションバージョンを作成します。

新しいソリューションバージョンを作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. データセットグループのページに移動し、新しいソリューションのデータセットグループを選択します。
3. ナビゲーションペインの [カスタムリソース] を選択し、[ソリューションとレシピ] を選択します。
4. [Solution and recipes] (ソリューションとレシピ) のページで、ソリューションバージョンを作成するソリューションを選択します。
5. ソリューションの概要のページで、[Create solution version] (ソリューションバージョンを作成) を選択して、新しいモデルのトレーニングを開始します。

ソリューションの詳細のページでは、[Solution versions] (ソリューションバージョン) のセクションでトレーニングの進捗状況を追跡できます。トレーニングが完了すると、ステータスは [Active] (アクティブ) になり、Amazon Personalize が提供するメトリクスを使用して評価できます。詳細については、「メトリクスを使用してソリューションバージョンを評価する」を参照してください。

エラーが原因でトレーニングが完了しない場合、トレーニングの料金は請求されません。ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、ソリューションバージョンの作成プロセスを停止できます。ソリューションバージョンの作成を停止するには、ソリューションバージョンの詳細のページに移動して、[Stop] (停止) を選択します。ソリューションバージョンの作成の停止 を参照してください。

アクティブなソリューションバージョンをどのように使用するかは、推奨事項がどのように表示されるかによって異なります。

- リアルタイムのレコメンデーションを行うには、Amazon Personalize キャンペーンを含む ACTIVE ソリューションバージョンをデプロイします。このキャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。
- バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) を参照してください。

ソリューションバージョンの作成 (AWS CLI)

ソリューションが ACTIVE である場合は、次のコマンドを実行してモデルをトレーニングします。`solution arn` を、ソリューションの作成と構成 からのソリューションの Amazon リソースネーム (ARN) に置き換えます。

```
aws personalize create-solution-version \
  --solution-arn solution arn
```

ソリューションバージョンの ARN が表示されます。次に例を示します。

```
{
  "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/SolutionName/
<version-id>"
}
```

`describe-solution-version` コマンドを使用して、ソリューションバージョンのトレーニングステータスを確認します。前のステップで返ったソリューションバージョンの ARN を指定します。API の詳細については、「DescribeSolutionVersion」を参照してください。

```
aws personalize describe-solution-version \
  --solution-version-arn solution version arn
```

ソリューションバージョンのプロパティとトレーニングの `status` が表示されます。最初、ステータスには CREATE PENDING と表示されます。次に例を示します。

```
{
  "solutionVersion": {
      "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/
solutionName/<version-id>",
      ...,
      "status": "CREATE PENDING"
```

```
  }
}
```

status が ACTIVE であればトレーニングは完了しており、Amazon Personalize が提供するメトリクスを使用して評価できます。詳細については、「メトリクスを使用してソリューションバージョンを評価する」を参照してください。エラーが原因でトレーニングが完了しない場合、トレーニングの料金は請求されません。

ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、この StopSolutionVersionCreation 操作を使用してソリューションバージョンの作成プロセスを停止できます。ソリューションバージョンの作成の停止 を参照してください。

アクティブなソリューションバージョンをどのように使用するかは、レコメンデーションを取得する方法によって異なります。

- リアルタイムのレコメンデーションを行うには、Amazon Personalize キャンペーンを含む ACTIVE ソリューションバージョンをデプロイします。このキャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。
- バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) を参照してください。

ソリューションバージョン (AWS SDK) の作成

ソリューションが ACTIVE である場合は、次のコードを使用してソリューションバージョンを作成します。ソリューションの作成と構成 から Amazon リソースネーム (ARN) を指定します。DescribeSolutionVersion の操作を使用してソリューションバージョンのステータスを取得します。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')
# Store the solution ARN
solution_arn = 'solution arn'

# Use the solution ARN to get the solution status.
solution_description = personalize.describe_solution(solutionArn = 'solution_arn')
['solution']
```

```
print('Solution status: ' + solution_description['status'])

# Use the solution ARN to create a solution version.
print ('Creating solution version')
response = personalize.create_solution_version(solutionArn = solution_arn)
solution_version_arn = response['solutionVersionArn']
print('Solution version ARN: ' + solution_version_arn)

# Use the solution version ARN to get the solution version status.
solution_version_description = personalize.describe_solution_version(
    solutionVersionArn = solution_version_arn)['solutionVersion']
print('Solution version status: ' + solution_version_description['status'])
```

SDK for Java 2.x

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
 personalizeClient, String solutionArn) {
        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String solutionStatus = "";
        String solutionVersionStatus = "";
        String solutionVersionArn = "";

        try {
            DescribeSolutionRequest describeSolutionRequest =
 DescribeSolutionRequest.builder()
                .solutionArn(solutionArn)
                .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            // Wait until solution is active.
            while (Instant.now().getEpochSecond() < maxTime) {

                solutionStatus =
 personalizeClient.describeSolution(describeSolutionRequest).solution().status();
                System.out.println("Solution status: " + solutionStatus);

                if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
 FAILED")) {
                    break;
                }
                try {
```

```
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }

            // Once the solution is active, start creating a solution version.

            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient.createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " + solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                    .solutionVersionArn(solutionVersionArn)
                    .build();

                maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

                while (Instant.now().getEpochSecond() < maxTime) {

                    // Use the solution version ARN to get the solution version
status.
                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion()
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
                        Thread.sleep(waitInMilliseconds);
```

```
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch(PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
  solutionArn: 'SOLUTION_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSolutionVersionCommand(solutionVersionParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

現在のソリューションバージョンのステータスを確認するには、DescribeSolutionVersion オペレーションを呼び出して、CreateSolutionVersion オペレーションから返されるソリューションバージョンの ARN を渡します。status が ACTIVE であればトレーニングは完了しており、Amazon Personalize が提供するメトリクスを使用して評価できます。詳細については、「メトリクスを使用してソリューションバージョンを評価する」を参照してください。エラーが原因でトレーニングが完了しない場合、トレーニングの料金は請求されません。

ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、この StopSolutionVersionCreation 操作を使用してソリューションバージョンの作成プロセスを停止できます。ソリューションバージョンの作成の停止 を参照してください。

アクティブなソリューションバージョンをどのように使用するかは、レコメンデーションを取得する方法によって異なります。

- リアルタイムのレコメンデーションを行うには、Amazon Personalize キャンペーンを含む ACTIVE ソリューションバージョンをデプロイします。このキャンペーンを使用して、ユーザー向けのレコメンデーションを取得します。キャンペーンの作成 を参照してください。
- バッチレコメンデーションでは、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに ACTIVE ソリューションバージョンを指定します。「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

ソリューションバージョンの作成の停止

ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、Amazon Personalize コンソールまたは StopSolutionVersionCreation 操作を使用して、ソリューションバージョンの作成 (モデルのトレーニング) を停止できます。停止した後、ソリューションバージョンの作成を再開することはできません。ソリューションバージョンの作成が停止するまでに使用されたリソースについての料金が請求されます。

ソリューションバージョンの作成を停止すると、モデルトレーニングは終了しますが、ソリューションバージョンは削除されません。ソリューションバージョンの詳細は、Amazon Personalize コンソールと DescribeSolutionVersion 操作で引き続き表示できます。

ソリューションバージョンの作成プロセスは、Amazon Personalize コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して停止できます。

トピック

- ソリューションバージョンの作成の停止 (コンソール)

- ソリューションバージョンの作成の停止 (AWS CLI)
- ソリューションバージョンの作成の停止 (AWS SDK)

ソリューションバージョンの作成の停止 (コンソール)

ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、ソリューションバージョンの作成 (モデルのトレーニング) を停止できます。

ソリューションバージョンの作成を停止するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のページで、停止するソリューションバージョンのデータセットグループを選択します。
3. ナビゲーションペインで、[Solutions and recipes] (ソリューションとレシピ) を選択します。
4. [Solution and recipes] (ソリューションとレシピ) のページで、停止するソリューションバージョンのソリューションを選択します。
5. [Solution versions] (ソリューションバージョン) で、停止するソリューションバージョンを選択します。
6. ソリューションバージョンの詳細のページで、[Stop creation] (作成の停止) を選択します。ソリューションバージョンの元の状態に応じて、ソリューションバージョンの状態は次のように変化します。

   - CREATE_PENDING が CREATE_STOPPED に変わります。
   - CREATE_IN_PROGRESS が CREATE_STOPPING に変わり、次に CREATE_STOPPED に変わります。

ソリューションバージョンの作成の停止 (AWS CLI)

ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、ソリューションバージョンの作成 (モデルのトレーニング) を停止できます。次の `stop-solution-version-creation` コマンドを使用して、AWS CLI を使用したソリューションバージョンの作成を停止します。`solution version arn` を、停止するソリューションバージョンの Amazon リソースネーム (ARN) に置き換えます。ソリューションバージョンの作成が停止するまでに使用されたリソースについての料金が請求されます。

```
aws personalize stop-solution-version-creation \
```

```
    --solution-version-arn solution version arn
```

describe-solution-version コマンドを使用して、ソリューションバージョンのトレーニングステータスを確認します。

```
aws personalize describe-solution-version \
    --solution-version-arn solution version arn
```

ソリューションバージョンの元の状態に応じて、ソリューションバージョンの状態は次のように変化します。

- CREATE_PENDING が CREATE_STOPPED に変わります。
- CREATE_IN_PROGRESS が CREATE_STOPPING に変わり、次に CREATE_STOPPED に変わります

ソリューションバージョンの作成の停止 (AWS SDK)

ソリューションバージョンのステータスが CREATE_PENDING または CREATE_IN_PROGRESS の場合、ソリューションバージョンの作成 (モデルのトレーニング) を停止できます。次のコードは、AWS SDK for Python (Boto3) または AWS SDK for Java 2.x を使用してソリューションバージョンの作成を停止する方法を示しています。ソリューションバージョンの作成が停止するまでに使用されたリソースについての料金が請求されます。

SDK for Python (Boto3)

次の stop_solution_version_creation メソッドを使用して、ソリューションバージョンの作成を停止します。solution_version_arn を、停止するソリューションバージョンの Amazon リソースネーム (ARN) に置き換えます。このメソッドは、DescribeSolutionVersion 操作を使用してソリューションバージョンのステータスを取得します。

```
import boto3

personalize = boto3.client('personalize')

response = personalize.stop_solution_version_creation(
    solutionVersionArn = solution_version_arn
)
```

```
# Use the solution version ARN to get the solution version status.
solution_version_description = personalize.describe_solution_version(
    solutionVersionArn = solution_version_arn)['solutionVersion']
print('Solution version status: ' + solution_version_description['status'])
```

SDK for Java 2.x

次の stopSolutionVersionCreation メソッドを使用して、ソリューションバージョンの作成を停止します。パラメータとして、Amazon Personalize サービスクライアントと、作成を停止するソリューションバージョンの Amazon リソースネーム (ARN) を渡します。次のコードは、DescribeSolutionVersion 操作を使用してソリューションバージョンのステータスを取得します。

```
public static void stopSolutionVersionCreation(PersonalizeClient personalizeClient,
 String solutionVersionArn) {
    String solutionVersionStatus = "";

    StopSolutionVersionCreationRequest stopSolutionVersionCreationRequest =
 StopSolutionVersionCreationRequest.builder()
        .solutionVersionArn(solutionVersionArn)
        .build();


 personalizeClient.stopSolutionVersionCreation(stopSolutionVersionCreationRequest);

    // Use the solution version ARN to get the solution version status.
    DescribeSolutionVersionRequest describeSolutionVersionRequest =
 DescribeSolutionVersionRequest.builder()
        .solutionVersionArn(solutionVersionArn)
        .build();

    solutionVersionStatus =
 personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
        .solutionVersion()
        .status();
    System.out.println("Solution version status: " + solutionVersionStatus);
}
```

ソリューションバージョンの元の状態に応じて、ソリューションバージョンの状態は次のように変化します。

- CREATE_PENDING が CREATE_STOPPED に変わります。

- CREATE_IN_PROGRESS が CREATE_STOPPING に変わり、次に CREATE_STOPPED に変わります。

メトリクスを使用してソリューションバージョンを評価する

オフラインおよびオンラインのメトリクスを使用して、ソリューションバージョンのパフォーマンスを評価できます。オンラインメトリクスは、リアルタイムのレコメンデーションを使用して、ユーザーのインタラクションで観察される経験的な結果です。例えば、ユーザーがカタログを閲覧するときのユーザーのクリック率を記録できます。オンラインメトリクスの生成および記録は、ユーザーの責任において行われます。

オフラインメトリクスは、ソリューションバージョンをトレーニングするときに Amazon Personalize が生成するメトリクスです。キャンペーンを作成してレコメンデーションを提供する前に、オフラインメトリクスを使用してモデルのパフォーマンスを評価できます。オフラインメトリクスを使用すると、ソリューションのハイパーパラメータを変更した場合の効果を表示したり、同じデータでトレーニングしたモデルの結果を比較したりできます。このセクションの残りの部分では、メトリクスという用語はオフラインメトリクスを意味します。

パフォーマンスメトリクスを取得するために、Amazon Personalize は、入力インタラクションデータをトレーニングセット、テストセット、および PERSONALIZED_ACTIONS については検証セットに分割します。スプリットは選択したレシピの種類によって異なります。

- USER_SEGMENTATION レシピの場合、トレーニングセットは各ユーザーのインタラクションデータの 80% で構成され、テストセットは各ユーザーのインタラクションデータの 20% で構成されます。
- 他のすべてのレシピタイプでは、トレーニングセットは 90% のユーザーとそのインタラクションデータで構成されます。テストセットは、ユーザーとそのインタラクションデータの残りの 10% で設定されています。

その後、Amazon Personalize は、トレーニングセットを使用してソリューションバージョンを作成します。トレーニングが完了すると、Amazon Personalize は、ソリューションバージョンに、テストセットからの各ユーザーのデータの最も古い 90% を入力として提供します。その後、Amazon Personalize は、ソリューションバージョンが生成するレコメンデーションを、テストセットからの各ユーザーのデータの最新の 10% における実際のインタラクションと比較することにより、メトリクスを計算します。

比較のためのベースラインを生成するには、最も人気のある上位 K 個のアイテムを推奨する Popularity-Count レシピを使用することをお勧めします。

トピック

- ソリューションバージョンのメトリクスの取得
- メトリクスの定義
- 例
- 追加リソース

ソリューションバージョンのメトリクスの取得

ソリューションバージョンを作成すると、メトリクスを使用してそのパフォーマンスを評価できます。Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、および AWS SDKsを使用して、ソリューションバージョンのメトリクスを取得できます。

トピック

- ソリューションバージョンのメトリクスの取得 (コンソール)
- ソリューションバージョンメトリクスの取得 (AWS CLI)
- ソリューションバージョンメトリクスの取得 (AWS SDKs)

ソリューションバージョンのメトリクスの取得 (コンソール)

レコメンダーメトリクスをコンソールに表示するには、ソリューションバージョンの詳細ページに移動します。

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. データセットグループページで、カスタムデータセットグループを選択します。
3. ナビゲーションペインから、[カスタムリソース] を選択し、[ソリューションとレシピ] を選択します。
4. ソリューションを選択します。
5. [ソリューションバージョン] で、ソリューションバージョンを選択してその詳細ページを表示します。メトリックは、下部のペインの [ソリューションバージョンメトリック] タブに一覧表示されます。メトリックの定義については、「メトリクスの定義」を参照してください。

ソリューションバージョンを評価できたら、ユースケースの最高のメトリクスを使用してソリューションバージョンをデプロイすることで、キャンペーンを作成できます。ソリューションのデプロイの詳細については、「キャンペーンの作成」を参照してください。

ソリューションバージョンメトリクスの取得 (AWS CLI)

GetSolutionMetrics を呼び出して、特定のソリューションバージョンのメトリクスを取得します。次のコードは、 AWS CLIでメトリクスを取得する方法を説明しています。

```
personalize get-solution-metrics --solution-version-arn solution version ARN
```

次に示すのは、追加の最適化目標を持つ User-Personalization レシピを使用して作成されたソリューションバージョンからの出力の例です。

```
{
    "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/SolutionName/
<version-id>",
    "metrics": {
        "coverage": 0.27,
        "mean_reciprocal_rank_at_25": 0.0379,
        "normalized_discounted_cumulative_gain_at_5": 0.0405,
        "normalized_discounted_cumulative_gain_at_10": 0.0513,
        "normalized_discounted_cumulative_gain_at_25": 0.0828,
        "precision_at_5": 0.0136,
        "precision_at_10": 0.0102,
        "precision_at_25": 0.0091,
        "average_rewards_at_k": 0.653
    }
}
```

各メトリックの説明については、「メトリクスの定義」を参照してください。ソリューションバージョンを評価できたら、ユースケースの最高のメトリクスを使用してソリューションバージョンをデプロイすることで、キャンペーンを作成できます。ソリューションのデプロイの詳細については、「キャンペーンの作成」を参照してください。

ソリューションバージョンメトリクスの取得 (AWS SDKs)

GetSolutionMetrics を呼び出して、特定のソリューションバージョンのメトリクスを取得します。メトリクスを取得するには、次のコードを使用します。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.get_solution_metrics(
    solutionVersionArn = 'solution version arn')

print(response['metrics'])
```

SDK for Java 2.x

```
public static void getSolutionVersionMetrics(PersonalizeClient personalizeClient,
 String solutionVersionArn) {

    try {
        GetSolutionMetricsRequest request = GetSolutionMetricsRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .build();
        Map<String, Double> metrics =
 personalizeClient.getSolutionMetrics(request).metrics();
        metrics.forEach((key, value) -> System.out.println(key + " " + value));
    } catch (PersonalizeException e ) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

次に示すのは、追加の最適化目標を持つ User-Personalization レシピを使用して作成されたソリューションバージョンからの出力の例です。

```
{
    "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/
MovieSolution/<version-id>",
    "metrics": {
        "coverage": 0.27,
        "mean_reciprocal_rank_at_25": 0.0379,
        "normalized_discounted_cumulative_gain_at_5": 0.0405,
        "normalized_discounted_cumulative_gain_at_10": 0.0513,
        "normalized_discounted_cumulative_gain_at_25": 0.0828,
        "precision_at_5": 0.0136,
```

```
        "precision_at_10": 0.0102,
        "precision_at_25": 0.0091,
        "average_rewards_at_k": 0.653
    }
}
```

各メトリックの説明については、「メトリクスの定義」を参照してください。ソリューションバージョンを評価できたら、ユースケースの最高のメトリクスを使用してソリューションバージョンをデプロイすることで、キャンペーンを作成できます。ソリューションのデプロイの詳細については、「キャンペーンの作成」を参照してください。

メトリクスの定義

Amazon Personalize がソリューションバージョン用に生成するメトリクスは、以下の用語を使用して説明されています。

- 関連レコメンデーションとは、ユーザーが実際に操作したアイテムに関するレコメンデーションです。これらのアイテムは、テストセットからの各ユーザーのインタラクションデータの最新の 10% からのものです。
- ランクは、レコメンデーションリスト内の推奨される事項の順位を示しています。順位 1 (リストの上位) は、ユーザーにとって最も高い関連事項であると推定されます。

各メトリクスでは、数字が大きい (1 に近い) ほど優良です。さらに詳しく調べるには、「追加リソース」に記載されているリソースを参照してください。

カバレッジ

カバレッジの値は、Amazon Personalize がデータセット内の一意のレコードの合計数のうち推奨する可能性のある一意のアイテム (アイテムレコメンデーションの場合) 、アクション (アクションレコメンデーションの場合) 、またはユーザー (ユーザーセグメントレコメンデーションの場合) の割合を示します。

カバレッジスコアが高いほど、Amazon Personalize は、同じレコードを繰り返し推奨するのではなく、より多くのアイテムを推奨します。User-Personalization など、アイテム探索を特徴とするレシピは、Similar-Items など、そうでないレシピよりもカバレッジが高くなります。

25 での平均逆ランク

このメトリクスは、あるモデルがランキングの上位で関連するアイテムレコメンデーションを生成できるかどうかを示します。

あるユーザーに関連する検索結果を生成していて、ユーザーがリストの下位にあるアイテムを選択することを期待していない場合、25 での平均逆ランクが高いモデルを選択できます。例えば、ユーザーは検索結果の最初の料理レシピを選ぶことがよくあります。Amazon Personalize は、PERSONALIZED_ACTIONS または USER_SEGMENTATION レシピでは、このメトリクスを生成しません。

Amazon Personalize は、レコメンデーションのリクエストの平均逆ランクスコアを使用してこのメトリックスを計算します。それぞれの相互ランクスコアは次のように計算されます。`1 / the rank of the highest item interacted with by the user`、ここで可能なランキングの合計は 25 です。ユーザーが操作するその他のランクの低いアイテムは無視されます。ユーザーが最初のアイテムを選択した場合、スコアは 1 です。アイテムを何も選択しなかった場合、スコアは 0 です。

例えば、3 人の異なるユーザーに、それぞれ 25 件のレコメンデーションを表示するとします。

- ユーザー 1 がランク 4 のアイテムとランク 10 のアイテムをクリックした場合、両者の相互ランクスコアは 1/4 になります。
- ユーザー 2 がランク 2 のアイテム、ランク 4 のアイテム、ランク 12 のアイテムをクリックした場合、相互のランクスコアは 1/2 になります。
- ユーザー 3 がランク 6 のアイテムを 1 つクリックした場合、相互のランクスコアは 1/6 になります。

すべてのレコメンデーションのリクエスト (この場合は 3) の平均相互ランクは、`(1/4 + 1/2 + 1/6) / 3 = .3056` として計算されます。

K (5/10/25) での正規化減損累積利得 (NDCG)

このメトリクスは、モデルがアイテムまたはアクションレコメンデーションをどのようにランク付けしているかを示しています。ここで、K は 5、10、または 25 のレコメンデーションのサンプルサイズです。このメトリクスは、最もランクの高いアイテムまたはアクションだけでなく、レコメンデーションのランク付けに最も関心がある場合に便利です (これについては、「」を参照してくださいmean reciprocal rank at 25) 。例えば、1 つのカルーセルに同時に最大 10 本の映画を表示するアプリケーションでは、NDCG at 10 のスコアが役に立ちます。

Amazon Personalize は、テストセット内の各ユーザーのランキング順位に基づいてレコメンデーションに重みを付けることで NDCG を計算します。各レコメンデーションはそれぞれの順位による要因で割引 (低い分量の付与) されます。最後のメトリクスは、テストセットNDCG at K内のすべてのユーザーの平均です。は、リストの下位にあるレコメンデーションは、リストの上位にあるレコメンデーションよりも関連性が低いことをNDCG at K前提としています。

Amazon Personalize は、リストの上位が順位 1 である 1/log(1 + position) の重量要素を使用します。

precision at K

K (5、10、25) の推奨サンプルサイズに基づいて、モデルのレコメンデーションがどの程度関連しているかを示すメトリクス。

Amazon Personalize では、このメトリクスは、上位 K 個のレコメンデーションのうち、関連するレコメンデーションの数を K で割った数に基づいて計算されます (K は 5、10、または 25 です)。最後のメトリクスは、テストセットに含まれる全ユーザーの平均です。

例えば、あるユーザーに 10 個のアイテムをレコメンドし、ユーザーがそのうちの 3 個とインタラクションした場合、K での精度は、正しく予測された 3 個のアイテムを、合計 10 個のレコメンデーションアイテムで割った値、つまり 3 / 10 = .30 です。

このメトリクスでは、関連事項の正確なレコメンデーションが評されます。スコアが 1 に近いほど、モデルの精度は高くなります。

precision

Next-Best-Action レシピを使用してソリューションバージョンをトレーニングすると、Amazon Personalize は precision at K の代わりに precision メトリクスを生成します。このメトリクスは、ユーザーが実際に実行するアクションを予測するうえで、モデルがどの程度優れているかを示します。

Amazon Personalize は、データセット内の各アクションについて、precision を計算するために、そのアクションを実行すると正しく予測されたユーザーの数を、アクションが推奨された合計回数で割って計算します。その後、Amazon Personalize はデータセット内のすべてのアクションの平均を計算します。

例えば、あるアクションが 100 人のユーザーに推奨され、60 人のユーザーがそのアクションを実行し、40 人のユーザーが実行しなかった場合、そのアクションの precision は 60 / 100 = .60 です。その後、Amazon Personalize はこの計算をすべてのアクションに適用し、平均を返します。

このメトリクスでは、関連するアクションの正確なレコメンデーションが評価されます。スコアが 1 に近いほど、モデルの精度は高くなります。

average_rewards_at_k

最適化の目的を持つソリューションのソリューションバージョンを作成する (モデルをトレーニングする) と、Amazon Personalize は average_rewards_at_k メトリクスを生成しま

す。average_rewards_at_k のスコアは、ソリューションバージョンが目的を達成する上でどれだけよく機能するかを示します。このメトリクスを計算するために、Amazon Personalize は各ユーザーについての報酬を次のように計算します。

```
rewards_per_user = total rewards from the user's interactions with their top 25 reward generating recommendations / total rewards from the user's interactions with recommendations
```

最終的な average_rewards_at_k は、1 以下 0 超の 10 進値に正規化されたすべての rewards_per_user の平均です。値が 1 に近いほど、レコメンデーションから期待できるユーザーあたりの平均利益が大きくなります。

例えば、クリックから得られる収益を最大化することが目的の場合、Amazon Personalize は、ユーザーが上位 25 位の最も高額なレコメンデーションからクリックしたアイテムによって生成された合計収益を、ユーザーがクリックしたすべての推奨アイテムからの収益で除することにより、各ユーザースコアを計算します。その後、Amazon Personalize は、すべてのユーザースコアの正規化された平均を返します。average_rewards_at_k が 1 に近いほど、レコメンデーションから得られるユーザーあたりの平均収益が増大することが期待できます。

詳細については、「追加の目的のためのソリューションの最適化」を参照してください。

トレンド予測精度

ソリューションバージョンを トレンド-ナウ レシピでトレーニングした場合、モデルが推奨するアイテムの人気上昇率。トレンド予測の精度が高いほど (1 に近いほど)、モデルはトレンドのアイテムをより正確に特定できます。

人気の加速率を計算するために、Amazon Personalize では、すべてのおすすめ商品の人気の増加率を、トレンド商品の上位 25 件の合計人気上昇率で割ります。これらの項目は、テストセットでの実際のインタラクションに基づいています。

データ分布や [トレンド発見頻度] で何を選択したかによって、トレンド予測の精度の値は 0.0 になる場合があります。

ヒット (K でのヒット)

USER_SEGMENTATION レシピを使用してソリューションバージョンをトレーニングした場合、予測された上位 K 件の結果の中で、実際のユーザーと一致するユーザーの平均数が表示されます。実際のユーザーとは、テストセット内の項目を実際に操作したユーザーです。K は、最も関連性の高いユーザーの上位 1% です。値が大きいほど、予測の精度が高くなります。

リコール (K でのリコール)

USER_SEGMENTATION レシピを使用してソリューションバージョンをトレーニングした場合、関連性の高い予測結果の K 件のうち、実際のユーザーと一致する予測ユーザーの平均パーセンテージです。実際のユーザーとは、テストセット内の項目を実際に操作したユーザーです。K は、最も関連性の高いユーザーの上位 1% です。値が大きいほど、予測の精度が高くなります。

リコール

Next-Best-Action レシピを使用してソリューションバージョンをトレーニングした場合、このメトリクスは、ユーザーが操作するアクションを検出するうえで、ソリューションバージョンがどの程度優れているかを示します。

Amazon Personalize は、データセット内の各アクションについて、`recall` を計算するために、そのアクションを実行すると正しく予測されたユーザーの数を、テストセット内でアクションを実際に実行した合計ユーザー数で割って計算します。その後、Amazon Personalize はデータセット内のすべてのアクションの平均を計算します。

例えば、100 人のユーザーがテストセットでアクションを実行し、Amazon Personalize がこれらのユーザーのうち 50 人がアクションを実行すると予測した場合、アクションの `recall` は `50 / 100 = .50` です。その後、Amazon Personalize はこの計算をすべてのアクションに適用し、平均を返します。

曲線の下の面積 (AUC)

PERSONALIZED_ACTIONS レシピを使用してソリューションバージョンをトレーニングした場合、ソリューションバージョンの受信者操作特性曲線の下の面積。このメトリクスは、ユーザーが実行するアクションを正しく識別するうえで、ソリューションバージョンがどの程度効果があるかを示します。

受信者操作特性曲線は、ソリューションバージョンのパフォーマンスを表します。さまざまなしきい値における真陽性 (関連があると正しく予測されたアクション) と偽陽性 (関連があると誤って予測されたアクション) の率をプロットしています。曲線の下の面積 (AUC) は、ソリューションバージョンのパフォーマンスを曲線に基づいて集計したスコアです。

ソリューションバージョンの AUC は、0 から 1 までです。1 に近いほど、モデルはユーザーに関連するアクションを予測するうえで優れています。

例

以下に示しているのは、特定のユーザー用にレコメンデーションのリストを作成するソリューションバージョンのシンプルな例です。2 番目と 5 番目のレコメンデーションは、このユーザーのテストデータのレコードと一致します。これらは関連するレコメンデーションです。K が 5 に設定されている場合、次のメトリクスがユーザーに生成されます。

reciprocal_rank

計算: 1/2

結果: 0.5000

normalized_discounted_cumulative_gain_at_5

計算: (1/log(1 + 2) + 1/log(1 + 5)) / (1/log(1 + 1) + 1/log(1 + 2))

結果: 0.6241

precision_at_5

計算: 2/5

結果: 0.4000

追加リソース

A/B テストによるソリューションバージョンの評価については、「A/B テストを使用して Amazon Personalize によって生成されたレコメンデーションの有効性を測定する」を参照してください。レコメンダーシステムのさまざまなメトリクスについて詳しくは、以下の外部リソースを参照してください。

- MRR と MAP と NDCG の比較:ランクに応じた評価メトリクスとそれをいつ使うべきか
- 割引後の累積利益:知っておくべきランキングメトリクス
- レコメンダーシステムの k での再現率と精度
- レコメンダーシステムのランク付け評価メトリクス
- 受信者操作特性

キャンペーンの作成

カスタムリソースを使ったリアルタイムのレコメンデーションについては、ソリューションバージョンの作成 を完了すると、ソリューションバージョンをキャンペーンでデプロイする準備が整います。

キャンペーンでは、アプリケーションユーザー向けのリアルタイムレコメンデーションを生成するためのプロビジョンドトランザクション容量を搭載した、ソリューションバージョン (トレーニング済みモデル) がデプロイされます。キャンペーンを作成したら、GetRecommendations または GetPersonalizedRanking の API 操作を使用してレコメンデーションを取得します。バッチレコメンデーションを取得しようとしている場合は、キャンペーンを作成する必要はありません。詳細については、「バッチレコメンデーションとユーザーセグメント (カスタムリソース)」を参照してください。

キャンペーンを作成するときに、以下を設定できます。

- ソリューションの最新のソリューションバージョンを使用するように自動的に更新するようにキャンペーンを設定できます。詳細については、自動キャンペーン更新の有効化を参照してください。
- レコメンデーションでアイテムメタデータを有効にできます。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。
- キャンペーンの 1 秒あたりのプロビジョニングされた最小トランザクション数を指定できます。これは Amazon Personalize によってプロビジョニングされたキャンペーンのベースライントランザクションスループットです。キャンペーンがアクティブである間の最低請求額を設定します。詳細については、「1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング」を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用してキャンペーンを作成できます。レコメンデーションでメタデータを有効にするなど、既存のキャンペーンの設定を変更する場合は、キャンペーンを更新する必要があります。詳細については、キャンペーンの更新を参照してください。

キャンペーンがアクティブになっている間は、キャンペーンコストが発生します。不要なコストを回避するには、完了したらキャンペーンを削除してください。キャンペーンのコストについては、「Amazon Personalize の料金」を参照してください。

トピック

- 自動キャンペーン更新の有効化

- 1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング
- レコメンデーションのアイテムメタデータ
- キャンペーンの作成 (コンソール)
- キャンペーンの作成 (AWS CLI)
- キャンペーンの作成 (AWS SDKs)
- キャンペーンの更新

自動キャンペーン更新の有効化

キャンペーンを作成するときに、キャンペーンの自動更新を有効にできます。自動更新では、キャンペーンは自動的に更新され、ソリューションの最新の自動または手動でトレーニングされたソリューションバージョンがデプロイされます。これにより、キャンペーンを最新の状態に保つことが容易になります。

例えば、ソリューションが自動トレーニングを使用して 7 日ごとに新しいソリューションバージョンを作成する場合、キャンペーンは毎週のトレーニングごとに最新のソリューションバージョンを使用するように自動的に更新されます。自動キャンペーン更新を使用しない場合は、キャンペーンを手動で更新して、最新のトレーニング済みモデルをデプロイする必要があります。

- Amazon Personalize コンソールでキャンペーンを作成するときに自動キャンペーン更新を有効にするには、キャンペーンの詳細 でソリューションの最新バージョンを使用するように自動更新を選択します。最新の更新のタイムスタンプは、キャンペーンの詳細ページで確認できます。

  詳細については、「キャンペーンの作成 (コンソール)」を参照してください。
- CreateCampaign API オペレーションの使用時にキャンペーンの自動更新を有効にするには、 `SolutionVersionArn`パラメータにソリューションの Amazon リソースネーム (ARN) を `SolutionArn/$LATEST`形式で指定します。で`campaignConfig`、`enableMetadataInInferenceResponse`を に設定します`true`。

  最新のキャンペーン更新のタイムスタンプを取得するには、 DescribeCampaign API オペレーションを使用してレスポンス`latestCampaignUpdate`の詳細を確認します。

  自動更新を有効にする方法を示すコードサンプルについては、キャンペーンの作成 (AWS CLI)「」または「」を参照してくださいキャンペーンの作成 (AWS SDKs)。

1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング

> **Important**
>
> minProvisionedTPS の値を高く設定するとコストが増加します。最初は minProvisionedTPS に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて を増やします。

Amazon Personalize のキャンペーンを作成する場合、キャンペーンの 1 秒あたりのプロビジョニングされる最小トランザクション数 (minProvisionedTPS) を指定できます。これは Amazon Personalize によってプロビジョニングされたキャンペーンのベーストライントランザクションスループットです。キャンペーンがアクティブである間の最低請求額を設定します。トランザクションは単一の GetRecommendations または GetPersonalizedRanking リクエストです。デフォルト minProvisionedTPS は 1 です。

TPS が minProvisionedTPS を超えて増加した場合、Amazon Personalize はプロビジョンド容量を自動スケーリングしますが、minProvisionedTPS を下回ることはありません。容量が引き上げられている間に短時間の遅延が生じます。これにより、トランザクションの損失が生じる可能性があります。トラフィックが減少すると、容量は minProvisionedTPS に戻ります。

プロビジョニングされた最小 TPS、またはリクエストが を超えた場合はminProvisionedTPS実際の TPS に対して課金されます。実際の TPS は、行ったレコメンデーションリクエストの総数です。低い から始めてminProvisionedTPS、Amazon CloudWatch メトリクスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて を増やすことをお勧めします。

キャンペーンのコストの詳細については、「Amazon Personalize の料金」を参照してください。

レコメンデーションのアイテムメタデータ

> **Important**
>
> User-Personalization-v2 または Personalized-Ranking-v2 レシピを使用する場合、メタデータに追加コストは発生しません。他のすべてのレシピとすべてのドメインのユースケースでは、追加料金が発生します。詳細については、「Amazon Personalize の料金」を参照してください。

レコメンデーションを取得すると、Amazon Personalize にレコメンデーション結果にアイテムメタデータを含めることができます。リクエストでは、含める Items データセットから列を選択できます。Amazon Personalize は、レコメンデーションレスポンス内の各アイテムについてこのデータを返します。

メタデータを使用して、映画のジャンルをカルーセルに追加するなど、ユーザーインターフェイスのレコメンデーションを充実させることができます。あるいは、レコメンデーションの質を視覚的に評価するのにも使えます。アプリで生成 AI を使用している場合は、メタデータを AI プロンプトに組み込んで、より関連性の高いコンテンツを生成できます。Amazon Personalize の生成 AI の使用に関する詳細については、「Amazon Personalize と生成 AI」を参照してください。

メタデータの有効化

レコメンデーションにメタデータを追加するには、メタデータの列を含むアイテムデータセットが必要です。トレーニングではメタデータを使用する必要はありません。データセットの作成については、「データセットとスキーマの作成」を参照してください。データの管理と更新については、「データセット内のトレーニングデータの管理」を参照してください。

User-Personalization-v2 または Personalized-Ranking-v2 レシピを使用する場合、キャンペーンにはアイテムメタデータをレコメンデーション結果に含めるオプションが自動的に用意されます。キャンペーンのメタデータを手動で有効にすることはできません。他のすべてのレシピとドメインのユースケースでは、メタデータオプションを有効にする必要があります。

- Amazon Personalize コンソールでメタデータを有効にするには、キャンペーンを作成するときに、[キャンペーン詳細] の [レコメンデーション結果にあるアイテムのメタデータを返す] を選択します。詳細については、「キャンペーンの作成 (コンソール)」を参照してください。
- AWS SDKs または でメタデータを有効にするには AWS CLI、 CreateCampaign API オペレーション および を `campaignConfigenableMetadataInInferenceResponse`に設定します`true`。詳細については、キャンペーンの作成 (AWS CLI)またはキャンペーンの作成 (AWS SDKs)を参照してください。

キャンペーンの作成 (コンソール)

> **Important**
>
> キャンペーンがアクティブになっている間は、キャンペーンコストが発生します。不要なコストを回避するには、完了したらキャンペーンを削除してください。キャンペーンのコストについては、「Amazon Personalize の料金」を参照してください。

ソリューションバージョンのステータスが [Active] (アクティブ) になったら、Amazon Personalize のキャンペーンでそのソリューションバージョンをデプロイする準備が整います。

キャンペーンを作成するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. デプロイするソリューションバージョンのデータセットグループを選択します。
3. ナビゲーションペインの [カスタムリソース] で、[キャンペーン] を選択します。
4. [Campaigns] (キャンペーン) のページで、[Create campaign] (キャンペーンを作成) を選択します。
5. [Create new campaign] (新しいキャンペーンを作成) ページの [Campaign details] (キャンペーンの詳細) で、次の情報を入力します。

   - キャンペーン名 - キャンペーンの名前を入力します。ここで入力したテキストは、[Campaign] (キャンペーン) ダッシュボードと詳細のページに表示されます。
   - ソリューション - 先ほど作成したソリューションを選択します。
   - ソリューションの最新のソリューションバージョンを使用するように自動的に更新する – キャンペーンで最新のアクティブなソリューションバージョンを自動的に使用するには、このオプションを選択します。これを選択しない場合は、新しいソリューションバージョンをデプロイするたびにキャンペーンを手動で更新する必要があります。詳細については、「自動キャンペーン更新の有効化」を参照してください。
   - ソリューションバージョン ID – 自動キャンペーン更新を使用して最新のソリューションバージョンを使用しない場合は、デプロイするソリューションバージョンの ID を選択します。
   - 1 秒あたりの最小プロビジョンドトランザクション (API では minProvisionedTPS と呼ばれます) - Amazon Personalize がサポートしている 1 秒あたりの最小プロビジョンドトランザクションを設定します。値を大きくすると、料金が増加します。1 (デフォルト) から始めることをお勧めします。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、`minProvisionedTPS`必要に応じて を増やします。詳細については、「1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング」を参照してください。
   - レコメンデーション結果にアイテムメタデータを返す – レコメンデーション結果にメタデータを含める場合は、このオプションを選択します。有効にすると、レコメンデーションを取得するときに Items データセットの列を指定できます。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。

6. User-Personalization レシピを使用した場合は、キャンペーン設定 で、オプションで探索の重みと探索項目の経過時間カットオフ の値を入力できます。詳細については、「User-Personalization」を参照してください。
7. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
8. [キャンペーンの作成] を選択します。
9. キャンペーンの詳細のページで、キャンペーンのステータスが [Active] (アクティブ) の場合、キャンペーンを使用してレコメンデーションを取得し、インプレッションを記録できます。詳細については、「ステップ 4: レコメンデーションを取得する」を参照してください。

   キャンペーンのステータスが ACTIVE になると、キャンペーンの準備が整います。ソリューションバージョンを再トレーニングする場合、またはキャンペーン設定を変更する場合は、キャンペーンを更新する必要があります。詳細については、「キャンペーンの更新」を参照してください。

キャンペーンの作成 (AWS CLI)

> **Important**
>
> キャンペーンがアクティブになっている間は、キャンペーンコストが発生します。不要なコストを回避するには、完了したらキャンペーンを削除してください。キャンペーンのコストについては、「Amazon Personalize の料金」を参照してください。

ソリューションバージョンがアクティブになったら、Amazon Personalize キャンペーンでデプロイする準備が整います。でキャンペーンを作成するには AWS CLI、 `create-campaign` コマンドを使用します。

次のコードサンプルは、キャンペーンを作成する方法を示しています。User-Personalization レシピを使用するソリューションの最新バージョンをデプロイします。作成するキャンペーンは、将来のソリューションバージョンを使用するように自動的に更新されます。このコードでは、次の設定を使用します。

- ソリューションの最新のソリューションバージョンを使用するように自動的に更新するようにキャンペーンを設定します。 `solution-version-arn`は *`solution ARN`*/$LATEST 形式、 は `syncWithLatestSolutionVersion`です`True`。コードを使用するには、 をソリューションの Amazon リソースネーム (ARN) `solution ARN`に置き換えます。

自動 を無効にするにはsyncWithLatestSolutionVersion、ソリューションバージョン ARN (なし/$LATEST) のみを指定し、 syncWithLatestSolutionVersionを に設定しますFalse。

- enableMetadataWithRecommendations オプションを に設定しますTrue。これにより、レコメンデーションリクエストオプションにアイテムデータセットのアイテムメタデータをレコメンデーション結果に含めることができます。このオプションを無効にするには、 に設定しますFalse。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。
- min-provisioned-tps は 1 (デフォルト) に設定されます。最初は minProvisionedTPS に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて を増やします。詳細については、「1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング」を参照してください。

すべてのパラメータの詳細なリストについては、「CreateCampaign」を参照してください。

```
aws personalize create-campaign \
--name campaign-name \
--solution-version-arn solution-arn/\$LATEST \
--min-provisioned-tps 1 \
--campaign-config "{"\"syncWithLatestSolutionVersion"\": "true",
 "\"enableMetadataWithRecommendations"\": "true"}"
```

キャンペーンのステータスが ACTIVE になると、キャンペーンの準備が整います。現在のステータスを取得するには、DescribeCampaign を呼び出して、status フィールドが ACTIVE であることを確認します。

ソリューションバージョンを再トレーニングし、最新のソリューションバージョンを使用するようにキャンペーンが自動的に更新されない場合、またはキャンペーン設定を変更する場合は、キャンペーンを更新する必要があります。詳細については、「キャンペーンの更新」を参照してください。

Amazon Personalize では、作成したキャンペーンを一覧表示ListCampaignsするなど、キャンペーンを管理するためのオペレーションが提供されます。キャンペーンを削除するには、DeleteCampaign を呼び出します。キャンペーンを削除しても、キャンペーンの一部であるソリューションバージョンは削除されません。

キャンペーンを作成したら、それを使用してレコメンデーションを作成できます。詳細については、「ステップ 4: レコメンデーションを取得する」を参照してください。

キャンペーンの作成 (AWS SDKs)

> **Important**
>
> キャンペーンがアクティブになっている間は、キャンペーンコストが発生します。不要なコストを回避するには、完了したらキャンペーンを削除してください。キャンペーンのコストについては、「Amazon Personalize の料金」を参照してください。

ソリューションバージョンがアクティブになったら、Amazon Personalize キャンペーンでデプロイする準備が整います。 AWS SDKsオペレーションを使用します。 CreateCampaign

次のコードサンプルは、キャンペーンを作成する方法を示しています。このコードは、User-Personalization レシピを使用するソリューションの最新バージョンをデプロイします。作成するキャンペーンは、将来のソリューションバージョンを使用するように自動的に更新されます。このコードでは、次の設定を使用します。

- ソリューションの最新のソリューションバージョンを使用するように自動的に更新するようにキャンペーンを設定します。 solutionVersionArnは *solution ARN*/$LATEST 形式、 は syncWithLatestSolutionVersionですTrue。コードを使用するには、 をソリューションバージョンの Amazon リソースネーム (ARN) solution ARNに置き換えます。

  自動 を無効にするにはsyncWithLatestSolutionVersion、ソリューションバージョン ARN (なし/$LATEST) のみを指定し、 syncWithLatestSolutionVersionを に設定しますFalse。
- enableMetadataWithRecommendations オプションを に設定しますTrue。これにより、レコメンデーションリクエストオプションにアイテムデータセットのアイテムメタデータをレコメンデーション結果に含めることができます。このオプションを無効にするには、 に設定しますFalse。詳細については、「レコメンデーションのアイテムメタデータ」を参照してください。
- minProvisionedTPS は 1 (デフォルト) に設定されます。 (minProvisionedTPSデフォルト) には 1 から始めることをお勧めします。Amazon CloudWatch メトリクスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて を増やします。詳細については、「1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング」を参照してください。

すべてのパラメータの詳細なリストについては、「CreateCampaign」を参照してください。

SDK for Python (Boto3)

```
import boto3
```

```
personalize = boto3.client('personalize')

response = personalize.create_campaign(
    name = 'campaign name',
    solutionVersionArn = 'solution ARN/$LATEST',
    minProvisionedTPS = 1,
    campaignConfig = {"syncWithLatestSolutionVersion": True,
 "enableMetadataWithRecommendations": True}
)

arn = response['campaignArn']

description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
print('Status: ' + description['status'])
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateCampaignCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({ region: "REGION" });

// set the campaign parameters
export const createCampaignParam = {
  solutionVersionArn: "SOLUTION_ARN/$LATEST" /* required */,
  name: "NAME" /* required */,
  minProvisionedTPS: 1 /* optional */,
  campaignConfig: {   /* optional */
    syncWithLatestSolutionVersion: true,
    enableMetadataWithRecommendations: true,
  },
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateCampaignCommand(createCampaignParam)
    );
```

```
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

キャンペーンのステータスが ACTIVE になると、キャンペーンの準備が整います。現在のステータスを取得するには、 を呼び出しDescribeCampaign、 statusフィールドが であることを確認しますACTIVE。

ソリューションバージョンを手動で再トレーニングする場合、またはキャンペーン設定を変更する場合は、キャンペーンを更新する必要があります。詳細については、「キャンペーンの更新」を参照してください。

Amazon Personalize では、作成したキャンペーンを一覧表示ListCampaignsするなど、キャンペーンを管理するためのオペレーションが提供されます。キャンペーンを削除するには、DeleteCampaign を呼び出します。キャンペーンを削除しても、キャンペーンの一部であるソリューションバージョンは削除されません。

キャンペーンを作成したら、これを使用して推奨を行うことができます。詳細については、「ステップ 4: レコメンデーションを取得する」を参照してください。

キャンペーンの更新

再トレーニングされたソリューションバージョンを既存のキャンペーンでデプロイするか、レコメンデーションでメタデータを有効にするなど、キャンペーンの 最小プロビジョンド TPS またはキャンペーン設定を変更するには、キャンペーンを手動で更新する必要があります。

User-Personalization-v2、User-Personalization、または Next-Best-Action を使用すると、Amazon Personalize は 2 時間ごとに最新のソリューションバージョン (を trainingModeに設定してトレーニング済みFULL) を自動的に更新し、新しいアイテムやアクションをレコメンデーションに含めます。キャンペーンは自動的に更新されたソリューションバージョンを使用します。trainingMode を FULL に設定してソリューションバージョンを手動で再トレーニングする場合、またはキャンペーンの minProvisionedTPS またはキャンペーン設定を変更する場合にのみ、キャンペーンを手動で更新します。自動更新については、「自動更新」を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs。

トピック

- キャンペーンの更新 (コンソール)
- キャンペーンの更新 (AWS CLI)
- キャンペーンの更新 (AWS SDKs)

キャンペーンの更新 (コンソール)

手動で再トレーニングされたソリューションバージョンをデプロイしたり、キャンペーン設定を変更したりするには、キャンペーンを更新する必要があります。

キャンペーンを更新するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. 更新するキャンペーンのデータセットグループを選択します。
3. ナビゲーションペインで、[キャンペーン] を選択します。
4. [Campaigns] (キャンペーン) のページで、更新するキャンペーンを選択します。
5. キャンペーンの詳細のページで、[Update] (更新) を選択します。
6. [Update campaign] (キャンペーンを更新) のページで、変更を加えます。例えば、再トレーニングされたソリューションバージョンをデプロイする場合、[Solution version ID] (ソリューションバージョン ID) で、新しいソリューションバージョンの識別番号を選択します。
7. [更新] を選択します。Amazon Personalize は、新しいソリューションバージョンと変更された設定を使用するようにキャンペーンを更新します。

キャンペーンの更新 (AWS CLI)

新しいソリューションバージョンをデプロイしたり、キャンペーンの 最小プロビジョンド TPS を変更したり、キャンペーンの設定を変更したりするには、キャンペーンを更新する必要があります。次の `update-campaign` コマンドを使用してキャンペーンを更新し、 AWS CLIで新しいソリューションバージョンを使用します。

`campaign arn` を、更新するキャンペーンの Amazon リソースネーム (ARN) に置き換えます。`new solution version arn` を、デプロイするソリューションバージョンに置き換えます。

```
aws personalize update-campaign \
--campaign-arn campaign arn \
--solution-version-arn new solution version arn \
```

```
--min-provisioned-tps 1
```

キャンペーンの更新 (AWS SDKs)

新しいソリューションバージョンをデプロイしたり、キャンペーンの 最小プロビジョンド TPS を変更したり、キャンペーンの設定を変更したりするには、キャンペーンを更新する必要があります。次のコードを使用して、SDK for Python (Boto3) または SDK for Java 2.x でキャンペーンを更新します。パラメータの詳細なリストについては、「UpdateCampaign」を参照してください。

SDK for Python (Boto3)

次の update_campaign メソッドを使用して、新しいソリューションバージョンをデプロイします。campaign arn を更新するキャンペーンの Amazon リソースネーム (ARN) に、new solution version arn を新しいソリューションバージョンの ARN に、それぞれ置き換えます。また、オプションで minProvisionedTPS を変更します。

```
import boto3

personalize = boto3.client('personalize')

response = personalize.update_campaign(
    campaignArn = 'campaign arn',
    solutionVersionArn = 'new solution version arn',
    minProvisionedTPS = 1,
)

arn = response['campaignArn']

description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

次の updateCampaign メソッドを使用して、新しいソリューションバージョンを使用するようにキャンペーンを更新します。パラメータとして、Amazon Personalize のサービスクライアント、新しいソリューションバージョンの Amazon リソースネーム (ARN)、および 最小プロビジョンド TPS を渡します。

```
public static void updateCampaign(PersonalizeClient personalizeClient,
```

```
                                String campaignArn,
                                String solutionVersionArn,
                                Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
 UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
              .campaignArn(campaignArn)
              .build();

        DescribeCampaignResponse campaignResponse =
 personalizeClient.describeCampaign(campaignRequest);
        Campaign updatedCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " + updatedCampaign.status());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

ステップ 4: レコメンデーションを取得する

リソースに応じて、リアルタイムで、またはバッチワークフローでレコメンデーションを取得できます。

- カスタムリソースを使用すると、リアルタイムのレコメンデーションまたはバッチレコメンデーションを取得できます。リアルタイムのレコメンデーションについては、レコメンデーションを取得する前にキャンペーンを作成する必要があります。バッチレコメンデーションについては、キャンペーンを作成する必要はありません。

- ドメインデータセットグループ内のレコメンデーションでは、リアルタイムのレコメンデーションのみを取得できます。

次のトピックでは、各レコメンデーションタイプを使用する方法とタイミングについて説明します。

トピック

レコメンデーションスコア

User-Personalization-v2、User-Personalization、Personalized-Ranking-v2、Personalized-Ranking、および PERSONALIZED_ACTIONS レシピで作成されたカスタムソリューションでは、Amazon Personalize は各アイテムのスコアをレコメンデーションに含めます。これらのスコアは、ユーザーが次にどのアイテムまたはアクションを選択するかについて、Amazon Personalize が持っている相対的な確実性を表しています。スコアが高いほど、確実性が高くなります。

- User-Personalization-v2 と User-Personalization のスコアについては、「」を参照してくださいレコメンデーションスコアリングの仕組み (カスタムリソース) 。
- PERSONALIZED_ACTIONS レシピのスコアについては、「アクションレコメンデーションのスコアの仕組み」を参照してください。
- Personalized-Ranking-v2 および Personalized-Ranking のレコメンデーションのスコアについては、「」を参照してくださいパーソナライズされたランキングスコアリングの仕組み。

バッチ推論ジョブでは、レコメンデーションスコアリングの仕組み (カスタムリソース) および パーソナライズされたランキングスコアリングの仕組み で説明されているように項目スコアが計算されます。バッチ推論ジョブの出力 JSON ファイルでスコアを表示できます。

リアルタイムレコメンデーションの取得

リアルタイムレコメンデーションとは、ユーザーがアプリケーションを使用する際にリクエストして表示するレコメンデーションです。レコメンダーを使用して (ドメインデータセットグループ向け) またはカスタムキャンペーンを使用して、リアルタイムレコメンデーションをAmazon Personalize から取得できます。

- ドメインレコメンダーでは、the section called “GetRecommendations” 操作でリアルタイムのレコメンデーションを取得できます。または、Amazon Personalize コンソールを使用してレコメンダーをテストできます。
- カスタムリソースでは、キャンペーンをサポートするソリューションバージョンの作成に使用したレシピに応じて、the section called “GetRecommendations”、GetActionRecommendations、または the section called “GetPersonalizedRanking” API 操作を使用して、ユーザー向けのレコメンデーションを取得します。または、Amazon Personalize コンソールを使用してレコメンダーをテストできます。

Top Picks for You ユースケースや User-Personalization レシピなど、リアルタイムのパーソナライゼーションを提供するドメインユースケースまたはレシピを使用する場合、Amazon Personalize は、カタログ操作の記録に関する最新のアクティビティに基づいてレコメンデーションを更新します。リアルタイムイベントの記録についての詳細は、「イベントの記録」を参照してください。

レコメンデーションアイテムに対してメタデータを返すようにキャンペーンを設定した場合は、GetRecommendations または GetPersonalizedRanking API オペレーションに含める列を指定できます。または、Amazon Personalize コンソールを使用してレコメンダーをテストするときに列を指定できます。

一部のユースケースやレシピでは、リクエストにプロモーションを指定できます。プロモーションは、設定可能なおすすめアイテムのサブセットに適用される追加のビジネスルールを定義します。詳細については、レコメンデーション内のアイテムのプロモーションを参照してください。

トピック

- アイテムレコメンデーションの取得
- アクションレコメンデーションの取得
- パーソナライズされたランキングの取得 (カスタムリソース)
- コンテキストメタデータを使用したレコメンデーションの関連性の向上

アイテムレコメンデーションの取得

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用して、Amazon Personalize レコメンダーまたはカスタムキャンペーンからアイテムレコメンデーションを取得できます。

> **Note**
>
> PERSONALIZED_RANKING レシピを使用した場合は、「パーソナライズされたランキングの取得 (カスタムリソース)」を参照してください。

トピック

- レコメンデーションスコアリングの仕組み (カスタムリソース)
- レコメンデーションの理由 (User-Personalization-v2)
- アイテムレコメンデーションの取得 (コンソール)
- アイテムレコメンデーションの取得 (AWS CLI)
- アイテムレコメンデーションの取得 (AWS SDKs)
- レコメンデーション内のアイテムのプロモーション

レコメンデーションスコアリングの仕組み (カスタムリソース)

User-Personalization-v2 レシピと User-Personalization レシピを使用すると、Amazon Personalize はユーザーのインタラクションデータとメタデータに基づいてアイテムのスコアを生成します。これらのスコアは、ユーザーが次に選択するアイテムがある、Amazon Personalize の相対的な確実性を表します。スコアが高いほど、確実性が高くなります。

> **Note**
>
> Amazon Personalize は、ドメインレコメンデーションや、類似商品、SIMS、Popularity-Count のレシピのスコアを表示しません。パーソナライズドランキングのレコメンデーションのスコアについては、「パーソナライズされたランキングスコアリングの仕組み」を参照してください。

Amazon Personalize は、0 から 1 のスケールで (両方を含む)、相互に関連する項目のスコアを生成します。User-Personalization-v2 を使用すると、Amazon Personalize はアイテムのサブセットのスコアを生成します。User-Personalization を使用すると、Amazon Personalize はカタログ内のすべての項目をスコアリングします。

User-Personalization-v2 を使用してレコメンデーションにフィルターを適用する場合、フィルターが削除するレコメンデーションの数に応じて、Amazon Personalize はプレースホルダー項目を追加す

ることがあります。これは、レコメンデーションリクエストnumResultsの を満たすために行われます。これらの項目は、フィルター基準を満たすインタラクションデータの量に基づいて人気のある項目です。ユーザーには関連性スコアがありません。

User-Personalization-v2 と User-Personalization の両方で、すべてのスコアの合計は 1 に等しくなります。例えば、ユーザーに対して映画のレコメンデーションを取得し、アイテムデータセットとインタラクションデータセットに 3 つの映画が表示されている場合、そのスコアは 0.6、0.3、およびになります0.1。同様に、インベントリに 10,000 本の映画がある場合、最高スコアの映画のスコアは非常に小さい可能性がありますが (平均スコアは .001)、スコアが相対的であるため、レコメンデーションは引き続き有効です。

数学用語では、各ユーザーと項目のペア (u,i) のスコアは、次の式に従って計算されます。ここで、 expは指数関数、$w_{\bar{u}}$ と $w_{i/j}$ はユーザーと項目の埋め込みをそれぞれ表し、ギリシャ語の文字の sigma (Σ) はスコアを持つすべての項目の合計を表します。

$$\text{score}(u,i) = \frac{\exp(\bar{w}_u^\top w_i)}{\sum_j \exp(\bar{w}_u^\top w_j)}$$

レコメンデーションの理由 (User-Personalization-v2)

User-Personalization-v2 を使用する場合、各推奨項目に、その項目がレコメンデーションに含まれていた理由のリストを含めることができます。考えられる理由は次のとおりです。

- 昇格済みアイテム – レコメンデーションリクエストに適用した昇格の一部としてアイテムが含まれていたことを示します。
- 探索 — 項目が探索に含まれたことを示します。探索では、レコメンデーションには、インタラクションデータやユーザーへの関連性が少ないアイテムが含まれます。探索の詳細については、「探索」を参照してください。
- 人気アイテム – アイテムがプレースホルダー人気アイテムとして含まれたことを示します。フィルターを使用する場合、フィルターが削除するレコメンデーションの数に応じて、Amazon Personalize はレコメンデーションリクエストnumResultsの を満たすプレースホルダー項目を追加する場合があります。これらの項目は、インタラクションデータに基づいて、フィルター条件を満たす人気のある項目です。ユーザーには関連性スコアがありません。

アイテムレコメンデーションの取得 (コンソール)

Amazon Personalize コンソールでレコメンデーションを取得するには、レコメンダー (ドメインデータセットグループ) またはカスタムキャンペーンの詳細のページでリクエスト情報を提供します。

推奨事項を取得するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. 使用しているキャンペーンやレコメンダーを含むデータセットグループを選択します。
3. ナビゲーションペインで、[キャンペーン] または [レコメンダー] を選択します。
4. ターゲットとなるキャンペーンまたはレコメンダーを選択します。
5. [キャンペーンの結果をテスト]で、使用したレシピに基づいたレコメンデーションのリクエストの詳細を入力します。レコメンダーの場合は、[テストレコメンダー] を選択し、ユースケースに基づいてレコメンデーションリクエストの詳細を入力します。

   ユーザーがログインする前にユーザーのイベントを記録した場合 (匿名ユーザー)、`userId` の代わりにそれらのイベントから `sessionId` を指定することにより、このユーザー向けのレコメンデーションを取得できます。匿名ユーザーのイベントの記録の詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。
6. オプションでフィルターを選択します。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。
7. コンテキストメタデータを使用する場合は、コンテキストごとにデータを提供してください。コンテキストごとに、「キー」にメタデータフィールドを入力します。値 にはコンテキストデータを入力します。詳細については、「コンテキストメタデータを使用したレコメンデーションの関連性の向上」を参照してください。
8. キャンペーンまたはレコメンダーのレコメンデーションでメタデータを有効にした場合、[Items データセット列]、レコメンデーション結果に含めるメタデータ列を選択します。キャンペーンのメタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。レコメンダーのメタデータの有効化については、「レコメンデーションのメタデータを有効にする」を参照してください。
9. 一部の商品をプロモーションする場合は、オプションで [プロモーション] フィールドに入力します。詳細については、レコメンデーション内のアイテムのプロモーションを参照してください。

10. [レコメンデーションの取得] を選択します。ユーザーの上位 25 個の推奨アイテムを含むテーブルが表示されます。User-Personalization-v2 を使用する場合、各推奨項目には、その項目がレコメンデーションに含まれた理由のリストが含まれます。詳細については、「レコメンデーションの理由 (User-Personalization-v2)」を参照してください。

アイテムレコメンデーションの取得 (AWS CLI)

以下のコードサンプルは、 AWS CLIを使用してアイテムレコメンデーションを取得するさまざまな方法を示しています。

トピック

- アイテムレコメンデーションの取得
- レコメンデーションへのアイテムメタデータの組み込み

アイテムレコメンデーションの取得

次のコードを使用してキャンペーンからレコメンデーションを取得します。レコメンダーからレコメンデーションを取得するには、`campaign-arn` パラメータを `recommender-arn` に置き換えます。

レコメンデーションを取得するユーザーの ID、およびキャンペーンまたはレコメンダーの Amazon リソースネーム (ARN) を指定します。ユーザーに推奨される上位 10 個のアイテムのリストが表示されます。User-Personalization-v2 を使用する場合、各推奨項目には、その項目がレコメンデーションに含まれた理由のリストが含まれます。詳細については、「レコメンデーションの理由 (User-Personalization-v2)」を参照してください。

推奨アイテムの数を変更するには、`numResults` の値を変更します。デフォルトのアイテム数は 25 です。アイテムの最大数は 500 です。RELATED_ITEMS レシピを使用して、キャンペーンをサポートするソリューションバージョンをトレーニングした場合は、`user-id` パラメータを `item-id` に置き換えて、アイテム ID を指定します。

ユーザーがログインする前にユーザーのイベントを記録した場合 (匿名ユーザー)、`userId` の代わりにそれらのイベントから `sessionId` を指定することにより、このユーザー向けのレコメンデーションを取得できます。匿名ユーザーのイベントの記録の詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。

```
aws personalize-runtime get-recommendations \
--campaign-arn campaign arn \
--user-id User ID \
```

```
--num-results 10
```

レコメンデーションへのアイテムメタデータの組み込み

キャンペーンまたはレコメンダーのレコメンデーションでメタデータを有効にした場合、レスポンスに含めるアイテムデータセットのメタデータ列を指定できます。キャンペーンのメタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。レコメンダーのメタデータの有効化については、「レコメンデーションのメタデータを有効にする」を参照してください。

次のコード例は、レコメンデーションのリクエストの一部としてメタデータ列を指定する方法を示しています。

```
aws personalize-runtime get-recommendations \
--campaign-arn campaign arn \
--user-id User ID \
--num-results 10 \
--metadata-columns "{\"ITEMS\": [\"\"columnNameA\"\",\"\"columnNameB\"\"]}"
```

アイテムレコメンデーションの取得 (AWS SDKs)

次のコードサンプルは、 AWS SDKs でアイテムレコメンデーションを取得する方法のさまざまなバリエーションを示しています。

トピック

- アイテムレコメンデーションの取得
- レコメンデーションへのアイテムメタデータの組み込み

アイテムレコメンデーションの取得

次のコードは、キャンペーンからユーザー向けの Amazon Personalizeのレコメンデーションを取得する方法を示しています。レコメンダーからレコメンデーションを取得するには、`campaignArn` パラメータを `recommenderArn` に置き換えます。

レコメンデーションを取得するユーザーの ID、およびキャンペーンまたはレコメンダーの Amazon リソースネーム (ARN) を指定します。ユーザーに推奨される上位 10 個のアイテムのリストが表示されます。User-Personalization-v2 を使用する場合、各推奨項目には、その項目がレコメンデーションに含まれた理由のリストが含まれます。詳細については、「レコメンデーションの理由 (User-Personalization-v2)」を参照してください。

推奨アイテムの数を変更するには、numResults の値を変更します。デフォルトのアイテム数は 25 です。アイテムの最大数は 500 です。RELATED_ITEMS レシピを使用して、キャンペーンをサポートするソリューションバージョンをトレーニングした場合は、userId パラメータを itemId に置き換えて、アイテム ID を指定します。

キャンペーンまたはレコメンダーのレコメンデーションでメタデータを有効にした場合、レスポンスに含めるアイテムデータセットのメタデータ列を指定できます。コードサンプルについては、「レコメンデーションへのアイテムメタデータの組み込み」を参照してください。メタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。

ユーザーがログインする前にユーザーのイベントを記録した場合 (匿名ユーザー)、userId の代わりにそれらのイベントから sessionId を指定することにより、このユーザー向けのレコメンデーションを取得できます。匿名ユーザーのイベントの記録の詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID',
    numResults = 10
)

print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
```

SDK for Java 2.x

```
    public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
 String campaignArn, String userId) {

        try {
            GetRecommendationsRequest recommendationsRequest =
 GetRecommendationsRequest.builder()
                    .campaignArn(campaignArn)
                    .numResults(20)
```

```
                    .userId(userId)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
 personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();
            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: 'CAMPAIGN_ARN', /* required */
  userId: 'USER_ID',      /* required */
  numResults: 15    /* optional */
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
```

```
    console.log("Error", err);
  }
};
run();
```

レコメンデーションへのアイテムメタデータの組み込み

キャンペーンまたはレコメンダーのレコメンデーションでメタデータを有効にした場合、レスポンスに含めるアイテムデータセットのメタデータ列を指定できます。キャンペーンのメタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。レコメンダーのメタデータの有効化については、「レコメンデーションのメタデータを有効にする」を参照してください。

次のコード例は、レコメンデーションのリクエストの一部としてメタデータ列を指定する方法を示しています。

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_recommendations(
  campaignArn = 'Campaign ARN',
  userId = 'User ID',
  numResults = 10
  metadataColumns = {
    "ITEMS": ['columnNameA','columnNameB']
  }
)

print("Recommended items")
for item in response['itemList']:
  print(item['itemId'])
  print(item['metadata'])
```

レコメンデーション内のアイテムのプロモーション

すべてのドメインユースケースと一部のカスタムレシピでは、リアルタイムのレコメンデーションを取得するときにプロモーションを指定できます。

プロモーションでは、設定可能な推奨アイテムのサブセットに適用される追加のビジネスルールを定義します。例えば、ストリーミングアプリがあり、自身の番組や映画を宣伝したいだけでなく、関連するタイトルもレコメンドしたい場合があるかもしれません。プロモーションを利用して、推奨アイテムの一定割合を社内のカテゴリからとするように指定できます。残りの推奨アイテムは、レシピやリクエストフィルターに基づいて、引き続き関連性あるレコメンデーションになります。

プロモーションを適用するには、レコメンデーションリクエストで以下を指定します。

- プロモーションフィルターを適用する推奨アイテムの割合。
- プロモーション条件を指定するフィルター。詳細については、「プロモーションフィルター」を参照してください。

レコメンデーションレスポンスでは、プロモートされたアイテムは他の推奨アイテムと比べてランダムに配置されますが、他のプロモーション対象アイテムとの相対的な順序で並べられます。レシピに応じて、プロモーションに含まれていない推奨アイテムは、ユーザーとの関連性、人気、類似度でソートされます。プロモーション条件を満たす商品が足りない場合、結果にはできるだけ多くのプロモーション対象アイテムが含まれます。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用して、レコメンデーションにプロモーションを適用できます。

トピック

- プロモーションをサポートするユースケースとレシピ
- プロモーションフィルター
- 新しいアイテムのプロモーション
- アイテムのプロモーション (コンソール)
- アイテムのプロモーション (AWS CLI)
- アイテムのプロモーション (AWS SDKs)

プロモーションをサポートするユースケースとレシピ

すべてのユースケースがプロモーションをサポートします。以下のカスタムレシピがプロモーションをサポートします。

- USER_PERSONALIZATION レシピ
- RELATED_ITEMS レシピ

- POPULAR_ITEMS レシピ

プロモーションフィルター

レコメンデーションリクエストにプロモーションを適用するときは、プロモーション条件を指定するフィルターを選択します。既存のフィルターを使用するか、新しいフィルターを作成できます。Amazon Personalize の他のフィルターと同様に、プロモーション用のフィルターを作成および管理します。フィルターの作成および管理の詳細については、「結果のフィルタ処理」を参照してください。

プロモーションフィルターとプロモーション外で選択するフィルター (リクエストフィルター) との唯一の違いは、Amazon Personalize がそれらをどのように適用するかです。プロモーションフィルターはプロモーション対象アイテムのみに適用され、リクエストフィルターはその他の推奨アイテムにのみ適用されます。リクエストフィルターとプロモーションフィルターを指定し、両方のフィルターをプロモーション対象アイテムに適用する場合は、プロモーションフィルターの式に両方の式を含める必要があります。2 つの式を組み合わせる方法は、使用するデータセットによって異なります。フィルター式、ルール、およびフィルター式の作成方法の詳細については、「フィルタ式」を参照してください。

フィルター式の例

次の式は、「社内」カテゴリのアイテムのみを含みます。この表現は、レコメンデーションで自身のコンテンツを宣伝したい場合に使用できます。

```
INCLUDE ItemID WHERE Items.OWNER IN ("in-house")
```

次の式には、指定したタイムスタンプよりも最近作成された項目のみが含まれます。この式を使用して、レコメンデーションで新しいアイテムをプロモーションできます。

```
INCLUDE ItemID WHERE Items.CREATION_TIMESTAMP > $DATE
```

次の式は、プロモーション対象アイテムにリクエストフィルターを適用する方法を示しています。プロモーション対象アイテムとして入手可能な衣料品のみが含まれます。このシナリオでは、`Items.AVAILABLE IN ("True")` をリクエストフィルターの式にも使用して、すべてのレコメンデーションが入手可能なアイテムに関するものになります。

```
INCLUDE ItemID WHERE Items.CATEGORY IN ("clothing") AND Items.AVAILABLE IN ("True")
```

フィルターの例の詳細なリストついての詳細は、「フィルター式の例」を参照してください。

新しいアイテムのプロモーション

を使用する場合User-Personalization-v2 レシピ、Amazon Personalize は最も関連性の高いアイテムをユーザーに推奨し、インタラクションデータを含む既存のアイテムをより頻繁に推奨します。レコメンデーションに新しいアイテムが含まれるようにするには、作成タイムスタンプに基づいてアイテムを含むレコメンデーションリクエストにプロモーションを適用できます。

プロモーションをまだ使用していない場合、フィルター式は特定の日付以降に作成されたアイテムをプロモーションできます。

```
INCLUDE ItemID WHERE Items.CREATION_TIMESTAMP > $DATE
```

既にプロモーションを使用している場合は、プロモーションと新しいアイテム条件ステートメントの両方を連鎖する式を作成します。

```
INCLUDE ItemID WHERE Items.CATEGORY IN ("clothing") OR Items.CREATION_TIMESTAMP > $DATE
```

アイテムのプロモーション (コンソール)

Amazon Personalize コンソールでレコメンデーション内の特定のアイテムをプロモーションするには、フィルターを作成して、レコメンデーションリクエストにプロモーションの詳細を入力します。その他フィールドについては、「アイテムレコメンデーションの取得 (コンソール)」を参照してください。

レコメンデーション内のアイテムのプロモーションを行うには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. 使用しているキャンペーンまたはレコメンダーを含むデータセットグループを選択します。
3. プロモーション条件を指定するフィルターをまだ作成していない場合は、作成します。プロモーション用のフィルターは、リクエストフィルターを作成するのと同じ方法で作成します。の作成および管理の詳細については、「結果のフィルタ処理」を参照してください。
4. ナビゲーションペインで、[レコメンデーション] または [キャンペーン] を選択します。
5. ターゲットとなるキャンペーンまたはレコメンダーを選択します。
6. [キャンペーンの結果をテスト] で、使用したレシピに基づいたレコメンデーションのリクエストの詳細を入力します。レコメンダーの場合は、[テストレコメンダー] を選択し、レコメンデーションリクエストの詳細を入力します。

7. オプションで、リクエストのフィルターを選択します。このフィルターはプロモーション対象外のアイテムにのみ適用されます。フィルターの作成および管理の詳細については、「結果のフィルタ処理」を参照してください。
8. コンテキストメタデータを使用する場合は、コンテキストごとにデータを提供してください。コンテキストごとに、[キー] にメタデータフィールドを入力します。[値]には、コンテキストデータを入力します。詳細については、「コンテキストメタデータを使用したレコメンデーションの関連性の向上」を参照してください。
9. [プロモーション] には以下を指定します。

   - プロモーション商品の割合: プロモーションを適用する推奨アイテムの割合を入力します。
   - フィルター: プロモーション条件を指定するフィルターを選択します。このフィルターは、ステップ 7 で指定したリクエストフィルターではなく、プロモーション対象のアイテムに適用されます。
   - プレースホルダーパラメータとともにフィルターを使用している場合は、各パラメータに値を入力して、フィルター基準を設定します。1 つのパラメータに複数の値を使用するには、各値をコンマで区切ります。
10. [レコメンデーションの取得] を選択します。ユーザーの上位 25 個の推奨アイテムを含むテーブルが表示されます。プロモート対象アイテム列には、そのアイテムがプロモーションのために含まれていたかどうかが示されます。プロモーション対象アイテムは、他の推奨アイテムとの相対位置はランダムですが、他のプロモーション対象アイテムとの相対的な順序で並んでいます。ユースケースやレシピに応じて、プロモーションに含まれていない推奨アイテムは、ユーザーとの関連性、人気、類似度でソートされます。プロモーション条件を満たすアイテムが十分でない場合、結果にはできるだけ多くのプロモーション対象アイテムが含まれます。

アイテムのプロモーション (AWS CLI)

次のコードは、 AWS CLI およびカスタムキャンペーンを使用してレコメンデーション内のアイテムをプロモーションする方法を示しています。レコメンダーを使って商品をプロモーションするには、`campaign-arn` パラメーターを `recommender-arn` に置き換え、レコメンダーの Amazon リソースネーム (ARN) を指定します。プロモーションフィールドには、以下を指定します。

- 名前: プロモーションに名前を付けます。レコメンデーションレスポンスでは、その名前を使用してプロモートされたアイテムを識別します。
- percent-promoted-items: プロモーションを適用する推奨アイテムの割合。この例では、50% の商品がプロモーション対象商品になります。

- filterArn: プロモーション条件を定義するフィルターの Amazon リソースネーム (ARN) を指定します。詳細については、「プロモーションフィルター」を参照してください。
- パラメータの名前と値:フィルター式にパラメータがある場合は、パラメータ名 (大文字と小文字が区別されます) と値を指定します。例えば、フィルター式に $GENRE パラメータがある場合は、キーとして「GENRE」を指定し、値として 1 つまたは複数のジャンル (コメディ など) を指定します。複数の値はコンマで区切ります。を使用する場合 AWS CLI、値ごとに /文字を使用して引用符と /文字の両方を エスケープする必要があります。次のコード例は、値をフォーマットする方法を示しています。

このコードは、リクエストフィルターとプロモーションフィルターの両方を使用する方法を示しています。プロモーションフィルターはプロモーション対象アイテムのみに適用され、リクエストフィルターは残りの推奨アイテムにのみ適用されます。詳細については、「プロモーションフィルター」を参照してください。

追加フィールドについては、「アイテムレコメンデーションの取得 (AWS SDKs)」および「コンテキストメタデータを使用した、パーソナライズされたランキングの取得」を参照してください。

```
aws personalize-runtime get-recommendations \
--campaign-arn CampaignArn \
--user-id 1 \
--num-results 10  \
--filter-arn RequestFilterArn \
--filter-values '{
    "RequestFilterParameterName": "\"value\"",
    "RequestFilterParameterName": "\"value1\",\"value2\",\"value3\""
  }' \
--promotions "[{
  \"name\": \"promotionName\",
  \"percentPromotedItems\": 50,
  \"filterArn\": \"PromotionFilterARN\",
  \"filterValues\": {\"PromotionParameterName\":\"\\\"value1, value2\\\"\"}
}]"
```

推奨アイテムのリストが表示されます。プロモーション対象アイテムは、他の推奨アイテムと比べてランダムに配置されますが、他のプロモーション対象アイテムとの相対的な順序で並べられます。レシピに応じて、プロモーションに含まれていない推奨アイテムは、ユーザーとの関連性、人気、類似度でソートされます。プロモーション条件を満たす商品が足りない場合、結果にはできるだけ多くのプロモーション対象アイテムが含まれます。

```
{
  "itemList": [
      {
          "itemId1": "123",
          "score": .0117211,
          "promotionName": "promotionName"
      },
      {
         "itemId2": "456",
         "score": .0077976
      },
      {
         "itemId3": "789",
         "score": .0067171
      },
      .....
 ]
```

アイテムのプロモーション (AWS SDKs)

次のコードは、SDK for Python (Boto3) と SDK for Java 2.x、およびカスタムキャンペーンを使用してレコメンデーション内のアイテムをプロモーションする方法を示しています。レコメンダーを使用して商品をプロモーションするには、`campaignArn` パラメータを `recommenderArn` に置き換え、レコメンダーの Amazon リソースネーム (ARN) を指定します。プロモーションフィールドには、以下を指定します。

- 名前: プロモーションの名前を指定します。レコメンデーションレスポンスには、プロモーション対象アイテムを特定するための名前が含まれます。
- percentPromotedItems: プロモーションを適用する推奨アイテムの割合。
- promotionFilterARN: プロモーション条件を定義するフィルターの Amazon リソースネーム (ARN)。詳細については、「プロモーションフィルター」を参照してください。
- 任意のパラメータの名前と値:フィルター式にパラメータがある場合は、フィルター式の各パラメータについて、パラメータ名 (大文字と小文字が区別されます) と値を指定します。例えば、フィルター式に `$GENRE` パラメータがある場合は、キーとして `"GENRE"` を指定し、値として 1 つまたは複数のジャンル (`"\"Comedy"\"` など) を指定します。複数の値はコンマで区切ります。例えば `"\"comedy\",\"drama\",\"horror"\"` です。

次のコードは、リクエストフィルターとプロモーションフィルターの両方を使用する方法を示しています。プロモーションフィルターはプロモーション対象アイテムのみに適用され、リクエストフィル

ターは残りの推奨アイテムにのみ適用されます。詳細については、「プロモーションフィルター」を参照してください。

追加フィールドについては、「アイテムレコメンデーションの取得 (AWS SDKs)」および「コンテキストメタデータを使用した、パーソナライズされたランキングの取得」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_recommendations(
  campaignArn = "CampaignARN",
  userId = '1',
  numResults = 10,
  filterArn = 'RequestFilterARN',
  filterValues = {
      "RequestFilterParameterName": "\"value1\"",
      "RequestFilterParameterName": "\"value1\",\"value2\",\"value3\""
      ....
  },
  promotions = [{
    "name" : "promotionName",
    "percentPromotedItems" : 50,
    "filterArn": "promotionFilterARN",
    "filterValues": {
      "PromotionParameterName": "\"Value1\",\"Value2\""
      ...
    }
  }]
)

print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
    if ("promotionName" in item):
        print(item['promotionName'])
```

SDK for Java 2.x

```
public static void getRecommendationsWithPromotedItems(PersonalizeRuntimeClient
 personalizeRuntimeClient,
```

```
                                        String campaignArn,
                                        String userId,
                                        String requestFilterArn,
                                        String requestParameterName,
                                        String requestParameterValue1,
                                        String requestParameterValue2,
                                        String promotionName,
                                        int percentPromotedItems,
                                        String promotionFilterArn,
                                        String promotionParameterName,
                                        String promotionParameterValue1,
                                        String promotionParameterValue2) {

  try {

      Map<String, String> promotionFilterValues = new HashMap<>();

      promotionFilterValues.put(promotionParameterName, String.format("\"%1$s\",
\"%2$s\"",
              promotionParameterValue1, promotionParameterValue2));

      Promotion newPromotion = Promotion.builder()
              .name(promotionName)
              .percentPromotedItems(percentPromotedItems)
              .filterArn(promotionFilterArn)
              .filterValues(promotionFilterValues)
              .build();

      List<Promotion> promotionList = new List<>();

      promotionsList.add(newPromotion);

      Map<String, String> requestfilterValues = new HashMap<>();

      requestfilterValues.put(requestParameterName, String.format("\"%1$s\",\"%2$s
\"",
              requestParameterValue1, requestParameterValue2));

      GetRecommendationsRequest recommendationsRequest =
 GetRecommendationsRequest.builder()
              .campaignArn(campaignArn)
              .numResults(20)
              .userId(userId)
              .filterArn(requestFilterArn)
```

```
                .fitlerValues(requestFilterValues)
                .promotions(promotionList)
                .build();

      GetRecommendationsResponse recommendationsResponse =
 personalizeRuntimeClient.getRecommendations(recommendationsRequest);
      List<PredictedItem> items = recommendationsResponse.itemList();

      for (PredictedItem item: items) {
          System.out.println("Item Id is : "+item.itemId());
          System.out.println("Item score is : "+item.score());
          System.out.println("Promotion name is : "+item.promotionName());
      }
  } catch (PersonalizeRuntimeException e) {
      System.err.println(e.awsErrorDetails().errorMessage());
      System.exit(1);
  }
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { GetRecommendationsCommand, PersonalizeRuntimeClient } from
  "@aws-sdk/client-personalize-runtime";

// create personalizeRuntimeClient.
const personalizeRuntimeClient = new PersonalizeRuntimeClient({
  region: "REGION",
});

// set recommendation request param
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN",  /* required */
  userId: "USER_ID", /* required */
  numResults: 25, /* optional */
  filterArn: "FILTER_ARN", /* provide if you are applying a custom filter */
  filterValues: {
    "PARAM_NAME": "\"PARAM_VALUE\"" /* provide if your filter has a placeholder
 parameter */
  },
  promotions: [
    {
```

```
      name: "PROMOTION_NAME", /* specify the name of the promotion. The
 recommendation response includes the name to identify promoted items. */
      percentPromotedItems: 50, /* the percentage of recommended items to apply the
 promotion to. */
      filterArn:
        "PROMOTION_FILTER_ARN", /* the Amazon Resource Name (ARN) of the filter that
 defines the promotion criteria. */
      filterValues: {
        "PARAM_NAME": "\"PARAM_VALUE\"" /* provide if your promotion filter has a
 placeholder parameter */
      },
    },
  ],
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", "\nItems are: ");
    response.itemList.forEach(element => console.log(element.itemId))
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

推奨アイテムのリストが表示されます。プロモーション対象アイテムは、他の推奨アイテムと比べてランダムに配置されますが、他のプロモーション対象アイテムとの相対的な順序で並べられます。レシピに応じて、プロモーションに含まれていない推奨アイテムは、ユーザーとの関連性、人気、類似度でソートされます。プロモーション条件を満たす商品が足りない場合、結果にはできるだけ多くのプロモーション対象アイテムが含まれます。

```
{
  "itemList": [
      {
          "itemId1": "123",
          "score": .0117211,
          "promotionName": "promotionName"
```

```
    },
    {
       "itemId2": "456",
       "score": .0077976
    },
    {
       "itemId3": "789",
       "score": .0067171
    },
    .....
]
```

アクションレコメンデーションの取得

PERSONALIZED_ACTIONS レシピを使用すると、キャンペーンからアクションレコメンデーションをリアルタイムで取得できます。Amazon Personalize コンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して、アクションレコメンデーションを取得できます。

トピック

- アクションレコメンデーションのスコアの仕組み
- アクションレコメンデーションの取得 (コンソール)
- アクションレコメンデーションの取得 (AWS CLI)
- アクションレコメンデーションの取得 (AWS SDK)

アクションレコメンデーションのスコアの仕組み

Next-Best-Action レシピでは、Amazon Personalize はユーザーがアクションを操作する可能性に基づいてアクションのスコアを生成します。スコアは 0 ~ 1.0 です。1.0 に近いほど、ユーザーがアクションを操作する可能性が高くなります。

アクションインタラクションデータをインポートしていない場合、推奨されたすべてのアクションのスコアは 0.0 になります。Amazon Personalize が探索の一環としてアクションを推奨した場合、そのアイテムのスコアは 0.0 になります。Amazon Personalize は探索機能を使用して、アクションインタラクションデータなしでアクションを推奨します。探索の詳細については、「探査」を参照してください。

アクションレコメンデーションの取得 (コンソール)

Amazon Personalize コンソールでアクションレコメンデーションを取得するには、カスタムキャンペーンの詳細ページでリクエスト情報を指定します。

アクションレコメンデーションを取得するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. 使用しているキャンペーンを含むデータセットグループを選択します。
3. ナビゲーションペインの [カスタムリソース] で、[キャンペーン] を選択します。
4. ターゲットとなるキャンペーンを選択します。
5. [キャンペーン結果をテスト] で、レコメンデーションリクエストの詳細を入力します。

   ユーザーがログインする前にユーザーのイベントを記録した場合 (匿名ユーザー)、userId の代わりにそれらのイベントから sessionId を指定することにより、このユーザー向けのレコメンデーションを取得できます。匿名ユーザーのイベントの記録の詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。
6. オプションでフィルターを選択します。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。
7. [レコメンデーションの取得] を選択します。ユーザーの上位 5 個の推奨アクションを含むテーブルが表示されます。

アクションレコメンデーションの取得 (AWS CLI)

次のコードを使用して、キャンペーンからアクションレコメンデーションを取得します。レコメンデーションを取得するユーザーの ID と、キャンペーンの Amazon リソースネーム (ARN) を指定します。

推奨アクションの数を変更するには、numResults の値を変更します。デフォルトは、5 つのアクションです。最大値は 100 アクションです。

アクションレコメンデーションをカスタム条件でフィルタリングするには、フィルターを作成して、get-action-recommendations 操作に適用できます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

ユーザーがログインする前にユーザーのイベントを記録した場合 (匿名ユーザー)、userId の代わりにそれらのイベントから sessionId を指定することにより、このユーザー向けのレコメンデーションを取得できます。匿名ユーザーのイベントの記録の詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。

```
aws personalize-runtime get-action-recommendations \
```

```
--campaign-arn campaign arn \
--user-id User ID \
--num-results 10
```

アクションレコメンデーションの取得 (AWS SDK)

次のコードは、キャンペーンからユーザー向けの Amazon Personalizeのレコメンデーションを取得する方法を示しています。レコメンデーションを取得するユーザーの ID と、キャンペーンの Amazon リソースネーム (ARN) を指定します。

推奨アクションの数を変更するには、numResults の値を変更します。デフォルトは、5 つのアクションです。最大値は 100 アクションです。

アクションレコメンデーションをカスタム条件でフィルタリングするには、フィルターを作成して GetActionRecommendations API リクエストに適用できます。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

ユーザーがログインする前にユーザーのイベントを記録した場合 (匿名ユーザー)、userId の代わりにそれらのイベントから sessionId を指定することにより、このユーザー向けのレコメンデーションを取得できます。匿名ユーザーのイベントの記録の詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_action_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID',
    numResults = 10
)

print("Recommended actions")
for item in response['actionList']:
    print (item['actionId'])
```

パーソナライズされたランキングの取得 (カスタムリソース)

パーソナライズされたランキングは、特定のユーザー向けに再ランク付けされた推奨事項のリストです。パーソナライズされたランキングを取得するには、GetPersonalizedRanking API 操作を呼び出すか、コンソールのキャンペーンからレコメンデーションを取得します。

> **Note**
>
> キャンペーンにデプロイするソリューションが PERSONALIZED_RANKING タイプのレシピを使用して作成済みであることが必要です。詳細については、「レシピの選択」を参照してください。

トピック

- パーソナライズされたランキングスコアリングの仕組み
- パーソナライズされたランキングの取得 (コンソール)
- パーソナライズされたランキングの取得 (AWS CLI)
- パーソナライズされたランキングの取得 (AWS SDKs)
- Personalized-Ranking サンプルノートブック

パーソナライズされたランキングスコアリングの仕組み

User-Personalization-v2 および User-Personalization レシピで作成されたソリューションの GetRecommendationsオペレーションによって返されるスコアと同様に、GetPersonalizedRankingスコアの合計は 1 になりますが、入力項目のみがスコアを受け取り、レコメンデーションスコアが高くなる傾向があります。

数学的には、 のスコアリング関数 GetPersonalizedRanking は、入力項目のみを考慮することを除いてGetRecommendations、 と同じです。これは、スコアを分割する他の選択肢が少なくなるため、1 に近いスコアの可能性が高くなることを意味します。

$$\text{score}(u,i) = \frac{\exp(\bar{w}_u^\top w_i)}{\sum_{j \in \text{input}} \exp(\bar{w}_u^\top w_j)}$$

パーソナライズされたランキングの取得 (コンソール)

Amazon Personalize コンソールからユーザー向けにパーソナライズされたランキングを取得するには、使用しているキャンペーンを選択してからユーザー ID を入力し、ユーザーのためにランク付けするアイテムのリストを指定します。また、オプションで、フィルターを選択し、任意のコンテキストデータを指定します。

ユーザー向けのパーソナライズされたランキングを取得するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. 使用しているキャンペーンを含むデータセットグループを選択します。
3. ナビゲーションペインで、[キャンペーン] を選択します。
4. [Campaigns] (キャンペーン) ページで、ターゲットキャンペーンを選択します。
5. [Test campaign results] (キャンペーン結果をテスト) で、レコメンデーションを取得するユーザーの [User ID] (ユーザー ID) を入力します。
6. [Item IDs] (アイテム ID) で、ユーザーのためにランク付けするアイテムのリストを入力します。
7. オプションでフィルターを選択します。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。
8. キャンペーンのレコメンデーションでメタデータを有効にした場合、[Items データセット列] で、レコメンデーション結果に含めるメタデータ列を選択します。メタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。
9. キャンペーンでコンテキストメタデータを使用する場合 (要件については「コンテキストメタデータを使用したレコメンデーションの関連性の向上」を参照)、オプションでコンテキストデータを入力します。

   各コンテキストについて、[Key] (キー) にメタデータフィールドを入力し、[Value] (値) にコンテキストデータを入力します。
10. [Get personalized item rankings] (パーソナライズされたアイテムのランキングを取得) を選択します。ユーザー向けの予測される関心の順にランク付けされたアイテムを含むテーブルが表示されます。

パーソナライズされたランキングの取得 (AWS CLI)

次のコード例は、 AWS CLIでパーソナライズされたランキングを取得する方法のさまざまなバリエーションを示しています。

トピック

- パーソナライズされたランキングの取得
- パーソナライズされたランキングへのアイテムメタデータの組み込み

パーソナライズされたランキングの取得

次の get-personalized-ranking コマンドを使用して、 AWS CLIでパーソナライズされたランキングを取得します。キャンペーンの Amazon リソースネーム (ARN)、ユーザーのユーザー ID を指定し、ユーザー向けにランク付けするアイテムのアイテム ID のリストを提供します (それぞれスペースで区切ります)。ランク付けするアイテムは、ソリューションバージョンのトレーニングに使用したデータに含まれている必要があります。ランク付けされたレコメンデーションのリストが表示されます。Amazon Personalize は、ユーザーにとって最も関心のあるリストの最初のアイテムを考慮します。

```
aws personalize-runtime get-personalized-ranking \
--campaign-arn Campaign ARN \
--user-id 12 \
--input-list 3 4 10 8 12 7
```

パーソナライズされたランキングへのアイテムメタデータの組み込み

キャンペーンのレコメンデーションでメタデータを有効にした場合、レスポンスに含めるアイテムデータセットのメタデータ列を指定できます。メタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。

次のコードサンプルは、パーソナライズしたランキングのリクエストの一部としてメタデータ列を指定する方法を示しています。

```
aws personalize-runtime get-personalized-ranking \
--campaign-arn Campaign ARN \
--user-id 12 \
--input-list 3 4 10 8 12 7
--metadata-columns "{\"ITEMS\": [\"\"columnNameA\"\",\"\"columnNameB\"\"]}"
```

パーソナライズされたランキングの取得 (AWS SDKs)

次のコードサンプルは、 AWS SDKs でパーソナライズされたランキングを取得する方法のさまざまなバリエーションを示しています。

トピック

- パーソナライズされたランキングの取得
- パーソナライズされたランキングへのアイテムメタデータの組み込み
- コンテキストメタデータを使用した、パーソナライズされたランキングの取得

パーソナライズされたランキングの取得

次のコードは、パーソナライズされたランキングを取得する方法を示しています。ユーザーの ID と、ユーザーのためにランク付けするアイテム ID のリストを指定します。アイテム ID は、ソリューションバージョンのトレーニングに使用したデータに含まれている必要があります。ランク付けられたレコメンデーションが一覧表示されます。Amazon Personalize は、ユーザーにとって最も関心のあるリストの最初のアイテムを考慮します。

SDK for Python (Boto3)

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_personalized_ranking(
    campaignArn = "Campaign arn",
    userId = "UserID",
    inputList = ['ItemID1','ItemID2']
)

print("Personalized Ranking")
for item in response['personalizedRanking']:
    print (item['itemId'])
```

SDK for Java 2.x

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
 personalizeRuntimeClient,
                                                 String campaignArn,
                                                 String userId,
                                                 ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
 GetPersonalizedRankingRequest.builder()
                .campaignArn(campaignArn)
                .userId(userId)
                .inputList(items)
                .build();

        GetPersonalizedRankingResponse recommendationsResponse =

 personalizeRuntimeClient.getPersonalizedRanking(rankingRecommendationsRequest);
```

```
        List<PredictedItem> rankedItems =
 recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-------------------------------------------");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { GetPersonalizedRankingCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN", /* required */
  userId: 'USER_ID',      /* required */
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"]
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetPersonalizedRankingCommand(getPersonalizedRankingParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
```

```
  }
};
run();
```

パーソナライズされたランキングへのアイテムメタデータの組み込み

キャンペーンのレコメンデーションでメタデータを有効にした場合、レスポンスに含めるアイテムデータセットのメタデータ列を指定できます。メタデータの有効化については、「レコメンデーションのアイテムメタデータ」を参照してください。

次のコードサンプルは、パーソナライズしたランキングのリクエストの一部としてメタデータ列を指定する方法を示しています。

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_personalized_ranking(
    campaignArn = "Campaign arn",
    userId = "UserID",
    inputList = ['ItemID1','ItemID2'],
    metadataColumns = {
      "ITEMS": ['columnNameA','columnNameB']
    }
)

print("Personalized Ranking")
for item in response['personalizedRanking']:
    print (item['itemId'])
    print (item['metadata'])
```

コンテキストメタデータを使用した、パーソナライズされたランキングの取得

コンテキストメタデータに基づいてパーソナライズされたランキングを取得するには、次のコードを使用します。context については、キーバリューペアごとに、メタデータフィールドをキーとして指定し、コンテキストデータを値として指定します。次のサンプルコードでは、キーは DEVICE で、値は mobile phone です。これらの値と Campaign ARN および User ID を、独自の値に置き換えます。また、inputList を、ソリューションのトレーニングに使用したデータにあるアイテム ID のリストに変更します。Amazon Personalize は、ユーザーにとって最も関心のあるリストの最初のアイテムを考慮します。

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_personalized_ranking(
    campaignArn = "Campaign ARN",
    userId = "User ID",
    inputList = ['ItemID1', 'ItemID2'],
    context = {
      'DEVICE': 'mobile phone'
    }
)

print("Personalized Ranking")
for item in response['personalizedRanking']:
  print(item['itemId'])
```

Personalized-Ranking サンプルノートブック

Personalized-Ranking レシピの使用方法を示すサンプル Jupyter ノートブックについては、「Personalize Ranking の例」を参照してください。

コンテキストメタデータを使用したレコメンデーションの関連性の向上

レコメンデーションの関連性を高めるには、アイテムレコメンデーションを取得したり、パーソナライズされたランキングを取得したりするときに、デバイスの種類や時刻など、ユーザーのコンテキストメタデータを含めます。

コンテキストメタデータを使用するには、アイテムインタラクションデータセットのスキーマにコンテキストデータ用のメタデータフィールドが必要です。例えば、DEVICE フィールド (「スキーマ」を参照)。

ドメインデータセットグループでは、以下のレコメンダーユースケースでコンテキストメタデータを使用できます。

- おすすめ (eコマースドメイン)
- 上位のおすすめ (VIDEO_ON_DEMAND ドメイン)

カスタムリソースについては、コンテキストメタデータを使用するレシピには次のものが含まれます。

- User-Personalization-v2 および User-Personalization
- Personalized-Ranking-v2 および Personalized-Ranking

コンテキスト情報の詳細については、 Machine AWS Machine Learning ブログ記事「コンテキスト情報 を活用して Amazon Personalize のレコメンデーションの関連性を高める」を参照してください。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用して、コンテキストメタデータを含むレコメンデーションを取得できます。

コンテキストメタデータを使用したレコメンデーションの取得 (AWS Python SDK)

レコメンデーションの関連性を高めるには、アイテムレコメンデーションを取得したり、パーソナライズされたランキングを取得したりするときに、デバイスの種類や時刻など、ユーザーのコンテキストメタデータを含めます。

次のコードを使用して、コンテキストメタデータに基づいてレコメンデーションを取得します。context については、キーバリューペアごとに、メタデータフィールドをキーとして指定し、コンテキストデータを値として指定します。次のサンプルコードでは、キーは DEVICE で、値は mobile phone です。これらの値と Campaign ARN および User ID を、独自の値に置き換えます。レコメンダーを作成した場合は、campaignArn を recommenderArn に置き換えます。ユーザーへの推奨アイテムのリストが表示されます。

```
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID',
    context = {
      'DEVICE': 'mobile phone'
    }
)

print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
```

バッチレコメンデーションとユーザーセグメント (カスタムリソース)

カスタムリソースを使用すると、バッチレコメンデーションまたはユーザーセグメントを非同期バッチフローで取得できます。例えば、メーリングリストに登録されているすべてのユーザー向けの製品のレコメンデーション、またはインベントリ全体における商品間の類似性を取得する場合があります。または、USER_SEGMENTATION レシピを使用して、インベントリ内の商品とユーザーのインタラクションに基づいて、データ駆動型広告用にユーザーセグメントを作成できます。

- バッチレコメンデーションを取得するには、バッチ推論ジョブを使用します。バッチ推論ジョブは、Amazon S3 バケットからバッチ入力データをインポートし、ソリューションバージョンを使用して商品のレコメンデーションを生成してから、そのレコメンデーションを Amazon S3 バケットにエクスポートするツールです。
- ユーザーセグメントを取得するには、バッチセグメントジョブを使用します。バッチセグメントジョブは、Amazon S3 バケットからバッチ入力データをインポートし、USER_SEGMENTATION レシピでトレーニングされたソリューションバージョンを使用して、ユーザーセグメントを生成してセグメントを Amazon S3 バケットにエクスポートするツールです。

トピック

バッチレコメンデーションの取得

カスタムリソースを使用すると、非同期のバッチフローでアイテムのレコメンデーションを取得できます。例えば、E メールリストのすべてのユーザー向けの製品レコメンデーションや、インベントリ全体のitem-to-item類似点を取得できます。

バッチレコメンデーションを取得するには、バッチ推論ジョブを使用します。バッチ推論ジョブは、Amazon S3 バケットからバッチ入力データをインポートし、ソリューションバージョンを使用してアイテムのレコメンデーションを生成してから、そのレコメンデーションを Amazon S3 バケットにエクスポートするツールです。レシピに応じて、入力データは、ユーザーもしくはアイテムのリスト、またはそれぞれがアイテムのコレクションを持つユーザーのリストです。

ソリューションが Similar Items レシピを使用し、Items データセットにテキストデータとアイテムタイトルデータがある場合、アイテムグループごとにテーマ付きのバッチレコメンデーションを生成

できます。詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

バッチレコメンデーションを生成する際、Amazon Personalize は最新のソリューションバージョン作成時に存在するすべてのバルクデータを考慮します。このデータは、フルまたは増分のインポートモードでインポートできます。新しい一括レコードがバッチレコメンデーションに影響を与えるようにするには、新しいソリューションバージョンを作成してから、バッチ推論ジョブを作成する必要があります。

Amazon Personalize は、次のようにバッチレコメンデーションを生成する際に、個々のインポートのデータを使用します。

- 既存のアイテムやユーザーとの新規インタラクション: USER_PERSONALIZATION または PERSONALIZED_RANKING レシピを使用する場合、Amazon Personalize はデータインポートから約 15 分以内に既存のアイテムやユーザーとの新規インタラクションデータを考慮します。イベントが確実に考慮されるように、インポート後 15 分以上待ってからバッチ推論ジョブを開始することをお勧めします。その他のレシピについては、バッチレコメンデーションに影響を与えるように、ストリーミングイベント用に新しいソリューションバージョンを作成する必要があります。
- 新規ユーザー: インタラクションデータを持たないユーザーの場合、レコメンデーションは最初は人気商品のみを対象としています。USER_PERSONALIZATION または PERSONALIZED_RANKING レシピを使用していて、ユーザーのイベントを記録すると、インポート後約 15 分以内に再トレーニングを行わずにレコメンデーションの関連性が高くなる可能性があります。イベントが確実に考慮されるように、インポート後 15 分以上待ってからバッチ推論ジョブを開始することをお勧めします。その他のレシピについては、インタラクションデータのないユーザーへのバッチレコメンデーションに影響を与えるために、ストリーミングイベント用に新しいソリューションバージョンを作成する必要があります。
- 新しいアイテム: User-Personalization-v2 と User-Personalization では、バッチ推論ジョブを作成し、ソリューション用に完全にトレーニングされた最新のソリューションバージョンを指定すると、Amazon Personalize はソリューションバージョンを自動的に更新して、探索を伴うレコメンデーションに新しいアイテムを含めます。最新のソリューションバージョンを指定しないと、更新は行われません。その他のレシピについては、レコメンデーションに含める新しいアイテム用に新しいソリューションバージョンを作成する必要があります。探索の詳細については、「探査」を参照してください。

トピック

- バッチワークフロー
- ガイドラインと要件

- バッチワークフローのスコアリング
- Content Generator のテーマ付きバッチレコメンデーション
- バッチレコメンデーション用の入力データを準備します。
- バッチ推論ジョブの作成
- Batch 推論ジョブの出力例

バッチワークフロー

バッチワークフローは次のとおりです。

1. 入力データを JSON 形式で準備して Amazon S3 バケットにアップロードします。入力データの形式は、使用するレシピによって異なります。バッチレコメンデーション用の入力データを準備します。 を参照してください。
2. フォルダまたは別の Amazon S3 バケットのいずれかで、出力データ用に個別の場所を作成します。
3. バッチ推論ジョブを作成します。バッチ推論ジョブの作成 を参照してください。
4. バッチ推論が完了したら、Amazon S3 の出力場所からアイテムのレコメンデーションを取得します。

ガイドラインと要件

バッチレコメンデーションを取得するためのガイドラインと要件は次のとおりです。

- Amazon Personalize の IAM サービスロールには、Amazon S3 バケットを読み取り、ファイルを追加するための許可が必要です。許可の付与については、「バッチワークフロー用のサービスロールのポリシー」を参照してください。バケットの許可の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「ユーザーポリシーの例」を参照してください。AWS Key Management Service（ AWS KMS) を暗号化に使用する場合は、Amazon Personalize と Amazon Personalize の IAM サービスロールに、 キーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。
- バッチ推論ジョブを作成する前に、カスタムソリューションとソリューションバージョンを作成する必要があります。ただし、Amazon Personalize のキャンペーンを作成する必要はありません。ドメインデータセットグループを作成した場合は、引き続きカスタムリソースを作成できます。
- テーマ付きバッチレコメンデーションを生成するには、Similar-Items レシピを使用する必要があります。また、テキストデータとアイテムタイトルデータを含むアイテムデータセットが必要で

す。テーマ別レコメンデーションの詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

- 入力データは、「ユーザーセグメントの入力データを準備しています。」で説明されている形式にする必要があります。
- Trending-Now レシピまたは Next-Best-Action レシピではバッチレコメンデーションを取得することはできません。
- プレースホルダーパラメータを含むフィルタを使用する場合は、`filterValues` オブジェクトの入力データにパラメータの値を含める必要があります。詳細については、「入力 JSON にフィルター値を指定します。」を参照してください。
- 出力データには入力データとは異なるの場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。
- バッチレコメンデーションは、リアルタイムレコメンデーションとまったく同じではない場合があります。これは、バッチ推論ジョブの完了に時間がかかり、ジョブの開始の 15 分前にのみ利用可能なデータを考慮するためです。

バッチワークフローのスコアリング

バッチレコメンデーションには、次のようなスコアが含まれます。

- Amazon Personalize は、ユーザーパーソナライゼーションレシピとパーソナライズランキングレシピを使用して、レコメンデーションスコアリングの仕組み (カスタムリソース) および パーソナライズされたランキングスコアリングの仕組み で説明されているようにバッチ推論ジョブの推奨スコアを計算します。バッチ推論ジョブの出力 JSON ファイルでスコアを表示できます。
- Similar-Items レシピでテーマ別のバッチレコメンデーションを取得する場合、Amazon Personalize は、各アイテムに対するテーマの関連性に基づいて関連アイテムの各セットをランク付けします。各アイテムには 0 ~ 1 のスコアが含まれます。スコアが高いほど、そのアイテムはテーマと密接に関連していることになります。テーマ付きレコメンデーションの詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

Content Generator のテーマ付きバッチレコメンデーション

> **Important**
>
> テーマ付きバッチレコメンデーションを取得すると、追加コストが発生します。詳細については、「Amazon Personalize の料金」を参照してください。

Similar-Items レシピを使用する場合、Amazon Personalize Content Generator はわかりやすいテーマをバッチレコメンデーションに追加できます。Content Generator は、Amazon Personalize が管理する生成系人工知能 (生成 AI) 機能です。

テーマ付きバッチレコメンデーションを受け取ると、Amazon Personalize Content Generator は類似アイテムのセットごとにわかりやすいテーマを追加します。テーマは、アイテムデータセットのアイテム説明とアイテム名のデータに基づいています。Amazon Personalize では、テーマはバッチ推論ジョブの出力に含められます。テーマを使用すると、アプリケーション内のテキストやマーケティングメッセージをより説得力のあるものにできます。

例えば、朝食用フードの関連商品のレコメンデーションが表示される場合、Amazon Personalize は元気な朝や朝の必需品などのテーマを生成する場合があります。このテーマを、よく一緒に買われるものなどの一般的なカルーセルタイトルの代わりに使用できます。また、このテーマをプロモーションメールやマーケティングキャンペーンに組み込んで、新しいメニューオプションを用意することもできます。

AWS は Content Generator のテーマを監視しません。テーマの品質を確認するには、推奨アイテムごとに作成されたスコアを使用できます。詳細については、「テーマ付きバッチレコメンデーションのランク付けとスコアリング」を参照してください。

トピック

- サポートされるリージョン
- ガイドラインと要件
- テーマ付きバッチレコメンデーションのランク付けとスコアリング
- テーマ付きバッチレコメンデーションの生成

サポートされるリージョン

Amazon Personalize Content Generator は、次の AWS リージョンでのみ利用できます。

- 米国東部 (バージニア北部)
- 米国西部 (オレゴン)
- アジアパシフィック (東京)

ガイドラインと要件

テーマ付きレコメンデーションを生成するためのガイドラインと要件は次のとおりです。

- 入力ファイルには、最大 100 個の項目を設定できます。バッチレコメンデーションの入力データの詳細については、「バッチレコメンデーション用の入力データを準備します。」を参照してください。
- ソリューションでは Similar-Items レシピを使用する必要があります。
- 次のデータを含むアイテムデータセットが必要です。このデータは、より関連性の高いテーマを作成するのに役立ちます。
  - DESCRIPTION フィールドなどのテキストフィールドが必要です。テキストデータについては、「非構造化テキストメタデータ」を参照してください。
  - TITLE フィールドなど、アイテム名データを含む文字列の列が必要です。

  アイテムデータセットにこのデータがない場合は、追加できます。既存のデータの更新については、「より多くのトレーニングデータをデータセットにインポートする」を参照してください。

テーマ付きバッチレコメンデーションのランク付けとスコアリング

テーマ付きバッチレコメンデーションを取得すると、Amazon Personalize は、各アイテムに対するテーマの関連性に基づいて各アイテムセットをランク付けします。各アイテムには -0.1 から 0.6 までの大まかな範囲のスコアが含まれています。スコアが高いほど、そのアイテムはテーマと密接に関連していることになります。スコアを使用して、テーマとの関連性が強いアイテムのみが表示されるようにしきい値を設定できます。

例えば、Amazon Personalize は `For your sweet tooth` というテーマを返し、関連アイテムとそのスコアは、ハードキャンディー (スコア 0.19884521)、チョコレート (スコア .17664525)、リンゴ (スコア .08994528)、アイスキャンディー (スコア .14294521)、サツマイモ (スコア .07794527)、にんじん (スコア .04994523) の場合があります。アプリケーションでは、スコア `.10` 以上のアイテムのみを対象とし、果物や野菜を除外するルールを追加できます。

次の例は、テーマ付きの映画のレコメンデーションを生成するバッチ推論ジョブの出力形式を示しています。

```
{"input":{"itemId":"40"},"output":{"recommendedItems":
["36","50","44","22","21","29","3","1","2","39"],"theme":"Movies
 with a strong female lead","itemsThemeRelevanceScores":
[0.19994527,0.183059963,0.17478035,0.1618133,0.1574806,0.15468733,0.1499242,0.14353688,0.135314
{"input":{"itemId":"43"},"output":{"recommendedItems":
["50","21","36","3","17","2","39","1","10","5"],"theme":"The best movies of
 1995","itemsThemeRelevanceScores":
[0.184988,0.1795761,0.11143453,0.0989443,0.08258403,0.07952615,0.07115086,0.0621634,-0.138913,-
```

```
...
```

テーマ付きバッチレコメンデーションの生成

テーマ付きバッチレコメンデーションを生成するには、「バッチワークフロー」の説明に従ってバッチワークフローを完了します。入力データは、RELATED_ITEMS レシピと同じ方法で準備します。例については、RELATED_ITEMS レシピを参照してください。

バッチ推論ジョブを作成するときは、テーマ生成を有効にして、アイテムデータセットのアイテムタイトル列を指定します。

- Amazon Personalize コンソールを使用してテーマを生成するバッチ推論ジョブを作成する方法については、「バッチ推論ジョブの作成」を参照してください。
- SDK for Python (Boto3) を使用して、テーマを生成するバッチ推論ジョブを作成する方法を示すコードサンプルについては、「テーマを生成するバッチ推論ジョブの作成」を参照してください。

バッチレコメンデーション用の入力データを準備します。

バッチ推論ジョブは、Amazon S3 バケットからバッチ入力データをインポートし、カスタムソリューションバージョンを使用してアイテムのレコメンデーションを生成してから、そのレコメンデーションを Amazon S3 バケットにエクスポートするツールです。バッチレコメンデーションまたはユーザーセグメントを取得する前に、JSON ファイルを準備して Amazon S3 バケットにアップロードする必要があります。Amazon S3 バケットに出力フォルダを作成するか、個別の出力 Amazon S3 バケットを使用することをお勧めします。その後、同じ入力データの場所を使用して、複数のバッチ推論ジョブを実行できます。

$GENRE などのプレースホルダーパラメータを持つフィルターを使用する場合は、入力 JSON の filterValues オブジェクトの値を指定する必要があります。詳細については、入力 JSON にフィルター値を指定します。を参照してください。

データを準備してインポートするには

1. バッチ入力データをレシピに応じてフォーマットします。Trending-Now レシピではバッチレコメンデーションを取得できません。

   - USER_PERSONALIZATION レシピと Popularity-Count レシピの場合、入力データは userID のリストを含む JSON ファイルです。
   - RELATED_ITEMS レシピの場合、入力データは itemId のリストです。

- PERSONALIZED_RANKING レシピの場合、入力データは userId のリストで、それぞれが itemId のコレクションと組み合わされています。

次のように、新しい行で各行を区切ります。入力データの例については、「バッチ推論ジョブの入力および出力 JSON の例」を参照してください。

2. 入力 JSON を Amazon S3 バケットの入力フォルダーにアップロードします。詳細については、Amazon Simple Storage Service ユーザーガイドの「ドラッグアンドドロップを使用したファイルとフォルダのアップロード」を参照してください。
3. フォルダまたは別の Amazon S3 バケットのいずれかで、出力データ用に個別の場所を作成します。出力 JSON 用に個別の場所を作成することにより、同じ入力データの場所を使用して複数のバッチ推論ジョブを実行できます。
4. バッチ推論ジョブを作成します。Amazon Personalize は、ソリューションバージョンから出力データの場所にレコメンデーションを出力します。

バッチ推論ジョブの入力および出力 JSON の例

使用するレシピに合わせて入力データをどのようにフォーマットする方法。$GENRE などのプレースホルダーパラメータを使用する場合は、入力 JSON の `filterValues` オブジェクトにパラメータの値を指定する必要があります。詳細については、入力 JSON にフィルター値を指定します。を参照してください。

次のセクションでは、バッチ推論ジョブと用に正しくフォーマットされた JSON 入力および出力の例を示します。Trending-Now レシピではバッチレコメンデーションを取得できません。

トピック

- USER_PERSONALIZATION レシピ
- POPULAR_ITEMS レシピ (Popularity-Count のみ)
- PERSONALIZED_RANKING レシピ
- RELATED_ITEMS レシピ

USER_PERSONALIZATION レシピ

以下は、USER_PERSONALIZATION レシピの正しい形式の JSON 入力および出力の例を示しています。User-Personalization-v2 を使用する場合、各推奨項目には、その項目がレコメンデーションに含まれた理由のリストが含まれます。このリストは空にすることができます。考えられる理由については、「」を参照してくださいレコメンデーションの理由 (User-Personalization-v2)。

Input

次のように、各 userId を改行で区切ります。

```
{"userId": "4638"}
{"userId": "663"}
{"userId": "3384"}
...
```

Output

```
{"input":{"userId":"4638"},"output":{"recommendedItems":
["63992","115149","110102","148626","148888","31685","102445","69526","92535","143355","6237
[0.0152238,0.0069081,0.0068222,0.006394,0.0059746,0.0055851,0.0049357,0.0044644,0.0042968,0.
{"input":{"userId":"663"},"output":{"recommendedItems":
["368","377","25","780","1610","648","1270","6","165","1196","1097","300","1183","608","104"
[0.0406197,0.0372557,0.0254077,0.0151975,0.014991,0.0127175,0.0124547,0.0116712,0.0091098,0.
{"input":{"userId":"3384"},"output":{"recommendedItems":
["597","21","223","2144","208","2424","594","595","920","104","520","367","2081","39","1035"
[0.0241061,0.0119394,0.0118012,0.010662,0.0086972,0.0079428,0.0073218,0.0071438,0.0069602,0.
...
```

POPULAR_ITEMS レシピ (Popularity-Count のみ)

次の例は、Popularity-Count レシピ用に正しくフォーマットされた JSON 入力および出力を示しています。Trending-Now レシピではバッチレコメンデーションを取得できません。

Input

次のように、各 userId を改行で区切ります。

```
{"userId": "12"}
{"userId": "105"}
{"userId": "41"}
...
```

Output

```
{"input": {"userId": "12"}, "output": {"recommendedItems": ["105", "106", "441"]}}
{"input": {"userId": "105"}, "output": {"recommendedItems": ["105", "106", "441"]}}
{"input": {"userId": "41"}, "output": {"recommendedItems": ["105", "106", "441"]}}
```

```
...
```

PERSONALIZED_RANKING レシピ

PERSONALIZED_RANKING レシピ用に正しくフォーマットされた JSON 入力および出力の例を次に示します。

Input

次のように、各 `userId` とランク付けする `itemIds` のリストを改行で区切ります。

```
{"userId": "891", "itemList": ["27", "886", "101"]}
{"userId": "445", "itemList": ["527", "55", "901"]}
{"userId": "71", "itemList": ["27", "351", "101"]}
...
```

Output

```
{"input":{"userId":"891","itemList":["27","886","101"]},"output":
{"recommendedItems":["27","101","886"],"scores":[0.48421,0.28133,0.23446]}}
{"input":{"userId":"445","itemList":["527","55","901"]},"output":
{"recommendedItems":["901","527","55"],"scores":[0.46972,0.31011,0.22017]}}
{"input":{"userId":"71","itemList":["29","351","199"]},"output":{"recommendedItems":
["351","29","199"],"scores":[0.68937,0.24829,0.06232]}}
...
```

RELATED_ITEMS レシピ

次の例は、RELATED_ITEMS レシピ用に正しくフォーマットされた JSON 入力および出力を示しています。

Input

次のように、各 `itemId` を改行で区切ります。

```
{"itemId": "105"}
{"itemId": "106"}
{"itemId": "441"}
...
```

Output

```
{"input": {"itemId": "105"}, "output": {"recommendedItems": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedItems": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedItems": ["2", "442", "435"]}}
...
```

次の例は、テーマ付き Similar-Items レシピ用に正しくフォーマットされた JSON 入力および出力を示しています。

Input

次のように、各 itemId を改行で区切ります。

```
{"itemId": "40"}
{"itemId": "43"}
...
```

Output

```
{"input":{"itemId":"40"},"output":{"recommendedItems":
["36","50","44","22","21","29","3","1","2","39"],"theme":"Movies
 with a strong female lead","itemsThemeRelevanceScores":
[0.19994527,0.183059963,0.17478035,0.1618133,0.1574806,0.15468733,0.1499242,0.14353688,0.135
{"input":{"itemId":"43"},"output":{"recommendedItems":
["50","21","36","3","17","2","39","1","10","5"],"theme":"The best movies of
 1995","itemsThemeRelevanceScores":
[0.184988,0.1795761,0.11143453,0.0989443,0.08258403,0.07952615,0.07115086,0.0621634,-0.13891
...
```

バッチ推論ジョブの作成

バッチ推論ジョブを作成して、Amazon S3 からの入力データに基づいてユーザー向けのバッチアイテムのレコメンデーションを取得します。入力データは、JSON 形式のユーザーもしくはアイテム (またはその両方) のリストにすることができます。バッチ推論ジョブは、Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用して作成できます。

バッチ推論ジョブを作成するときは、入力場所と出力場所への Amazon S3 パスを指定します。Amazon S3 はプレフィックスベースです。入力データの場所にプレフィックスを指定する

と、Amazon Personalize はそのプレフィックスに一致するすべてのファイルを入力データとして使用します。例えば、s3://<name of your S3 bucket>/folderName を指定し、バケットにもパスが s3://<name of your S3 bucket>/folderName_test のフォルダがある場合、Amazon Personalize は両方のフォルダのすべてのファイルを入力データとして使用します。特定のフォルダ内のファイルのみを入力データとして使用するには、Amazon S3 パスの末尾に / のようなプレフィックス区切り文字を付けます: s3://<name of your S3 bucket>/folderName/ Amazon S3 がオブジェクトを整理する方法の詳細については、「オブジェクトの整理、一覧表示、操作」を参照してください。

許可の要件、レコメンデーションスコアリング、入力データの準備とインポートなど、Amazon Personalize のバッチワークフローの詳細については、「バッチレコメンデーションの取得」を参照してください。

トピック

- バッチ推論ジョブの作成 (コンソール)
- バッチ推論ジョブの作成 (AWS CLI)
- バッチ推論ジョブの作成 (AWS SDKs)

バッチ推論ジョブの作成 (コンソール)

バッチレコメンデーション用の入力データを準備します。 を完了すると、バッチ推論ジョブを作成する準備が整います。この手順は、ソリューションとソリューションバージョン (トレーニング済みモデル) を既に作成していることを前提としています。

バッチ推論ジョブを作成するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。
3. ナビゲーションペインの [カスタムリソース] で、[バッチ推論ジョブ] を選択します。
4. [Create batch inference job (バッチ推論ジョブの作成)] を選択します。
5. バッチ推論ジョブタイプを選択します。

    - テーマなしでアイテムのレコメンデーションを生成するには、[アイテムのレコメンデーション] を選択します。
    - Similar-Items レシピを使用し、類似アイテムのグループにわかりやすいテーマを追加する場合は、[Content Generator のテーマ別レコメンデーション] を選択します。テーマを生成する

には、アイテム名データとテキストデータを含むアイテムデータセットが必要です。詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

6. [Batch inference job details (バッチ推論ジョブ詳細)] の [Batch inference job name (バッチ推論ジョブ名)] で、バッチ推論ジョブの名前を指定します。
7. [Solution] (ソリューション) で、ソリューションを選択してから、レコメンデーションの生成に使用する [Solution version ID] (ソリューションバージョン ID) を選択します。
8. [Number of results] (結果の数) で、オプションで、入力データの各行についてのレコメンデーション数を指定します。デフォルトは 25 です。
9. バッチジョブでテーマ付きレコメンデーションが生成される場合は、[テーマ別レコメンデーションの詳細] で、アイテムデータセット内のアイテムの名前またはタイトルを含む列を選択します。このデータは、より関連性の高いテーマを作成するのに役立ちます。詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。
10. [入力ソース] で、入力ファイルへの Amazon S3 パスを指定します。

    次の構文を使用します: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`**

    入力データは、ソリューションで使用するレシピに適した形式である必要があります。入力データの例については、「バッチ推論ジョブの入力および出力 JSON の例」を参照してください。
11. 復号キー で、バケットの暗号化に独自の AWS KMS キーを使用する場合は、キーの Amazon リソースネーム (ARN) を指定します。Amazon Personalize は、キーを使用する許可を持っている必要があります。許可の付与の詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。
12. [出力先] で、出力場所へのパスを指定します。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。

    次の構文を使用します: **`s3://<name of your S3 bucket>/<output folder name>/`**
13. 暗号化キー では、暗号化に独自の AWS KMS キーを使用する場合は、キーの ARN を指定します。Amazon Personalize は、キーを使用する許可を持っている必要があります。許可の付与の詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。
14. [IAM service role] (IAM サービスロール) で、設定時に Amazon Personalize 用に作成した IAM サービスロールを選択します。このロールには、入力および出力の Amazon S3 バケットへの読み取りおよび書き込みアクセスがそれぞれ必要です。

15. [フィルター] では、フィルターを選択し、バッチレコメンデーションにフィルターを適用することもできます。フィルターでプレースホルダーパラメータを使用する場合は、パラメータの値が入力 JSON に含まれていることを確認してください。詳細については、入力 JSON にフィルター値を指定します。を参照してください。
16. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
17. [Create batch inference job (バッチ推論ジョブの作成)] を選択します。バッチ推論ジョブの作成が開始され、[Batch inference jobs (バッチ推論ジョブ)] ページが開いて、[Batch inference job detail (バッチ推論ジョブ詳細)] セクションが表示されます。

    バッチ推論ジョブのステータスが [Active] (アクティブ) に変わると、指定した出力 Amazon S3 バケットからジョブの出力を取得できます。出力ファイルの名前は *input-name*.out という形式になります。

バッチ推論ジョブの作成 (AWS CLI)

バッチレコメンデーション用の入力データを準備します。 を完了すると、CreateBatchInferenceJob 操作を使用してバッチ推論ジョブを作成する準備が整います。

トピック

- バッチ推論ジョブの作成
- テーマを生成するバッチ推論ジョブの作成

バッチ推論ジョブの作成

`create-batch-inference-job` コマンドを使用して、バッチ推論ジョブを作成できます。ジョブ名を指定し、`Solution version ARN` をソリューションバージョンの Amazon リソースネーム (ARN) に置き換えます。また、`IAM service role ARN` を、設定中に Amazon Personalize 用に作成した IAM サービスロールの ARN に置き換えます。このロールには、入力および出力の Amazon S3 バケットへの読み取りおよび書き込みアクセスがそれぞれ必要です。オプションで、レコメンデーションをフィルタリングするためのフィルター ARN を選択します。フィルターでプレースホルダーパラメータを使用する場合は、パラメータの値が入力 JSON に含まれていることを確認してください。詳細については、バッチレコメンデーションとユーザーセグメントのフィルタリング (カスタムリソース)を参照してください。

`S3 input path` と `S3 output path` を、入力ファイルへの Amazon S3 パスと出力場所に置き換えます。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧

めします。入力および出力の場所には、**s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json** および **s3://<name of your S3 bucket>/<output folder name>/** の構文を使用します。

この例には、オプションの User-Personalization レシピ固有の `itemExplorationConfig` ハイパーパラメータ (`explorationWeight` および `explorationItemAgeCutOff`) が含まれています。オプションで、`explorationWeight` および `explorationItemAgeCutOff` の値を含めて、探索を設定します。詳細については、「User-Personalization レシピ」を参照してください。

```
aws personalize create-batch-inference-job \
--job-name Batch job name \
--solution-version-arn Solution version ARN \
--filter-arn Filter ARN \
--job-input s3DataSource={path=s3://S3 input path} \
--job-output s3DataDestination={path=s3://S3 output path} \
--role-arn IAM service role ARN \
--batch-inference-job-config "{\"itemExplorationConfig\":{\"explorationWeight\":
\"0.3\",\"explorationItemAgeCutOff\":\"30\"}}"
```

テーマを生成するバッチ推論ジョブの作成

類似アイテムのテーマを生成するには、Similar-Items レシピを使用し、アイテムデータセットにテキストフィールドとアイテム名データの列が必要です。テーマ付きレコメンデーションの詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

次のコードは、テーマ付きレコメンデーションを生成するバッチ推論ジョブを作成します。`batch-inference-job-mode` は `THEME_GENERATION` に設定したままにしておきます。`COLUMN_NAME` をアイテム名データが格納されている列の名前に置き換えます。

```
aws personalize create-batch-inference-job \
--job-name Themed batch job name \
--solution-version-arn Solution version ARN \
--filter-arn Filter ARN \
--job-input s3DataSource={path=s3://S3 input path} \
--job-output s3DataDestination={path=s3://S3 output path} \
--role-arn IAM service role ARN \
--batch-inference-job-mode THEME_GENERATION \
--theme-generation-config "{\"fieldsForThemeGeneration\": {\"itemName\":
\"COLUMN_NAME\"}}"
```

バッチ推論ジョブの作成 (AWS SDKs)

バッチレコメンデーション用の入力データを準備します。 を完了すると、CreateBatchInferenceJob 操作を使用してバッチ推論ジョブを作成する準備が整います。

トピック

- バッチ推論ジョブの作成
- テーマを生成するバッチ推論ジョブの作成

バッチ推論ジョブの作成

次のコードを使用して、バッチ推論ジョブを作成できます。ジョブ名、ソリューションバージョンの Amazon リソースネーム (ARN)、設定中に Amazon Personalize 用に作成した IAM サービスロールの ARN を指定します。このロールには、入力および出力の Amazon S3 バケットへの読み取りおよび書き込みアクセスが必要です。

出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。入力および出力の場所には、**s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json** および **s3://<name of your S3 bucket>/<output folder name>/** の構文を使用します。

numResults には、入力データの各行に Amazon Personalize に予測させたいアイテムの数を指定します。オプションで、レコメンデーションをフィルタリングするためのフィルター ARN を選択します。フィルターでプレースホルダーパラメータを使用する場合は、パラメータの値が入力 JSON に含まれていることを確認してください。詳細については、「バッチレコメンデーションとユーザーセグメントのフィルタリング (カスタムリソース)」を参照してください。

SDK for Python (Boto3)

この例には、オプションの User-Personalization レシピ固有の itemExplorationConfig ハイパーパラメータ (explorationWeight および explorationItemAgeCutOff) が含まれています。オプションで、explorationWeight および explorationItemAgeCutOff の値を含めて、探索を設定します。詳細については、「User-Personalization レシピ」を参照してください。

```
import boto3

personalize_rec = boto3.client(service_name='personalize')

personalize_rec.create_batch_inference_job (
```

```
    solutionVersionArn = "Solution version ARN",
    jobName = "Batch job name",
    roleArn = "IAM service role ARN",
    filterArn = "Filter ARN",
    batchInferenceJobConfig = {
        # optional USER_PERSONALIZATION recipe hyperparameters
        "itemExplorationConfig": {
            "explorationWeight": "0.3",
            "explorationItemAgeCutOff": "30"
        }
    },
    jobInput =
       {"s3DataSource": {"path": "s3://<name of your S3 bucket>/<folder name>/<input
 JSON file name>.json"}},
    jobOutput =
       {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder
 name>/"}}
)
```

SDK for Java 2.x

この例には、オプションの User-Personalization レシピ固有の `itemExplorationConfig` フィールド (`explorationWeight` および `explorationItemAgeCutOff`) が含まれています。オプションで、`explorationWeight` および `explorationItemAgeCutOff` の値を含めて、探索を設定します。詳細については、「User-Personalization レシピ」を参照してください。

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
 personalizeClient,
                                                         String solutionVersionArn,
                                                         String jobName,
                                                         String filterArn,
                                                         String
 s3InputDataSourcePath,
                                                         String
 s3DataDestinationPath,
                                                         String roleArn,
                                                         String explorationWeight,
                                                         String
 explorationItemAgeCutOff) {

  long waitInMilliseconds = 60 * 1000;
  String status;
  String batchInferenceJobArn;
```

```
 try {
     // Set up data input and output parameters.
     S3DataConfig inputSource = S3DataConfig.builder()
             .path(s3InputDataSourcePath)
             .build();
     S3DataConfig outputDestination = S3DataConfig.builder()
             .path(s3DataDestinationPath)
             .build();

     BatchInferenceJobInput jobInput = BatchInferenceJobInput.builder()
             .s3DataSource(inputSource)
             .build();
     BatchInferenceJobOutput jobOutputLocation = BatchInferenceJobOutput.builder()
             .s3DataDestination(outputDestination)
             .build();

     // Optional code to build the User-Personalization specific item exploration
config.
     HashMap<String, String> explorationConfig = new HashMap<>();

     explorationConfig.put("explorationWeight", explorationWeight);
     explorationConfig.put("explorationItemAgeCutOff", explorationItemAgeCutOff);

     BatchInferenceJobConfig jobConfig = BatchInferenceJobConfig.builder()
             .itemExplorationConfig(explorationConfig)
             .build();
     // End optional User-Personalization recipe specific code.

     CreateBatchInferenceJobRequest createBatchInferenceJobRequest =
CreateBatchInferenceJobRequest.builder()
             .solutionVersionArn(solutionVersionArn)
             .jobInput(jobInput)
             .jobOutput(jobOutputLocation)
             .jobName(jobName)
             .filterArn(filterArn)
             .roleArn(roleArn)
             .batchInferenceJobConfig(jobConfig)   // Optional
             .build();

     batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
             .batchInferenceJobArn();
```

```
      DescribeBatchInferenceJobRequest describeBatchInferenceJobRequest =
 DescribeBatchInferenceJobRequest.builder()
              .batchInferenceJobArn(batchInferenceJobArn)
              .build();

      long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

      // wait until the batch inference job is complete.
      while (Instant.now().getEpochSecond() < maxTime) {

          BatchInferenceJob batchInferenceJob = personalizeClient
                  .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                  .batchInferenceJob();

          status = batchInferenceJob.status();
          System.out.println("Batch inference job status: " + status);

          if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
              break;
          }
          try {
              Thread.sleep(waitInMilliseconds);
          } catch (InterruptedException e) {
              System.out.println(e.getMessage());
          }
      }
      return batchInferenceJobArn;

  } catch (PersonalizeException e) {
      System.out.println(e.awsErrorDetails().errorMessage());
  }
  return "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});
```

```
// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: 'JOB_NAME',
  jobInput: {         /* required */
    s3DataSource: {   /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  jobOutput: {         /* required */
    s3DataDestination: {   /* required */
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional integer*/
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateBatchInferenceJobCommand(createBatchInferenceJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

バッチジョブの処理が完了するまでに時間がかかる場合があります。DescribeBatchInferenceJob を呼び出し、入力パラメータとして batchRecommendationsJobArn を渡すことで、ジョブのステータスを確認できます。を呼び出すことで、 AWS 環境内のすべての Amazon Personalize バッチ推論ジョブを一覧表示することもできますListBatchInferenceJobs。

テーマを生成するバッチ推論ジョブの作成

類似アイテムのテーマを生成するには、Similar-Items レシピを使用し、アイテムデータセットにテキストフィールドとアイテム名データの列が必要です。テーマ付きレコメンデーションの詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

次のコードは、テーマ付きレコメンデーションを生成するバッチ推論ジョブを作成します。batchInferenceJobMode は "THEME_GENERATION" に設定したままにしておきます。COLUMNN_NAME をアイテム名データが格納されている列の名前に置き換えます。

```
import boto3

personalize_rec = boto3.client(service_name='personalize')

personalize_rec.create_batch_inference_job (
    solutionVersionArn = "Solution version ARN",
    jobName = "Batch job name",
    roleArn = "IAM service role ARN",
    filterArn = "Filter ARN",
    batchInferenceJobMode = "THEME_GENERATION",
    themeGenerationConfig = {
      "fieldsForThemeGeneration": {
          "itemName": "COLUMN_NAME"
      }
    },
    jobInput =
       {"s3DataSource": {"path": "s3://<name of your S3 bucket>/<folder name>/<input
 JSON file name>.json"}},
    jobOutput =
       {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder
 name>/"}}
)
```

Batch 推論ジョブの出力例

バッチ推論ジョブを作成する際は、Amazon S3 バケットからバッチ入力データをインポートし、ソリューションバージョンを使用して商品のレコメンデーションを生成してから、そのレコメンデーションを Amazon S3 バケットに JSON 形式でエクスポートします。

次のセクションでは、バッチ推論ジョブ用の出力ファイルの例を recipe タイプ別に示します。Trending-Now レシピまたは Next-Best-Action レシピではバッチレコメンデーションを取得することはできません。

トピック

USER_PERSONALIZATION の recipe

USER_PERSONALIZATION レシピの出力 JSON ファイルの例を以下に示します。

```
{"input":{"userId":"4638"},"output":{"recommendedItems":
["63992","115149","110102","148626","148888","31685","102445","69526","92535","143355","62374",
[0.0152238,0.0069081,0.0068222,0.006394,0.0059746,0.0055851,0.0049357,0.0044644,0.0042968,0.004
{"input":{"userId":"663"},"output":{"recommendedItems":
["368","377","25","780","1610","648","1270","6","165","1196","1097","300","1183","608","104","4
[0.0406197,0.0372557,0.0254077,0.0151975,0.014991,0.0127175,0.0124547,0.0116712,0.0091098,0.008
{"input":{"userId":"3384"},"output":{"recommendedItems":
["597","21","223","2144","208","2424","594","595","920","104","520","367","2081","39","1035","2
[0.0241061,0.0119394,0.0118012,0.010662,0.0086972,0.0079428,0.0073218,0.0071438,0.0069602,0.005
...
```

POPULAR_ITEMS レシピ

以下の例は、 Popularity-Count レシピの出力 JSON ファイルの形式を示しています。Trending-Now レシピではバッチレコメンデーションを取得できません。

```
{"input": {"userId": "12"}, "output": {"recommendedItems": ["105", "106", "441"]}}
{"input": {"userId": "105"}, "output": {"recommendedItems": ["105", "106", "441"]}}
{"input": {"userId": "41"}, "output": {"recommendedItems": ["105", "106", "441"]}}
...
```

PERSONALIZED_RANKING レシピ

以下の例は、PERSONALIZED_RANKING レシピの出力 JSON ファイルの形式を示しています。

```
{"input":{"userId":"891","itemList":["27","886","101"]},"output":{"recommendedItems":
["27","101","886"],"scores":[0.48421,0.28133,0.23446]}}
{"input":{"userId":"445","itemList":["527","55","901"]},"output":{"recommendedItems":
["901","527","55"],"scores":[0.46972,0.31011,0.22017]}}
```

```
{"input":{"userId":"71","itemList":["29","351","199"]},"output":{"recommendedItems":
["351","29","199"],"scores":[0.68937,0.24829,0.06232]}}
...
```

RELATED_ITEMS レシピ

以下の例は、RELATED_ITEMS レシピの出力 JSON ファイルの形式を示しています。

```
{"input": {"itemId": "105"}, "output": {"recommendedItems": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedItems": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedItems": ["2", "442", "435"]}}
...
```

次の例は、テーマ付き Similar-Items レシピの出力 JSON ファイルの形式を示しています。テーマ付きレコメンデーションの詳細については、「Content Generator のテーマ付きバッチレコメンデーション」を参照してください。

```
{"input":{"itemId":"40"},"output":{"recommendedItems":
["36","50","44","22","21","29","3","1","2","39"],"theme":"Movies
 with a strong female lead","itemsThemeRelevanceScores":
[0.19994527,0.183059963,0.17478035,0.1618133,0.1574806,0.15468733,0.1499242,0.14353688,0.135314
{"input":{"itemId":"43"},"output":{"recommendedItems":
["50","21","36","3","17","2","39","1","10","5"],"theme":"The best movies of
 1995","itemsThemeRelevanceScores":
[0.184988,0.1795761,0.11143453,0.0989443,0.08258403,0.07952615,0.07115086,0.0621634,-0.138913,-
...
```

ユーザーセグメントを取得する

ユーザーセグメント取得するには、バッチセグメントジョブを使用します。バッチセグメントジョブは、Amazon S3 バケットからバッチ入力データをインポートし、USER_SEGMENTATION レシピでトレーニングされたソリューションバージョンを使用して、入力データの各行のユーザーセグメントを生成するツールです。

レシピに応じて、入力データはアイテムまたはアイテムメタデータ属性のリスト (JSON 形式) です。アイテム属性については、入力データに、複数のメタデータ属性に基づいてユーザーセグメントを作成するための式を含めることができます。バッチセグメントジョブは、出力の Amazon S3 バケットにユーザーセグメントをエクスポートします。各ユーザーセグメントは、各ユーザーが入力データ内のアイテムを操作する蓋然性に基づいて、降順にソートされます。

ユーザーセグメントを生成する際、Amazon Personalize は、一括インポートと個別インポートのデータセット内のデータを考慮します。

- バルクデータの場合、Amazon Personalize は前回のフルソリューションバージョントレーニングで提示されたバルクデータのみを使用してセグメントを生成します。また、インポートモードが FULL で (既存のデータを置き換えて) インポートしたバルクデータのみを使用します。
- 個々のデータインポートオペレーションのデータについては、Amazon Personalize は前回のフルソリューションバージョントレーニングで得られたデータを使用してユーザーセグメントを生成します。新しいレコードがユーザーセグメントに影響を与えるには、新しいソリューションバージョンを作成してから、バッチセグメントジョブを作成します。

ユーザーセグメントの生成は次のように行われます。

1. 入力データを JSON 形式で準備して Amazon S3 バケットにアップロードします。入力データの形式は、使用するレシピと作成するジョブによって異なります。ユーザーセグメントの入力データを準備しています。 を参照してください。
2. フォルダまたは別の Amazon S3 バケットのいずれかで、出力データ用に個別の場所を作成します。
3. バッチセグメントジョブを作成します。バッチセグメントジョブの作成 を参照してください。
4. バッチセグメントジョブが完了したら、Amazon S3 の出力場所からユーザーセグメントを取得します。

トピック

- ガイドラインと要件
- ユーザーセグメントの入力データを準備しています。
- バッチセグメントジョブの作成
- バッチセグメントジョブの出力例

ガイドラインと要件

バッチセグメントを一括取得するためのガイドラインと要件は次のとおりです。

- USER_SEGMENTATION レシピを使用する必要があります。
- Amazon Personalize の IAM サービスロールには、Amazon S3 バケットを読み取り、ファイルを追加するための許可が必要です。許可の付与については、「バッチワークフロー用のサービス

ロールのポリシー」を参照してください。バケットの許可の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「ユーザーポリシーの例」を参照してください。

暗号化に AWS Key Management Service（ AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize IAM サービスロールに、 キーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

- バッチ推論ジョブを作成する前に、カスタムソリューションとソリューションバージョンを作成する必要があります。ただし、Amazon Personalize のキャンペーンを作成する必要はありません。ドメインデータセットグループを作成した場合は、引き続きカスタムリソースを作成できます。
- 入力データは、「ユーザーセグメントの入力データを準備しています。」で説明されている形式にする必要があります。
- Item-Attribute-Affinity レシピを使用する場合、入力データの属性に、製品の説明などの非構造化テキストアイテムメタデータを含めることはできません。
- プレースホルダーパラメータを含むフィルタを使用する場合は、`filterValues` オブジェクトの入力データにパラメータの値を含める必要があります。詳細については、「入力 JSON にフィルター値を指定します。」を参照してください。
- 出力データには入力データとは異なるの場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。

ユーザーセグメントの入力データを準備しています。

バッチセグメントジョブは、ソリューションバージョンを使用して、入力 JSON ファイルで提供するデータに基づいてユーザーセグメントを作成します。バッチユーザーセグメントを取得する前に、JSON ファイルを準備して Amazon S3 バケットにアップロードする必要があります。Amazon S3 バケットに出力フォルダを作成するか、個別の出力 Amazon S3 バケットを使用することをお勧めします。その後、同じ入力データの場所を使用して、複数のバッチ推論ジョブを実行できます。

`$GENRE` などのプレースホルダーパラメータを持つフィルターを使用する場合は、入力 JSON の `filterValues` オブジェクトの値を指定する必要があります。詳細については、入力 JSON にフィルター値を指定します。を参照してください。

データを準備してインポートするには

1. ソリューションで使用するレシピに応じて、バッチ入力データをフォーマットします。入力データ要素を新しい行で区切ります。入力データは itemId のリスト (Item-Affinity) またはアイテム属性 (Item-Attribute-Affinity) のいずれかです。

アイテム属性については、入力データには、クエリごとに複数のアイテムまたは属性用にユーザーを取得するための AND 演算子を持つ論理式を含めることができます。詳細については、「Item-Attribute-Affinity レシピのアイテム属性の指定」を参照してください。

両方のレシピの入力データの例については、「」を参照してくださいバッチセグメントジョブの入力および出力 JSON の例。

2. 入力 JSON を Amazon S3 バケットの入力フォルダーにアップロードします。詳細については、Amazon Simple Storage Service ユーザーガイドの「ドラッグアンドドロップを使用したファイルとフォルダのアップロード」を参照してください。
3. フォルダまたは別の Amazon S3 バケットのいずれかで、出力データ用に個別の場所を作成します。出力 JSON 用に個別の場所を作成することにより、同じ入力データの場所を使用して複数のバッチセグメントジョブを実行できます。

入力データを準備して Amazon S3 バケットにアップロードしたら、バッチセグメントジョブでユーザーセグメントを生成する準備が整いました。詳細については、「バッチセグメントジョブの作成」を参照してください。

トピック

- Item-Attribute-Affinity レシピのアイテム属性の指定
- バッチセグメントジョブの入力および出力 JSON の例

Item-Attribute-Affinity レシピのアイテム属性の指定

Item-Attribute-Affinity レシピを使用する場合、入力データはアイテム属性のリストです。メタデータのさまざまな列を混在させることができます。例えば、1 つの行が数値列で、次の行がカテゴリ列である場合があります。非構造化テキストアイテムメタデータをアイテム属性として使用することはできません。

入力項目のメタデータには、複数の属性のユーザーセグメントを取得するための AND演算子を含む論理式を含めることができます。例えば、入力データの行は `{"itemAttributes": "ITEMS.genres = "\Comedy\" AND ITEMS.genres = "\Action\""}` または `{"itemAttributes": "ITEMS.genres = "\Comedy\" AND ITEMS.audience = "\teen \""}` です。

2 つの属性を AND 演算子と組み合わせると、ユーザーのインタラクション履歴に基づいて、両方の属性を持つアイテムとインタラクションする可能性が高いユーザーを含むユーザーセグメントを作成

します。フィルター式 (文字列の等式に IN 演算子を使用) とは異なり、バッチセグメント入力式は、文字列の照合のために等式の = 記号のみをサポートします。

バッチセグメントジョブの入力および出力 JSON の例

バッチセグメントジョブについては、入力データは itemId のリスト (Item-Affinity レシピ) またはアイテム属性 (Item-Attribute-Affinity) のいずれかです。入力データの各行は、個別の推論クエリです。各ユーザーセグメントは、各ユーザーがインベントリ内のアイテムを操作する蓋然性に基づいて、降順にソートされます。

$GENRE などのプレースホルダーパラメータを持つフィルターを使用する場合は、入力 JSON の `filterValues` オブジェクトの値を指定します。詳細については、入力 JSON にフィルター値を指定します。を参照してください。

レシピごとに整理されたバッチセグメントジョブ用に正しくフォーマットされた JSON 入力および出力を次に示します。

Item-Affinity

Input

入力データには、最大 500 個のアイテムを含めることができます。次のように、各 `itemId` を改行で区切ります。

```
{"itemId": "105"}
{"itemId": "106"}
{"itemId": "441"}
...
```

Output

```
{"input": {"itemId": "105"}, "output": {"recommendedUsers": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedUsers": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedUsers": ["2", "442", "435"]}}
...
```

Item-Attribute-Affinity

Input

入力データには最大 10 個のクエリを含めることができます。各クエリはテキスト以外の項目属性です。次のように、各属性を改行で区切ります。

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action\""}
{"itemAttributes": "ITEMS.genres = \"Comedy\""}
{"itemAttributes": "ITEMS.genres = \"Horror\" AND ITEMS.genres = \"Action\""}
...
```

Output

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action\"",
 "output": {"recommendedUsers": ["25", "78", "108"]}}
{"itemAttributes": "ITEMS.genres = \"Adventure\"", "output": {"recommendedUsers":
 ["87", "31", "129"]}}
{"itemAttributes": "ITEMS.genres = \"Horror\" AND ITEMS.genres = \"Action\"",
 "output": {"recommendedUsers": ["8", "442", "435"]}}
...
```

バッチセグメントジョブの作成

USER_SEGMENTATION レシピを使用した場合は、バッチセグメントジョブを作成して、ソリューションバージョンでユーザーセグメントを取得できます。各ユーザーセグメントは、各ユーザーがインベントリ内のアイテムを操作する蓋然性に基づいて、降順にソートされます。レシピに応じて、入力データはアイテム (Item-Affinity レシピ) またはアイテム属性 (Item-Attribute-Affinity レシピ) のリスト (JSON 形式) である必要があります。Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用してバッチセグメントジョブを作成できます。

バッチセグメントジョブを作成するときは、入力場所と出力場所への Amazon S3 パスを指定します。Amazon S3 はプレフィックスベースです。入力データの場所にプレフィックスを指定すると、Amazon Personalize はそのプレフィックスに一致するすべてのファイルを入力データとして使用します。例えば、`s3://<name of your S3 bucket>/folderName` を指定し、バケットにもパスが `s3://<name of your S3 bucket>/folderName_test` のフォルダがある場合、Amazon Personalize は両方のフォルダのすべてのファイルを入力データとして使用します。特定のフォルダ内のファイルのみを入力データとして使用するには、Amazon S3 パスの末尾に / のようなプレフィックス区切り文字を付けます: `s3://<name of your S3 bucket>/folderName/` Amazon S3 がオブジェクトを整理する方法の詳細については、「オブジェクトの整理、一覧表示、操作」を参照してください。

トピック

- バッチセグメントジョブの作成 (コンソール)
- バッチセグメントジョブの作成 (AWS CLI)
- バッチセグメントジョブの作成 (AWS SDKs)

バッチセグメントジョブの作成 (コンソール)

バッチレコメンデーション用の入力データを準備します。 を完了すると、バッチセグメントジョブを作成する準備が整います。この手順は、USER_SEGEMENTATION レシピを使用してソリューションとソリューションバージョン (トレーニング済みモデル) を既に作成していることを前提としています。

バッチセグメントジョブを作成するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Datasets group] (データセットグループ) のページで、データセットグループを選択します。
3. ナビゲーションペインで [batch segment jobs] (バッチセグメントジョブ) を選択してから、[Create batch segment job] (バッチセグメントジョブを作成) を選択します。
4. [batch segment job details] (バッチセグメントジョブの詳細) で、[Batch segment job name] (バッチセグメントジョブ名) に、バッチセグメントジョブの名前を指定します。
5. [Solution] (ソリューション) で、ソリューションを選択してから、レコメンデーションの生成に使用する [Solution version ID] (ソリューションバージョン ID) を選択します。バッチセグメントジョブは、USER_SEGEMENTATION レシピを使用した場合にのみ作成できます。
6. [Number of users] (ユーザーの数) で、オプションで、Amazon Personalize が各ユーザーセグメントについて生成するユーザーの数を指定します。デフォルトは 25 です。最大数は 500 万です。
7. [Input source] (入力ソース) で、入力ファイルへの Amazon S3 パスを指定するか、[Browse S3] (S3 を参照) を使用して Amazon S3 バケットを選択します。

   次の構文を使用します: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`**

   入力データは、ソリューションで使用するレシピに適した形式である必要があります。入力データの例については、「バッチセグメントジョブの入力および出力 JSON の例」を参照してください。

8. [Output destination] (出力先) で、出力場所へのパスを指定するか、[Browse S3] (S3 を参照) を使用して Amazon S3 バケットを選択します。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。

   次の構文を使用します: **`s3://<name of your S3 bucket>/<output folder name>/`**
9. [IAM role] (IAM ロール) で、以下のいずれかのオプションを選択します。

   - [Create and use new service role] (新しいサービスロールを作成および使用) を選択し、[Service role name] (サービスロール名) を入力して新しいロールを作成するか、
   - 正しい許可を持つロールを既に作成している場合は、[Use an existing service role] (既存のサービスロールを使用) を選択し、IAM ロールを選択します。

   使用するロールには、入力および出力の Amazon S3 バケットへの読み取りおよび書き込みアクセスがそれぞれ必要です。
10. フィルター構成では、オプションでフィルターを選択し、ユーザーセグメントにフィルターを適用します。フィルタでプレースホルダーパラメータを使用する場合は、パラメータの値が入力 JSON に含まれていることを確認してください。詳細については、入力 JSON にフィルター値を指定します。を参照してください。
11. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
12. [Create batch segment job] (バッチセグメントジョブを登録) を選択します。バッチセグメントジョブの作成が開始され、[Batch segment jobs] (バッチセグメントジョブ) のページが表示されます。当該ページには、[Batch segment job detail] (バッチセグメントジョブの詳細) のセクションが表示されます。
13. バッチセグメントジョブのステータスが [Active] (アクティブ) に変わると、指定した出力 Amazon S3 バケットからジョブの出力を取得できます。出力ファイルの名前は *`input-name`*`.out` という形式になります。

バッチセグメントジョブの作成 (AWS CLI)

バッチレコメンデーション用の入力データを準備します。 を完了すると、次の `create-batch-segment-job` コードを使用してバッチセグメントジョブを作成する準備が整います。ジョブ名を指定し、`Solution version ARN` をソリューションバージョンの Amazon リソースネーム (ARN) に置き換えます。また、`IAM service role ARN` を、設定中に Amazon Personalize 用に作成した IAM サービスロールの ARN に置き換えます。このロールには、入力および出力の Amazon S3 バケットへの読み取りおよび書き込みアクセスがそれぞれ必要です。`num-results` では、入力デー

タの各行に Amazon Personalize に予測させたいユーザーの人数を指定します。デフォルトは 25 です。最大数は 500 万です。オプションで `filter-arn` を指定してユーザーセグメントを絞り込みます。フィルタでプレースホルダーパラメータを使用する場合は、パラメータの値が入力 JSON に含まれていることを確認してください。詳細については、バッチレコメンデーションとユーザーセグメントのフィルタリング (カスタムリソース)を参照してください。

`S3 input path` と `S3 output path` を、入力ファイルへの Amazon S3 パスと出力場所に置き換えます。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。入力および出力の場所には、**`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`** および **`s3://<name of your S3 bucket>/<output folder name>/`** の構文を使用します。

```
aws personalize create-batch-segment-job \
                --job-name Job name \
                --solution-version-arn Solution version ARN \
                --num-results The number of predicted users \
                --filter-arn Filter ARN \
                --job-input s3DataSource={path=s3://S3 input path} \
                --job-output s3DataDestination={path=s3://S3 output path} \
                --role-arn IAM service role ARN
{
   "batchSegmentJobArn": "arn:aws:personalize:us-west-2:acct-id:batch-segment-job/
batchSegmentJobName"
}
```

バッチセグメントジョブの作成 (AWS SDKs)

バッチレコメンデーション用の入力データを準備します。 を完了すると、`CreateBatchSegmentJob` 操作を使用してバッチセグメントジョブを作成する準備が整います。次のコードは、バッチセグメントジョブを作成する方法を示しています。ジョブに名前を付け、使用するソリューションバージョンの Amazon リソースネーム (ARN) を指定し、Amazon Personalize IAM ロールの ARN を指定し、入力ファイルと出力場所への Amazon S3 パスを指定します。IAM サービスロールには、入力および出力の Amazon S3 バケットへの読み取りおよび書き込みアクセスがそれぞれ必要です。

出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。入力および出力の場所には、**`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`** および **`s3://<name of your S3 bucket>/<output folder name>/`** の構文を使用します。

numResults には、入力データの各行に Amazon Personalize に予測させたいユーザーの人数を指定します。デフォルトは 25 です。最大数は 500 万です。オプションで filterArn を指定してユーザーセグメントを絞り込みます。フィルタでプレースホルダーパラメータを使用する場合は、パラメータの値が入力 JSON に含まれていることを確認してください。詳細については、[バッチレコメンデーションとユーザーセグメントのフィルタリング (カスタムリソース)]を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize_rec = boto3.client(service_name='personalize')

personalize_rec.create_batch_segment_job (
    solutionVersionArn = "Solution version ARN",
    jobName = "Job name",
    numResults = 25,
    filterArn = "Filter ARN",
    roleArn = "IAM service role ARN",
    jobInput =
       {"s3DataSource": {"path": "s3://<name of your S3 bucket>/<folder name>/<input
 JSON file name>.json"}},
    jobOutput =
       {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder
 name>/"}}
)
```

SDK for Java 2.x

```
public static String createBatchSegmentJob(PersonalizeClient personalizeClient,
                                                        String solutionVersionArn,
                                                        String jobName,
                                                        String filterArn,
                                                        int numResults,
                                                        String
 s3InputDataSourcePath,
                                                        String
 s3DataDestinationPath,
                                                        String roleArn,
                                                        String explorationWeight,
                                                        String
 explorationItemAgeCutOff) {
```

```
 long waitInMilliseconds = 60 * 1000;
 String status;
 String batchSegmentJobArn;

 try {
     // Set up data input and output parameters.
     S3DataConfig inputSource = S3DataConfig.builder()
             .path(s3InputDataSourcePath)
             .build();
     S3DataConfig outputDestination = S3DataConfig.builder()
             .path(s3DataDestinationPath)
             .build();

     BatchSegmentJobInput jobInput = BatchSegmentJobInput.builder()
             .s3DataSource(inputSource)
             .build();
     BatchSegmentJobOutput jobOutputLocation = BatchSegmentJobOutput.builder()
             .s3DataDestination(outputDestination)
             .build();


     CreateBatchSegmentJobRequest createBatchSegmentJobRequest =
CreateBatchSegmentJobRequest.builder()
             .solutionVersionArn(solutionVersionArn)
             .filterArn(filterArn)
             .jobInput(jobInput)
             .jobOutput(jobOutputLocation)
             .jobName(jobName)
             .numResults(numResults)
             .roleArn(roleArn)
             .build();

     batchSegmentJobArn =
personalizeClient.createBatchSegmentJob(createBatchSegmentJobRequest)
             .batchSegmentJobArn();
     DescribeBatchSegmentJobRequest describeBatchSegmentJobRequest =
DescribeBatchSegmentJobRequest.builder()
             .batchSegmentJobArn(batchSegmentJobArn)
             .build();

     long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

     // wait until the batch segment job is complete.
     while (Instant.now().getEpochSecond() < maxTime) {
```

```
            BatchSegmentJob batchSegmentJob = personalizeClient
                    .describeBatchSegmentJob(describeBatchSegmentJobRequest)
                    .batchSegmentJob();

            status = batchSegmentJob.status();
            System.out.println("batch segment job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return batchSegmentJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: 'NAME',
  jobInput: {         /* required */
    s3DataSource: {   /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */'
```

```
    }
  },
  jobOutput: {         /* required */
    s3DataDestination: {   /* required */
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateBatchSegmentJobCommand(createBatchSegmentJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

バッチジョブの処理が完了するまでに時間がかかる場合があります。DescribeBatchSegmentJob を呼び出し、入力パラメータとして batchSegmentJobArn を渡すことで、ジョブのステータスを確認できます。を呼び出すことで、 AWS 環境内のすべての Amazon Personalize バッチセグメントジョブを一覧表示することもできますListBatchSegmentJobs。

バッチセグメントジョブの出力例

バッチセグメントジョブは、Amazon S3 バケットからバッチ入力データをインポートし、USER_SEGMENTATION レシピでトレーニングされたソリューションバージョンを使用してユーザーセグメントを生成し、Amazon S3 バケットにセグメントをエクスポートします。

次のセクションでは、レシピ別のバッチセグメントジョブについて、正しい形式の JSON 出力例を示します。

トピック

- Item-Affinity

- Item-Attribute-Affinity

Item-Affinity

次の例は、Item-Affinity レシピの出力 JSON ファイルの形式を示しています。

```
{"input": {"itemId": "105"}, "output": {"recommendedUsers": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedUsers": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedUsers": ["2", "442", "435"]}}
...
```

Item-Attribute-Affinity

次の例は、Item-Attribute-Affinity レシピの出力 JSON ファイルの形式を示しています。

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action\"", "output":
 {"recommendedUsers": ["25", "78", "108"]}}
{"itemAttributes": "ITEMS.genres = \"Adventure\"", "output": {"recommendedUsers":
 ["87", "31", "129"]}}
{"itemAttributes": "ITEMS.genres = \"Horror\" AND ITEMS.genres = \"Action\"", "output":
 {"recommendedUsers": ["8", "442", "435"]}}
...
```

レコメンデーションの関連性の維持

関連するレコメンデーションは、カタログの拡大に合わせて、アプリケーションのユーザーエンゲージメント、クリックスルー率、およびコンバージョン率を高めます。ユーザー向けの Amazon Personalize レコメンデーションの関連性を維持および改善するため、データとソリューションバージョンを最新の状態に保ちます。これにより、Amazon Personalize はユーザーの最新の行動から学習し、最新のアイテムをレコメンデーションに含めることができます。

トピック

データセットを最新の状態に保つ

カタログが大きくなってきたら、一括または個別のデータインポート操作で履歴データを更新します。履歴データのインポートの詳細については、「ステップ 2: データの準備とインポート」を参照してください。モデルのトレーニング後にインポートしたデータがレコメンデーションにどのように影響するかについては、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。

パーソナライズされたリアルタイムのレコメンデーションを提供するユースケースやレシピについては、アイテムインタラクションデータセットをユーザーの行動で最新の状態に保ちます。これを行うには、イベントトラッカーと PutEvents API オペレーションとのアイテムインタラクションを記録します。Amazon Personalize は、ユーザーがアプリケーションを操作する際のユーザーの最新のアクティビティに基づいて、レコメンデーションを更新します。リクエストパーソナライゼーションについては、「リアルタイムパーソナライゼーション」を参照してください。リアルタイムイベントの記録の詳細については、「イベントの記録」を参照してください。

ドメインレコメンダーの維持

Amazon Personalize は 7 日ごとにレコメンダーをサポートするモデルを自動的に再トレーニングします。これは、データセット内のデータ全体に基づいてまったく新しいモデルを作成する完全な再トレーニングです。トレーニングに使用した列を変更すると、Amazon Personalize はレコメンダーを裏付けるモデルの完全な再トレーニングを自動的に開始します。

- 「あなたにおすすめ」と「おすすめ」のユースケースについては、Amazon Personalize がレコメンダーを更新して、新しい商品をレコメンデーションとして考慮します。自動更新は、モデルがユーザーの行動から学習するような完全な再トレーニングではありません。代わりに、自動更新により、レコメンダーが次回の全面的な再トレーニングを行う前に、Amazon Personalize が新しい商品をレコメンデーションに掲載できるようになります。自動更新については、「自動更新」を参照してください。
- Trending Now ユースケースを使用する場合、Amazon Personalize は 2 時間ごとにインタラクションデータを自動的に評価し、トレンドアイテムを特定します。レコメンダーが再トレーニングするのを待つ必要はありません。

レコメンダーの再トレーニングが進行中でも、レコメンダーからレコメンデーションを受けることができます。再トレーニングが完了するまで、レコメンダーは以前の構成とモデルを使用します。更新を追跡するには、Amazon Personalize コンソールのレコメンダー詳細ページで最新レコメンダー更新のタイムスタンプを確認できます。または、`latestRecommenderUpdate` オペレーションから DescribeRecommender の詳細を表示することもできます。

カスタムソリューションの維持

デフォルトでは、すべての新しいソリューションは自動トレーニングを使用して 7 日ごとに新しいソリューションバージョンを作成します。トレーニングは、ソリューションを削除するまで続行されます。

ソリューションを作成するときは、自動トレーニングを使用してソリューションバージョンの作成を管理することをお勧めします。これにより、ソリューションの維持が容易になります。これにより、ソリューションが最新のデータから学習するために必要な手動トレーニングが削除されます。自動トレーニングを使用しない場合は、ソリューションが最新のデータから学習できるように、新しいソリューションバージョンを手動で作成する必要があります。自動トレーニングの設定の詳細については、「」を参照してください自動トレーニングの設定。

トレーニングの頻度は、ビジネス要件、使用するレシピ、データをインポートする頻度によって異なります。すべてのレシピについて、少なくとも毎週トレーニングすることをお勧めします。自動トレーニングでは、これがデフォルトのトレーニング頻度です。新しいアイテムやアクションを頻繁に追加する場合は、レシピに応じてトレーニング頻度を高くすることをお勧めします。

- User-Personalization-v2、User-Personalization、または Next-Best-Action を使用すると、ソリューションが自動的に更新され、新しいアイテムやアクションがレコメンデーションの対象として考慮されます。自動更新は、自動トレーニングとは異なります。自動更新では、まったく新しいソ

リューションバージョンは作成されず、モデルは最新のデータから学習しません。ソリューションを維持するには、トレーニング頻度を少なくとも週に 1 回にする必要があります。追加のガイドラインや要件など、自動更新の詳細については、「」を参照してください自動更新。

- Trending-Now を使用すると、Amazon Personalize は、設定可能な時間間隔で、インタラクションデータ内の最もトレンドの高い項目を自動的に識別します。Trending-Now は、バルクまたはストリーミングインタラクションデータを通じて、前回のトレーニング以降に追加されたアイテムをレコメンデーションできます。トレーニング頻度は少なくとも週に 1 回にする必要があります。詳細については、「Trending-Now レシピ」を参照してください。
- 自動更新または Trending-Now レシピでレシピを使用しない場合、Amazon Personalize は次のトレーニング後にのみ新しいアイテムをレコメンデーションと見なします。例えば、Similar-Items レシピを使用していて、毎日新しいアイテムを追加する場合は、その日にレコメンデーションに表示されるように、これらのアイテムの毎日のトレーニング頻度を使用する必要があります。

イベントの記録

イベントとは、ユーザーとカタログとのやり取りのことです。ユーザーが商品を購入したり、動画を視聴したりするなど、アイテムとのやり取り、またはアクションを実行するなどのアクションとのやり取りの場合があります。例えば、クレジットカードの申請、メンバーシッププログラムへの登録などです。

Amazon Personalize は、リアルタイムイベントデータのみ、履歴イベントデータのみ、または両方の組み合わせに基づいてレコメンデーションを作成できます。お客様がアクションレコメンデーションに反応する様子をリアルタイムで記録します。これにより、インタラクションデータが蓄積され、データが最新の状態に保たれます。また、Amazon Personalize にユーザーの現在の関心を伝えることができるため、レコメンデーションの関連性が高まります。

ドメインのユースケースやカスタムレシピがリアルタイムのパーソナライズをサポートしている場合、Amazon Personalize はイベントをリアルタイムで使用して、ユーザーの関心の高まりに応じてレコメンデーションを更新および調整します。

リアルタイムイベントの記録方法は、インポートするインタラクションデータの種類によって異なります。

- アイテムインタラクションでは、PutEvents API オペレーションを使用してリアルタイムイベントを記録します。Amazon Personalize は、このデータをデータセットグループ内の アイテムインタラクションデータセット に追加します。詳細については、「イベントの記録」を参照してください。
- アクションインタラクションでは、PutActionInteractions API オペレーションを使用してリアルタイムイベントを記録します。Amazon Personalize は、これを使用して、データセットグループのアクションインタラクションデータセットに新しいデータを追加します。PERSONALIZED_ACTIONS レシピのみがアクションインタラクションデータを使用します。詳細については、「アクションインタラクションイベントの記録」を参照してください。

トピック

- サードパーティーのイベント追跡サービス
- サンプル実装

リアルタイムイベントがレコメンデーションに与える影響

レシピがリアルタイムのパーソナライズをサポートしている場合、レコメンダーまたはカスタムキャンペーンを作成すると、Amazon Personalize はインポートから数秒以内に既存の商品またはアクションについて新しく記録されたイベントデータを使用します。リアルタイムパーソナライゼーションをサポートしているユースケースとレシピは次のとおりです。

- おすすめ (eコマース ユースケース)
- 上位のおすすめ (VIDEO_ON_DEMAND ユースケース)
- User-Personalization-v2 レシピ
- User-Personalization レシピ
- Personalized-Ranking-v2 レシピ
- Personalized-Ranking レシピ
- Next-Best-Action レシピ

Trending-Now レシピを使用すると、Amazon Personalize は設定可能な間隔で新しいイベントデータの項目を自動的に検討します。新しいソリューションバージョンを作成する必要はありません。詳細については、「Trending-Now レシピ」を参照してください。

イベント内のアイテム、アクション、またはユーザーが新しい場合、Amazon Personalize がデータをどのように使用するかは、ユースケースまたはレシピによって異なります。詳細については、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。

アイテムインタラクションイベントの記録

アイテムインタラクションイベントは、ユーザーとカタログ内のアイテムとのやり取りです。例えば、ユーザーが靴を購入したり、映画を見たりする場合です。

お客様におすすめの商品を見せながら、商品とのやり取りをリアルタイムで記録します。これにより、インタラクションデータが蓄積され、データが最新の状態に保たれます。また、Amazon Personalize にユーザーの現在の関心を伝えることができるため、レコメンデーションの関連性が高まります。

PutEvents API オペレーションを使用してアイテムインタラクションイベントを記録します。Amazon Personalize は、イベントデータをデータセットグループのアイテムインタラクションデータセットに追加します。まったく同じタイムスタンプとプロパティを持つ 2 つのイベントを記録した場合、Amazon Personalize はどちらかのイベントのみを保持します。 AWS SDKsを使用して、アイテムインタラクションイベントを記録できますAWS CLI。 AWS AWS Command Line Interface

Apache Kafka を使用している場合は、Amazon Personalize 用 Kafka コネクタを使用して、Amazon Personalize にアイテムインタラクションをリアルタイムでストリーミングできます。詳細については、personalize-kafka-connector の Github リポジトリにある「Amazon Personalize 用 Kafka コネクタ」を参照してください。

AWS Amplify には、ウェブクライアントアプリケーションからのアイテムインタラクションイベントを記録するための JavaScript ライブラリと、サーバーコードでイベントを記録するためのライブラリが含まれています。詳細については、「Amplify - 分析」を参照してください。

トピック

- アイテムインタラクションイベントの記録とモデルのトレーニングに関する要件
- アイテムインタラクションイベントトラッカーの作成
- PutEvents オペレーションの使用
- イベントメトリクスと属性レポート

アイテムインタラクションイベントの記録とモデルのトレーニングに関する要件

アイテムインタラクションイベントを記録するには、以下が必要です。

- `Item interactions` データセットを含むデータセットグループ。これは空の場合があります。開始 ガイドを完了した場合は、作成したものと同じデータセットグループおよびデータセットを使用できます。データセットグループとデータセットの作成については、「ステップ 2: データの準備とインポート」を参照してください。
- イベントトラッカー。
- PutEvents API オペレーションの呼び出し。
- AWS Lambda 関数を使用して PutEvents オペレーションを呼び出す場合、関数の実行ロールには、 `Resource`要素`*`のワイルドカードを使用して`personalize:PutEvents`アクションを実行するアクセス許可が必要です。

空のアイテムインタラクションデータセットから始めて、十分なデータを記録したら、新しく記録されたイベントのみを使用してモデルをトレーニングできます。すべてのユースケース (ドメインデータセットグループ) とレシピ (カスタムデータセットグループ) では、トレーニング前にインタラクションデータに以下が含まれている必要があります。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

アイテムインタラクションイベントトラッカーの作成

アイテムインタラクションイベントを記録する前に、イベントトラッカーを作成する必要があります。イベントトラッカーは、新しいイベントデータをデータセットグループのアイテムインタラクションデータセットに送信します。

イベントトラッカーは、Amazon Personalize コンソールまたは CreateEventTracker API 操作を使用して作成します。ターゲットのアイテムインタラクションデータセットを含むデータセットグループの Amazon リソースネーム (ARN) をパラメータとして渡します。Amazon Personalize コンソールを使用してイベントトラッカーを作成する手順については、「イベントトラッカーの作成 (コンソール)」を参照してください。

イベントトラッカーには追跡 ID が含まれており、 PutEventsオペレーションを使用するときにパラメータとして渡します。その後、Amazon Personalize は、イベントトラッカーで指定したデータセットグループのアイテムインタラクションデータセットに新しいイベントデータを追加します。

> **Note**
>
> データセットグループ用に作成できるアイテムインタラクションイベントトラッカーは 1 つだけです。

Python

```
import boto3
```

```
personalize = boto3.client('personalize')

response = personalize.create_event_tracker(
    name='MovieClickTracker',
    datasetGroupArn='arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieClickGroup'
)
print(response['eventTrackerArn'])
print(response['trackingId'])
```

イベントトラッカーの ARN と追跡 ID が表示されます。次に例を示します。

```
{
    "eventTrackerArn": "arn:aws:personalize:us-west-2:acct-id:event-tracker/
MovieClickTracker",
    "trackingId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

AWS CLI

```
aws personalize create-event-tracker \
    --name MovieClickTracker \
    --dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/
MovieClickGroup
```

イベントトラッカーの ARN と追跡 ID が表示されます。次に例を示します。

```
{
    "eventTrackerArn": "arn:aws:personalize:us-west-2:acct-id:event-tracker/
MovieClickTracker",
    "trackingId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

SDK for Java 2.x

```
public static String createEventTracker(PersonalizeClient personalizeClient,
                                        String eventTrackerName,
                                        String datasetGroupArn) {

    String eventTrackerId = null;
    String eventTrackerArn = null;
```

```
    long maxTime = 3 * 60 * 60;
    long waitInMilliseconds = 30 * 1000;
    String status;

    try {
        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
            personalizeClient.createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();

        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
            System.out.println("EventTracker status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return eventTrackerId;
    }
```

```
    catch (PersonalizeException e){
        System.out.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return eventTrackerId;
}
```

PutEvents オペレーションの使用

データセットグループのアイテムインタラクションデータセットとイベントトラッカーを作成したら、アイテムインタラクションイベントを記録する準備が整います。アイテムインタラクションイベントを記録するには、PutEvents API オペレーションを使用します。以下のセクションでは、1 つのイベントを記録する方法、イベント値データを使用して複数のイベントを記録する方法、およびインプレッションデータをイベントに含める方法を示します。

匿名ユーザーのイベントを録画する方法については、「匿名ユーザー向けのイベントの記録」を参照してください。

トピック

- 1 つのアイテムインタラクションイベントの記録
- 複数のアイテムインタラクションイベントをイベント値データと共に記録する
- インプレッションデータの記録

1 つのアイテムインタラクションイベントの記録

次の例は、1 つのアイテムインタラクションイベントを渡す PutEvents オペレーションを示しています。対応するスキーマが表示され、アイテムインタラクションデータセットから行の例が表示されます。

アプリケーションは、ユーザーが最初にウェブサイトにアクセスしたとき、またはアプリケーションを使用したときに、一意の sessionId を生成します。セッション中のすべてのイベントで同じ sessionId を使用する必要があります。Amazon Personalize は、sessionId を使用して、ユーザーがログインする前にイベントをそのユーザーに関連付けます (匿名)。詳細については、匿名ユーザー向けのイベントの記録を参照してください。

イベントリストは Event オブジェクトの配列です。各イベントについて eventType が必要ですが、この例では、eventType データはスキーマに含まれていないため、トレーニングでは使用されません。要件を満たすためにプレースホルダーの値を指定できます。

trackingId は、アイテムインタラクションイベントトラッカーの作成 で作成したイベントトラッカーから取得されます。Interactions の userId、itemId、sentAt パラメータは、対応する履歴データセットの USER_ID、ITEM_ID、TIMESTAMP フィールドにマップされます。詳細については、「 スキーマ」を参照してください。

対応するデータセット列

```
Dataset columns: USER_ID, ITEM_ID, TIMESTAMP
Item interactions dataset data: user123, item-xyz, 1543631760
```

コードサンプル

Python

```
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_events(
    trackingId = 'tracking_id',
    userId= 'USER_ID',
    sessionId = 'session_id',
    eventList = [{
        'sentAt': TIMESTAMP,
        'eventType': 'eventTypePlaceholder',
        'itemId': 'ITEM_ID'
        }]
)
```

AWS CLI

```
aws personalize-events put-events \
    --tracking-id tracking_id \
    --user-id USER_ID \
    --session-id session_id \
    --event-list '[{
        "sentAt": TIMESTAMP,
        "eventType": "eventTypePlaceholder",
        "itemId": "ITEM_ID"
      }]'
```

SDK for Java 2.x

```
public static void putEvents(PersonalizeEventsClient personalizeEventsClient,
                             String trackingId,
                             String sessionId,
                             String userId,
                             String itemId) {

    try {
        Event event = Event.builder()
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 *
 1000))
            .itemId(itemId)
            .eventType("typePlaceholder")
            .build();

        PutEventsRequest putEventsRequest = PutEventsRequest.builder()
            .trackingId(trackingId)
            .userId(userId)
            .sessionId(sessionId)
            .eventList(event)
            .build();

        int responseCode = personalizeEventsClient.putEvents(putEventsRequest)
            .sdkHttpResponse()
            .statusCode();
        System.out.println("Response code: " + responseCode);

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
}
```

この例を終了したら、必要なプロパティのみを使用してモデルのトレーニングに進みます。

複数のアイテムインタラクションイベントをイベント値データと共に記録する

次の例は、イベントタイプとイベント値が異なる複数のアイテムインタラクションイベントを記録する方法を示しています。

ソリューションを設定するときに、インタラクションデータセットに EVENT_TYPE フィールドと EVENT_VALUE フィールドが含まれている場合、トレーニングからレコードを除外するためのしき

い値として特定の値を設定できます。詳細については、「」を参照してくださいトレーニングに使用するアイテムインタラクションデータの選択

この例では、特定のレシピによってメタデータとして使用される追加のプロパティ numRatings の記録も示しています。

```
Dataset columns: USER_ID, ITEM_ID, TIMESTAMP, EVENT_TYPE, EVENT_VALUE, NUM_RATINGS
Item interactions dataset: user123, movie_xyz, 1543531139, rating, 5, 12
                      user321, choc-ghana, 1543531760, like, 4
                      user111, choc-fake, 1543557118, like, 3
```

Python

```
import boto3
import json

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_events(
    trackingId = 'tracking_id',
    userId= 'user555',
    sessionId = 'session1',
    eventList = [{
        'eventId': 'event1',
        'sentAt': 1553631760,
        'eventType': 'like',
        'properties': json.dumps({
            'itemId': 'choc-panama',
            'eventValue': 4,
            'numRatings': 0
            })
        }, {
        'eventId': 'event2',
        'sentAt': 1553631782,
        'eventType': 'rating',
        'properties': json.dumps({
            'itemId': 'movie_ten',
            'eventValue': 3,
            'numRatings': 13
            })
        }]
)
```

AWS CLI

```
aws personalize-events put-events \
    --tracking-id tracking_id \
    --user-id user555 \
    --session-id session1 \
    --event-list '[{
        "eventId": "event1",
        "sentAt": 1553631760,
        "eventType": "like",
        "properties": "{\"itemId\": \"choc-panama\", \"eventValue\": \"true\"}"
      }, {
        "eventId": "event2",
        "sentAt": 1553631782,
        "eventType": "rating",
        "properties": "{\"itemId\": \"movie_ten\", \"eventValue\": \"4\",
 \"numRatings\": \"13\"}"
      }]'
```

SDK for Java 2.x

```
public static void putMultipleEvents(PersonalizeEventsClient
 personalizeEventsClient,
                             String trackingId,
                             String sessionId,
                             String userId,
                             String event1Type,
                             Float event1Value,
                             String event1ItemId,
                             int event1NumRatings,
                             String event2Type,
                             Float event2Value,
                             String event2ItemId,
                             int event2NumRatings) {

    ArrayList<Event> eventList = new ArrayList<Event>();

    try {
        Event event1 = Event.builder()
            .eventType(event1Type)
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 *
 1000))
            .itemId(event1ItemId)
```

```
            .eventValue(event1Value)
            .properties("{\"numRatings\": "+ event1NumRatings +"}")
            .build();

        eventList.add(event1);

        Event event2 = Event.builder()
            .eventType(event2Type)
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 *
 1000))
            .itemId(event2ItemId)
            .eventValue(event2Value)
            .properties("{\"numRatings\": "+ event2NumRatings +"}")
            .build();

        eventList.add(event2);

        PutEventsRequest putEventsRequest = PutEventsRequest.builder()
            .trackingId(trackingId)
            .userId(userId)
            .sessionId(sessionId)
            .eventList(eventList)
            .build();

        int responseCode = personalizeEventsClient.putEvents(putEventsRequest)
            .sdkHttpResponse()
            .statusCode();

        System.out.println("Response code: " + responseCode);

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

> **Note**
>
> プロパティキーは、Interactions スキーマのフィールドと一致するキャメルケース名を使用します。例えば、フィールド「NUM_RATINGS」がインタラクションスキーマで定義されている場合、プロパティキーは numRatings である必要があります。

インプレッションデータの記録

User-Personalization レシピを使用するか、ドメインデータセットグループ内のデータセットのスキーマに IMPRESSIONS フィールドを追加すると、オペレーションで PutEventsインプレッションデータを記録できます。インプレッションは、ユーザーが特定のアイテムを操作した (例えば、クリックや視聴した) ときに表示されたアイテムのリストです。Amazon Personalize は、インプレッションデータを使用して探索をガイドします。レコメンデーションには、インタラクションデータや関連性が少ないアイテムが含まれます。Amazon Personalize がモデル化できる暗黙的および明示的なインプレッションについては、「インプレッションデータ」を参照してください。

> **Important**
>
> `PutEvents` リクエストで競合する暗黙的および明示的なインプレッションデータを提供する場合、Amazon Personalize は、デフォルトで明示的なインプレッションを使用します。

ユーザーに表示する Amazon Personalize のレコメンデーションをインプレッションデータとして記録するには、PutEvents リクエストに `recommendationId` を含めます。Amazon Personalize は、レコメンデーションデータに基づいて暗黙的なインプレッションを派生させます。

イベントのインプレッションデータを手動で記録するには、PutEvents コマンドの `impression` 入力パラメータにインプレッションをリストします。次のコードサンプルは、 SDK for Python (Boto3) または SDK for Java 2.x のいずれかを使用して `recommendationId`および `impression` をPutEvents オペレーションに含める方法を示しています。両方を含めると、Amazon Personalize は、デフォルトで明示的なインプレッションを使用します。

SDK for Python (Boto3)

```
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_events(
    trackingId = 'tracking_id',
    userId= 'userId',
    sessionId = 'sessionId',
    eventList = [{
        'eventId': 'event1',
        'eventType': 'rating',
        'sentAt': 1553631760,
```

```
        'itemId': 'item id',
        'recommendationId': 'recommendation id',
        'impression': ['itemId1', 'itemId2', 'itemId3']
        }]
)
```

SDK for Java 2.x

次の putEvents メソッドを使用して、インプレッションデータと recommendationId を使用してイベントを記録します。impressions パラメータには、itemIds のリストを として渡します ArrayList。

```
public static void putEvents(PersonalizeEventsClient personalizeEventsClient,
                             String trackingId,
                             String sessionId,
                             String userId,
                             String eventType,
                             Float eventValue,
                             String itemId,
                             ArrayList<String> impressions,
                             String recommendationId) {

    try {
        Event event = Event.builder()
            .eventType(eventType)
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 *
 1000))
            .itemId(itemId)
            .eventValue(eventValue)
            .impression(impressions)
            .recommendationId(recommendationId)
            .build();

        PutEventsRequest putEventsRequest = PutEventsRequest.builder()
            .trackingId(trackingId)
            .userId(userId)
            .sessionId(sessionId)
            .eventList(event)
            .build();

        int responseCode = personalizeEventsClient.putEvents(putEventsRequest)
            .sdkHttpResponse()
            .statusCode();
```

```
            System.out.println("Response code: " + responseCode);

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
    }
```

イベントメトリクスと属性レポート

Amazon Personalize に送信されるイベントのタイプと数をモニタリングするには、Amazon CloudWatch メトリクスを使用します。詳細については、「Amazon Personalize のモニタリング」を参照してください。

レコメンデーションの影響を示す CloudWatch レポートを生成するには、メトリクス属性を作成し、リアルタイムのレコメンデーションを使用したユーザーインタラクションを記録します。メトリクス属性の作成方法については、「レコメンデーションの影響の測定」を参照してください。

イベントごとに、ユーザーに表示したレコメンデーションのレコメンデーション ID を含めてください。または、サードパーティなどのイベントソースを含めてください。このデータをインポートして、さまざまなキャンペーン、レコメンダー、サードパーティを比較します。インポートできるイベント属性ソースは最大 100 個です。

- `recommendationId` を指定すると、Amazon Personalize はソースキャンペーンまたはレコメンダーを自動的に判断し、レポートの EVENT_ATTRIBUTION_SOURCE 列で特定します。
- 両方の属性を指定した場合、Amazon Personalize は `eventAttributionSource` のみを使用します。
- ソースを指定しない場合、Amazon Personalize はレポートでソース `SOURCE_NAME_UNDEFINED` にラベルを付けます。

次のコードは、 PutEvents オペレーションでイベント`eventAttributionSource`に を提供する方法を示しています。

```
response = personalize_events.put_events(
    trackingId = 'eventTrackerId',
    userId= 'userId',
    sessionId = 'sessionId123',
    eventList = [{
        'eventId': 'event1',
```

```
        'eventType': 'watch',
        'sentAt': '1667260945',
        'itemId': '123',
        'metricAttribution': {
            'eventAttributionSource': 'thirdPartyServiceXYZ'
        }
    }]
)
statusCode = response['ResponseMetadata']['HTTPStatusCode']
print(statusCode)
```

次のコードは、 PutEvents オペレーションでイベントrecommendationIdに を提供する方法を示しています。

```
response = personalize_events.put_events(
    trackingId = 'eventTrackerId',
    userId= 'userId',
    sessionId = 'sessionId123',
    eventList = [{
        'eventId': 'event1',
        'eventType': 'watch',
        'sentAt': '1667260945',
        'itemId': '123',
        'recommendationId': 'RID-12345678-1234-1234-1234-abcdefghijkl'
    }]
)
statusCode = response['ResponseMetadata']['HTTPStatusCode']
print(statusCode)
```

アクションインタラクションイベントの記録

アクションインタラクションイベントは、ユーザーとアクションの間のインタラクションです。例えば、ユーザーがメンバーシッププログラムに登録したり、クレジットカードを申請したりする場合です。

PERSONALIZED_ACTIONS カスタムレシピを使用する場合は、お客様がアクションレコメンデーションとやり取りするときに、リアルタイムのアクションインタラクションイベントを記録します。これにより、インタラクションデータが蓄積され、データが最新の状態に保たれます。また、Amazon Personalize にユーザーの現在の関心を伝えることができるため、レコメンデーションの関連性が高まります。PERSONALIZED_ACTIONS カスタムレシピのみがアクションインタラクションデータを使用します。

PutActionInteractions API オペレーションでアクションインタラクションイベントを記録します。Amazon Personalize は、これを使用して、データセットグループのアクションインタラクションデータセットに新しいデータを追加します。

アクションインタラクションイベントには、次のいずれかのイベントタイプ属性が必要です。

- 実行済み - ユーザーが推奨されるアクションを実行したときの実行済みイベントを記録します。
- 未実行 - ユーザーが閲覧後にアクションを実行しないという意図的な選択をした場合に、未実行イベントを記録します。例えば、ユーザーに対してアクションを表示したときに、ユーザーがいいえを選択した場合です。未実行イベントは、お客様がそのアクションに興味がないことを示している可能性があります。
- 閲覧済み - ユーザーがアクションを実行するかしないかの選択をする前にアクションを示したときに、閲覧済みイベントを記録します。Amazon Personalize は、閲覧イベントを使用してユーザーの関心を学習します。例えば、ユーザーが表示されたアクションを実行しなかった場合、このユーザーは今後、このアクションに興味を持たなくなる可能性があります。

AWS SDKsを使用して、リアルタイムイベントを記録できますAWS CLI。 AWS Command Line Interface まったく同じタイムスタンプとプロパティを持つ 2 つのイベントを記録した場合、Amazon Personalize はどちらかのイベントのみを保持します。

トピック

- アクションインタラクションイベントを記録するための要件
- アクションインタラクションイベントトラッカーの ID を検索する
- PutActionInteractions オペレーションの使用

アクションインタラクションイベントを記録するための要件

リアルタイムのアクションインタラクションイベントを記録するには、以下が必要です。

- `Action interactions dataset` を含むデータセットグループ。これは空の場合があります。データセットグループとデータセットの作成については、「ステップ 2: データの準備とインポート」を参照してください。
- イベントトラッカーの ID。この ID は PutActionInteractions オペレーションで指定します。アクションインタラクションデータセットを作成すると、Amazon Personalize は自動的にアクションインタラクションイベントトラッカーを作成します。詳細については、「アクションインタラクションイベントトラッカーの ID を検索する」を参照してください。

- PutActionInteractions オペレーションの呼び出し。

アクションインタラクションイベントトラッカーの ID を検索する

アクションインタラクションデータセットを作成すると、Amazon Personalize は自動的にアクションインタラクションイベントトラッカーを作成します。 PutActionInteractions API オペレーションでトラッカーの ID を指定します。Amazon Personalize は、これを使用して、データセットグループのアクションインタラクションデータセットに新しいデータを送信します。

イベントトラッカーの ID は、Amazon Personalize コンソールのアクションインタラクションデータセットの詳細ページで確認できます。また、 DescribeDataset API オペレーションを呼び出すことで ID を見つけることができます。次の Python コードは、アクションインタラクションデータセットの追跡 ID を出力します。

```
import boto3

personalize = boto3.client(service_name='personalize')

response = personalize.describe_dataset(
  datasetArn="Action interactions dataset ARN"
)

print(response['trackingId'])
```

PutActionInteractions オペレーションの使用

アクションインタラクションデータセットを作成したら、PutActionInteractions オペレーションでアクションインタラクションイベントを記録する準備が整います。以下のセクションでは、1 つのイベントを記録する方法と、イベント値データを使用して複数のイベントを記録する方法を示します。

トピック

- 1 つのアクションインタラクションイベントの記録
- 複数のアクションインタラクションイベントの記録

1 つのアクションインタラクションイベントの記録

次の例は、TAKEN イベントを渡す PutActionInteractions オペレーションを示しています。このイベントは、Amazon Personalize からのレコメンデーションをユーザーに表示し、ユーザーがクレジットカードの申請などのアクションを実行したときに記録できます。

は ActionInteraction オブジェクトの配列actionInteractionsです。trackingId これは、アクションインタラクションデータセットを作成したときに Amazon Personalize が作成したイベントトラッカーからのものです。詳細については、「アクションインタラクションイベントトラッカーの ID を検索する」を参照してください。

アプリケーションは、ユーザーが最初にウェブサイトにアクセスしたとき、またはアプリケーションを使用したときに、一意の sessionId を生成します。セッション中のすべてのイベントで同じ sessionId を使用する必要があります。Amazon Personalize は、sessionId を使用して、ユーザーがログインする前にイベントをそのユーザーに関連付けます (匿名)。詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。

userId、actionId、および sentAt パラメータは、アクションインタラクションデータセットの USER_ID、ACTION_ID、EVENT_TYPE、TIMESTAMP の各フィールドにマップされます。

対応するアクションインタラクションデータセット

```
USER_ID, ACTION_ID, TIMESTAMP, EVENT_TYPE
user123, action-xyz, 1543631760, TAKEN
```

コードサンプル

AWS CLI

```
aws personalize-events put-action-interactions \
--tracking-id 12345678-xxxx-xxxx-xxxx-xxxxxxxxxxxx \
--action-interactions '[{
  "userId": "user123",
  "sessionId": "abcdefg",
  "timestamp": 1543631760,
  "eventType": "TAKEN",
  "actionId": "action-xyz"}]'
```

SDK for Python (Boto3)

```
import boto3
```

```
personalize_events = boto3.client(service_name='personalize-events')

response = personalize_events.put_action_interactions(
  trackingId='12345678-xxxx-xxxx-xxxx-xxxxxxxxxxxx',
  actionInteractions=[{
    'userId': 'user123',
    'sessionId': 'abcdefg',
    'timestamp': 1543631760,
    'eventType': 'Taken',
    'actionId': 'action-xyz'
  }]
)
```

複数のアクションインタラクションイベントの記録

次のコードは、同じ sessionId を持つ同じユーザーの複数のアクションインタラクションイベントを記録する方法を示しています。

対応するアクションインタラクションデータセット

```
USER_ID, ACTION_ID, EVENT_TYPE, TIMESTAMP
user123, action123, Taken, 1543531139
user123, action345, Not Taken, 1543531139
```

AWS CLI

```
aws personalize-events put-action-interactions \
--tracking-id 6ddfe6b7-cd83-4dd4-b09d-4c35ecbacfe1 \
--action-interactions '[{
  "userId": "user123",
  "sessionId": "abcdefg",
  "timestamp": 1543531139,
  "eventType": "Taken",
  "actionId": "action123"
},
{
  "userId": "user123",
  "sessionId": "abcdefg",
  "timestamp": 1543531139,
  "eventType": "Not Taken",
```

```
  "actionId": "action345"}]'
```

SDK for Python (Boto3)

```
import boto3

personalize_events = boto3.client(service_name='personalize-events')

response = personalize_events.put_action_interactions(
  trackingId='12345678-xxxx-xxxx-xxxx-xxxxxxxxxxxx',
  actionInteractions=[{
    'userId': 'user123',
    'sessionId': 'abcdefg',
    'timestamp': 1697848587,
    'eventType': 'Taken',
    'actionId': 'action123'
  },
  {
    'userId': 'user123',
    'sessionId': 'abcdefg',
    'timestamp': 1697848622,
    'eventType': 'Not Taken',
    'actionId': 'action345'
  }]
)
```

匿名ユーザー向けのイベントの記録

> **Important**
>
> 1 人のユーザーに対して sessionId と userId を持つイベントを少なくとも 1つ記録しないと、Amazon Personalize は sessionId のみに追跡されたアクティビティをトレーニング時に使用しません。また、トレーニングが完了すると、sessionId に追跡されたアクティビティに基づくレコメンデーションは行われなくなります。

ユーザーがアカウントを作成する前に、アイテムインタラクションまたはアクションインタラクションイベントを記録できます。匿名ユーザーのイベントを記録して、ログイン前とログイン後のイベントを含む継続的なイベント履歴を構築できます。これにより、Amazon Personalize はユーザーに関

するより多くのインタラクションデータを提供し、より関連性の高いレコメンデーションを作成するのに役立ちます。

匿名ユーザー (ログインしていないユーザー) のイベントを記録するには、イベントごとに sessionId のみを指定します。アプリケーションは、ユーザーが最初にウェブサイトにアクセスしたとき、またはアプリケーションを使用したときに、一意の sessionId を生成します。セッション中のすべてのイベントで同じ sessionId を使用する必要があります。Amazon Personalize は、sessionId を使用して、ユーザーがログインする前にイベントをユーザーに関連付けます。

Amazon Personalize は、匿名ユーザーからのイベントを、userId に関連付けるまでトレーニングに使用しません。詳細については、「匿名ユーザー向けの継続的なイベント履歴の作成」を参照してください。

匿名ユーザーにリアルタイムのパーソナライゼーションを提供するには、 を GetRecommendationsまたは GetActionRecommendations リクエストuserIdの sessionIdとして指定します。

- PutEvents オペレーション および sessionId と userId を使用してアイテムインタラクションイベントを記録する方法を示すコードサンプルについては、「」を参照してください PutEvents オペレーションの使用。
- PutActionInteractions オペレーション および sessionId と userId を使用してアクションインタラクションイベントを記録する方法を示すコードサンプルについては、「」を参照してください PutActionInteractions オペレーションの使用。

匿名ユーザー向けの継続的なイベント履歴の作成

匿名ユーザーのイベント履歴を作成し、Amazon Personalize がトレーニング時にそのイベントを使用できるようにするには、sessionId と userId の両方を使用して少なくとも 1 つのイベントを記録します。そうすれば、userId のイベントをいくつでも記録できます。userId を提供し始めると、sessionId は変わる可能性があります。次回の完全再トレーニング時に、Amazon Personalize は userId を元のユーザー履歴に追跡された匿名ユーザー履歴と関連付けます。sessionId

再トレーニングが完了すると、レコメンデーションは、匿名イベントからの sessionId と追跡されたアクティビティと、userId に追跡されたイベントの両方に基づいて行われます。

> **Note**
>
> ユーザーがアカウントを作成せず、Amazon Personalize にトレーニング時にそのデータを使用させたい場合は、sessionId をイベント内 userId として使用できます。ただし、ユー

ザーが最終的にアカウントを作成した場合、そのユーザーの匿名ブラウジングで発生したイベントを新しい `userId` に関連付けることはできません。

サードパーティーのイベント追跡サービス

以下のカスタマーデータプラットフォーム (CDP) は、アプリケーションからイベントデータを収集して、Amazon Personalize に送信するのに役立ちます。

- Amplitude — Amplitude を使用してユーザーアクションを追跡し、ユーザーの行動を理解するのに役立ちます。Amplitude と Amazon Personalize の使用方法については、 AWS パートナーネットワーク (APN) のブログ記事「Amplitude と Amazon Personalize によるパーソナライゼーションの効果の測定」を参照してください。
- mParticle — mParticle を使用してアプリからイベントデータを収集できます。mParticle と Amazon Personalize を使用してパーソナライズされた製品レコメンデーションを実装する方法を示す例については、「CDP の力を機械学習に活用する方法:パート 2」を参照してください。
- セグメント — セグメントを使用してデータを Amazon Personalize に送信できます。セグメントと Amazon パーソナライズの統合についての詳細は、「Amazon Personalize デスティネーション」を参照してください。

サンプル実装

Amazon Personalize を使用して、イベントトラッカーと PutEventsオペレーションを使用してユーザーのリアルタイムの動作に対応する方法を示すサンプル Jupyter Notebook については、amazon-personalize-samples GitHub リポジトリの getting_started フォルダの「2.View_Campaign_And_Interactions.ipynb」を参照してください。

レコメンデーションを操作するユーザーからイベントをストリーミングする方法を示す例については、Amazon Personalize サンプル GitHub リポジトリの「streaming_events」を参照してください。

Amazon Personalize リソースとクライアントアプリケーションの間にリアルタイム APIs「リアルタイムパーソナライゼーション APIs」を参照してください。 AWS GitHub このプロジェクトには、以下の実装方法が含まれています。

- ユーザーコンテキストとユーザーイベントの収集
- レスポンスキャッシュ
- アイテムメタデータに基づくレコメンデーションのデコレーション

- A/B テスト
- API 認証

データセット内のトレーニングデータの管理

データセットにデータをインポートしたら、次の操作を実行できます。

- カタログの拡大に合わせてデータセット内のデータを更新します。これにより、Amazon Personalize のレコメンデーションの関連性を維持および改善できます。一括または個別のデータインポートオペレーションを使用して、より多くのデータをインポートできます。詳細については、「より多くのトレーニングデータをデータセットにインポートする」を参照してください。
- データセット内のトレーニングデータを分析します。データインサイトや列と行の統計からデータについて知ることができます。また、データを改善するためにどのようなアクションを取ればよいのかを知ることができます。これらのアクションは、モデルのトレーニングの要件などの Amazon Personalize のリソース要件を満たすのに役立つ場合や、レコメンデーションの改善につながる場合があります。詳細については、「データセット内のデータの品質と量を分析する」を参照してください。
- データを Amazon S3 バケットにエクスポートします。Amazon Personalize がレコメンデーションの生成に使用するデータを検証および検査し、以前にリアルタイムで記録したアイテムインタラクションイベントを表示したり、またはデータのオフライン分析を実行したりできます。詳細については、「データセット内のトレーニングデータを Amazon S3 にエクスポートする」を参照してください。
- Items and Users データセットの場合、データセットのスキーマを置き換えて、データの新しい列を追加できます。データセットの作成後にデータ構造が変更された場合は、データセットのスキーマを置き換えることができます。詳細については、「データセットのスキーマを置き換えて新しい列を追加する」を参照してください。
- データセット内のすべてのデータを削除できます。または、ユーザーとそのメタデータやインタラクションデータなどのデータをデータセットグループから削除することもできます。詳細については、「データ削除ジョブによるユーザーとそのデータの削除」および「データセットを削除してすべてのデータを削除する」を参照してください。

トピック

- データ削除ジョブによるユーザーとそのデータの削除
- データセットを削除してすべてのデータを削除する

リアルタイムイベントがレコメンデーションに与える影響

レコメンダーまたはカスタムソリューションバージョンを作成した後、新しいデータがリアルタイムのレコメンデーションにどのように影響するかは、そのタイプ、インポート方法、使用するドメインユースケースまたはカスタムレシピによって異なります。以下のセクションでは、次のトレーニングの前に、新しいデータがリアルタイムのレコメンデーションにどのように影響するかについて説明します。

トレーニングは、レコメンダーの毎週の自動トレーニングでも、自動または手動のソリューションバージョンの作成でもかまいません。User-Personalization を使用した手動トレーニングでは、`trainingMode`を に設定する必要がありますFULL。

新しいレコードがバッチレコメンデーションにどのように影響するかについては、「バッチレコメンデーションの取得」を参照してください。新しいレコードがバッチセグメントジョブにどのように影響するかについては、「ユーザーセグメントを取得する」を参照してください。

トピック

- 新しいインタラクション
- 新しいアイテム
- 新規のユーザー
- 新しいアクション

新しいインタラクション

新しいインタラクションは、最新のトレーニング後にインポートするアイテムまたはアクションのインタラクションです。

リアルタイムデータとバルクデータの両方について、インタラクションに新しいアイテムやアクションが含まれる場合、Amazon Personalize はトレーニングなしでレコメンデーションとしてそれを考慮することがあります。詳細については、新しいアイテムまたは新しいアクションを参照してください。

リアルタイムイベント

リアルタイムのパーソナライゼーションを特徴とするユースケースとレシピの場合、Amazon Personalize は、同じユーザーに対するレコメンデーションを生成するときに、ユーザーと既存のアイテムまたはアクション (最新のトレーニングで記録されているレコード) とのリアルタイムのインタラクションをすぐに使用します。リアルタイムのパーソナライゼーションについての詳細は、「リアルタイムパーソナライゼーション」を参照してください。

類似アイテムのレコメンデーションなど、リアルタイムのパーソナライゼーションを特徴付けないドメインユースケースやカスタムレシピの場合、モデルはトレーニング後にのみリアルタイムのインタラクションデータから学習します。

バルクインタラクション

バルクインタラクション の場合、増分データセットインポートジョブとフルデータセットインポートジョブの両方で、モデルは次のトレーニング後にのみバルクアイテムインタラクションまたはアクションインタラクションデータから学習します。リアルタイムのパーソナライゼーションのためのレコメンデーションの更新には、バルクデータは使用されません。

既存のバルクデータの更新の詳細については、「既存のバルクレコードの更新」を参照してください。

新しいアイテム

新しいアイテムは、最新のトレーニング後にインポートするアイテムです。アイテムデータセットのインタラクションデータまたはアイテムデータセットのアイテムメタデータのいずれかから取得できます。

新しいアイテムは、次のようにレコメンデーションの対象として考慮されます。

- 上位のおすすめとおすすめのドメインケースまたは User-Personalization-v2、User-Personalization、Next-Best-Action レシピの場合、Amazon Personalize は 2 時間ごとにモデルを自動的に更新します。更新のたびに、Amazon Personalize は新しいアイテムを探索の一環としてレコメンデーションと見なします。新しいアイテムを検討する際、Amazon Personalize はそのアイテムのメタデータをすべて考慮します。ただし、このデータは、アイテムのインタラクションを記録して新しいモデルをトレーニングした後でのみ、レコメンデーションに大きな影響を与えます。更新については、「自動更新」を参照してください。
- Trending Now ユースケースを使用する場合、Amazon Personalize は 2 時間ごとにインタラクションデータを自動的に評価し、トレンドアイテムを特定します。レコメンダーがトレーニングするのを待つ必要はありません。Trending-Now レシピを使用する場合、Amazon Personalize は設定可能な間隔ですべての新しい項目をトレーニングなしで自動的に考慮します。設定の間隔については、「Trending-Now レシピ」を参照してください。

- Trending-Now レシピを使用しない場合、またはユースケースやレシピが自動更新をサポートしていない場合、Amazon Personalize は次のトレーニング後にのみ新しいアイテムを考慮します。

新規のユーザー

新しいユーザーとは、最新のトレーニング後にインポートするユーザーです。ユーザーデータセットのインタラクションデータまたはユーザーメタデータのいずれかから取得できます。新規の匿名ユーザー (userId を持たないユーザー) については、`sessionId` を使用してイベントをレコードできます。また、Amazon Personalize はユーザーがログインする前にイベントをユーザーに関連付けます。詳細については、匿名ユーザー向けのイベントの記録を参照してください。

Amazon Personalize は、新規ユーザー向けのレコメンデーションを次のように生成します。

- Trending Now ドメインのユースケースまたは Trending-Now カスタムレシピを使用すると、新規ユーザーには最上位のトレンドアイテムのレコメンデーションがすぐに届きます。Popularity-Count レシピを使用すると、新規ユーザーには、インタラクションが最も多い商品のレコメンデーションがすぐに届きます。
- パーソナライズされたレコメンデーションをユーザーに提供するレシピやユースケースの場合、新規ユーザーへのレコメンデーションは既存ユーザーの初期の操作履歴に基づいて行われます。既存ユーザーが最初に操作したアイテムまたはアクションは、新規ユーザーにレコメンデーションされる可能性が高くなります。User-Personalization または Personalized-Ranking レシピでは、`recency_mask` を `true` に設定した場合、レコメンデーションにはインタラクションデータ内の最新の人気トレンドに基づくアイテムも含まれます。

次の設定を行うと、新規ユーザーへのレコメンデーションの関連性が高まります。

- インタラクションデータ — 新規ユーザーのレコメンデーション関連性を向上させる主な方法は、そのユーザーによるアイテムの操作からデータをインポートすることです。新しいインタラクションデータがレコメンデーションにどのように影響するかについては、「新しいインタラクション」を参照してください。
- ユーザーメタデータ — GENDER や MEMBERSHIP_STATUS などのユーザーメタデータをインポートすると、レコメンデーションを改善できます。メタデータがレコメンデーションに影響を与えるには、ドメインレコメンダーによる毎週の自動の再トレーニングが完了するまで待つ必要があります。または、新しいソリューションバージョンを手動で作成する必要があります。
- コンテキストメタデータ - ユースケースまたはレシピがコンテキストメタデータをサポートしていて、アイテムインタラクションデータセットにコンテキストデータ用のメタデータフィールドがあ

る場合は、レコメンデーションのリクエストでユーザーのコンテキストを指定できます。これには再トレーニングは必要ありません。詳細については、「コンテキストメタデータを使用したレコメンデーションの関連性の向上」を参照してください。

新しいアクション

新しいアクションは、最新のトレーニング以降にインポートするアクションです。アクションインタラクションデータから取得することも、アクションデータセット内のアクションから取得することもできます。

Next-Best-Action レシピを使用すると、Amazon Personalize は、2 時間ごとにソリューションバージョンを自動的に更新します。更新のたびに、Amazon Personalize は調査の一環としてレコメンデーションのために新しいアクションを考慮します。新しいアクションを検討する際、Amazon Personalize はそのアクションのすべてのメタデータを考慮します。ただし、このデータがレコメンデーションに与える影響は、そのアクションのアクションインタラクションを記録して完全に再トレーニングした後でのみ大きくなります。更新については、「自動更新」を参照してください。

より多くのトレーニングデータをデータセットにインポートする

カタログが大きくなるにつれて、追加のトレーニングデータをデータセットにインポートします。これにより、Amazon Personalize のレコメンデーションの関連性を維持および改善できます。一括または個別のデータインポートオペレーションを使用して、より多くのデータをインポートできます。

既存のデータセットを更新してデータ列を追加する場合は、データセットのスキーマを、列が追加された新しいスキーマに置き換えることができます。次に、新しいデータ列をインポートできます。詳細については、「データセットのスキーマを置き換えて新しい列を追加する」を参照してください。

トピック

- 個々のインポートオペレーションによるデータのインポート
- 既存のバルクレコードの更新

個々のインポートオペレーションによるデータのインポート

Amazon Personalize データセットにデータをインポートしたら、アイテムインタラクション、アクションインタラクション、ユーザー、アイテム、アクションなど、追加の個別レコードをインポートしてデータを更新できます。データを個別にインポートすると、カタログの拡大に合わせて Amazon Personalize のデータセットに小さなバッチのレコードを追加できます。

レコードを個別にインポートすると、Amazon Personalize は新しいレコードをデータセットに追加します。個々のアイテム、ユーザー、またはアクションを更新するには、ID は同じ、属性が変更されたレコードをインポートします。1 つのインポート操作で最大 10 個のレコードをインポートできます。

レコードを個別にインポートする方法についての詳細は、「個々のレコードのインポート」を参照してください。リアルタイムイベントの記録については、「」を参照してくださいイベントの記録。

既存のバルクレコードの更新

以前にデータセットのデータセットインポートジョブを作成した場合は、別のインポートジョブ を作成してバルクデータに追加または置き換えます。データセットのインポートジョブは、一括でインポートしたデータセット内の既存のデータを置き換えます。代わりに、ジョブのインポートモードを変更することで、新しいレコードを既存のデータに追加できます。

バルクデータを更新するためのガイドラインと要件は次のとおりです。

- データセットのインポートジョブでアイテムインタラクションデータセットまたはアクションインタラクションデータセットにデータを追加するには、少なくとも 1,000 件の新しいインタラクションレコードまたはアクションインタラクションレコードが必要です。
- レコメンダーをすでに作成しているか、キャンペーンを含むカスタムソリューションバージョンをデプロイしている場合、新しいバルクレコードがレコメンデーションにどのように影響するかは、使用するドメインのユースケースまたはレシピによって異なります。詳細については、「リアルタイムイベントがレコメンデーションに与える影響」を参照してください。
- 一括インポートの完了から 20 分以内に、Amazon Personalize は、データセットグループで作成したすべてのフィルターを、新しい一括データで更新します。この更新により、Amazon Personalize はユーザーのレコメンデーションをフィルタリングするときに最新のデータを使用できるようになります。

データセットのインポートジョブの作成の詳細については、「」を参照してくださいデータセットのインポートジョブを使用したバルクレコードのインポート。

データセット内のデータの品質と量を分析する

アイテムインタラクション、ユーザー、またはアイテムデータセットにデータをインポートしたら、Amazon Personalize コンソールを使用してデータを分析できます。データインサイトや列と行の統計からデータについて知ることができます。また、データを改善するためにどのようなアクションを取ればよいのかを知ることができます。これらのアクションは、モデルのトレーニングの要件な

どの Amazon Personalize のリソース要件を満たすのに役立つ場合や、レコメンデーションの改善につながる場合があります。

> **Important**
>
> Amazon Personalize コンソールを使用して、アクションインタラクションまたはアクションデータセットのデータを分析することはできません。

推奨される変更を加えたら、データを再度インポートして、問題が解決されたか、またはデータセットの統計が改善されたかを確認できます。データの更新については、「より多くのトレーニングデータをデータセットにインポートする」を参照してください。

インサイトが見当たらない場合は、データは Amazon Personalize データの期待に沿っているということです。ドメインデータセットグループまたはカスタムデータセットグループのデータを分析できます。

Amazon Personalize は、インサイトを生成して統計を計算する際、非匿名ユーザーからのすべての一括データおよびストリーミングデータを考慮します。匿名ユーザーからのイベントは、`userId` に関連付けるまで考慮されません。詳細については、「匿名ユーザー向けのイベントの記録」を参照してください。

トピック

- データ分析に必要なアクセス権限
- データインサイト
- データセットのインサイトと統計を表示する

データ分析に必要なアクセス権限

Amazon Personalize へのフルアクセスをユーザーに付与する場合、アクセス権限を変更する必要はありません。Amazon Personalize でタスクを実行するために必要なアクセス許可のみをユーザーに付与する場合、 AWS Identity and Access Management (IAM) ポリシーに次の追加のデータインサイトアクションを含める必要があります。

- パーソナライズ：CreateDataInsightsJob
- パーソナライズ：ListDataInsightsJob
- パーソナライズ：DescribeDataInsightsJob

- Personalize:GetDataInsight

データインサイト

Amazon Personalize で生成できる、考えられるデータインサイトは次のとおりです。

インサイト	アクション	関連するデータセット
インタラクションデータセットには X 個のインタラクションしかありません。モデルのトレーニングには、1,000 回以上のインタラクションが必要です。50,000 件以上のレコメンデーションを行います。	モデルをトレーニングする前に、Y 個の一意のインタラクションレコードを追加でインポートします。	アイテムインタラクション
インタラクションデータセットには、2 つ以上のインタラクションを持つ一意のユーザーが X 人しかいません。モデルのトレーニングには、このようなユーザーが少なくとも 25 人必要です。1,000 件以上のレコメンデーションを行います。	Y 人の追加ユーザーについて、それぞれ 2 つ以上のインタラクションレコードをインポートします。	アイテムインタラクション
アイテムデータセット内の X% の商品にはインタラクションデータセット内でインタラクションがないため、推奨されない可能性があります。	インタラクションデータをすべてインポートして、アイテムとインタラクションデータセットの間で ID が一致していないかどうかを確認してください。以下の商品とインタラクションデータセットのデータセットのデータセット統計をチェックして、必要な行数がインポートされていることを確認してください。ユースケースやレシピで探索を使用している場合は、探索設定を変更して、インタラクションデータを含まない商品をさらに推奨するようにします。	アイテムインタラクションとアイテム

インサイト	アクション	関連するデータセット
ユーザーデータセットのユーザーの X% は、インタラクションデータセットにインタラクションがありません。これらのユーザーには、人気商品のレコメンデーションが届きます。	インタラクションデータをすべてインポートして、ユーザーとインタラクションデータセットの間に ID が不一致がないかを確認してください。以下のユーザーとインタラクションデータセットのデータセット統計を確認して、必要な行数がインポートされていることを確認してください。追加のインタラクションをインポートして、より多くのユーザーがインタラクションデータを取得できるようにします。	アイテムインタラクションとユーザー
<Users or Items or Interactions> データセットには値が欠落した業が X% あります。これはレコメンデーションに悪影響を及ぼす可能性があります。必須フィールドとオプションフィールドはすべて 70% 以上入力することを推奨します。	完全なレコードを追加インポートするか、不完全な行がないデータを再度インポートするか、欠損値を代替データ (数値列の平均値やカテゴリ列の最も一般的な値など) に置き換えてデータを再度インポートします。	すべて
<datasetType > データセットの次の列 (複数可) は、完了率が 70% 未満です: <ColumnName、 ColumnName...>。このデータがトレーニングに含まれている場合、レコメンデーションに悪影響を及ぼす可能性があります。NULL 値を許容する列は 70% 以上入力することを推奨します。	完全なレコードを追加インポートするか、不完全な行がないデータを再度インポートするか、欠損値を代替データ (数値列の平均値やカテゴリ列の最も一般的な値など) に置き換えてデータを再度インポートします。	すべて

インサイト	アクション	関連するデータセット
次の (数値) 列 (複数可) には外れ値があります: <ColumnName，ColumnName...>。外れ値は必ずしも問題とは限りませんが、レコメンデーションに悪影響を及ぼすことがあります。	以下の列統計を使用して、これらの列の最小値と最大値が期待どおりかどうかを確認してください。これらの値が想定外の場合は、これらの列のデータに誤りがないかを確認し、データ収集とデータ処理に問題がないかを確認します。	すべて
次の列 (複数可) には、<ColumnName、ColumnName...> の 1000 を超えるカテゴリがあります。このデータがトレーニングに含まれている場合、<ColumnName，ColumnName...> というレコメンデーションに悪影響を及ぼす可能性があります。	カテゴリデータをチェックして、スペルの違いによるカテゴリの重複などの問題がないかを確認します。誤りがあれば解決し、データをもう一度インポートします。	すべて
次のテキストメタデータ列 (複数可) は 85% 未満で、モデルトレーニングでは使用されません: <ColumnName，ColumnName...>。	追加の行をインポートするか、これらの列のテキストデータを含む行を再度インポートします。	項目
インタラクションデータセットには 10 種類以上の一意のイベントタイプがあるため、モデルのトレーニングが失敗する可能性があります。	イベントタイプ列をチェックして、スペルの違いによるイベントタイプの重複などの不正確な点がないかを確認します。不要なイベントタイプを削除して、データをもう一度インポートします。	アイテムインタラクション
インタラクションデータセットのタイムスタンプはすべてのレコードで同じです。USER_SEGMENTATION レシピを使用し、すべてのレコードのタイムスタンプが同じ場合、モデルのトレーニングは失敗します。	データにタイムスタンプの問題がないか確認し、重複するタイムスタンプを一意のタイムスタンプに置き換えます。	アイテムインタラクション

データセットのインサイトと統計を表示する

Amazon Personalize データセット内のデータに関するインサイトと統計を表示するには、Amazon Personalize コンソールでデータセットに移動し、[分析を実行] を選択します。

インサイトと統計を表示するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。
3. [ナビゲーション] ペインの [データセット] から、[データ分析] を選択します。
4. 右上の [分析を実行] を選択します。Amazon Personalize がデータの分析を開始します。これには最大 15 分かかることがあります。成功すると、結果がこのページに表示されます。
5. インサイトで以下を使用して表示されるインサイトを絞り込みます。
   - 特定の言語を含むインサイトを検索するには、「インサイトを探す」に条件を入力します。テキストを入力すると、リストが更新され、インサイトまたは推奨されたアクションとまったく同じ文字列を含むインサイトのみが表示されます。
   - インサイトをデータセットタイプでフィルタリングするには、[すべてのデータセット] を特定のデータセットタイプに変更します。リストが更新され、このデータセットに関連するインサイトのみが含まれるようになります。
6. データセットのデータセット統計を表示するには、次の手順を実行します。
   - インタラクションデータセットの行、一意のユーザー、一意の商品の数など、データセットに関する一般的な詳細と統計情報を表示するには、データセットのセクションを展開します。
   - 列の詳細な統計情報を表示するには、データセットセクションを展開し、[列レベルの統計] を選択して、列のラジオボタンを選択します。
7. データ内の問題を修正して再度インポートし、別の分析を実行して検証します。データのインポートの詳細については、「より多くのトレーニングデータをデータセットにインポートする」を参照してください。

データセット内のトレーニングデータを Amazon S3 にエクスポートする

データを Amazon Personalize データセットにインポートした後、そのデータを Amazon S3 バケットにエクスポートできます。Amazon Personalize がレコメンデーションの生成に使用するデータを検証および検査し、以前にリアルタイムで記録したアイテムインタラクションイベントを表示したり、またはデータのオフライン分析を実行したりできます。

一括でインポートしたデータ (Amazon Personalize のデータセットのインポートジョブを使用してインポートされたもの) のみをエクスポートするか、個別にインポートしたデータ (コンソール、または `PutEvents`、`PutUsers`、もしくは `PutItems` 操作を使用してインポートされた履歴およびリアルタイムのレコード) のみをエクスポートするか、その両方を実行するかを選択できます。

> **Note**
>
> Action インタラクションデータセットまたは Actions データセットのデータをエクスポートすることはできません。

すべてのフィールドと完全に一致するレコードの場合、Amazon Personalize はレコードを 1 つだけエクスポートします。2 つのレコードの ID が同じでも 1 つ以上のフィールドが異なる場合、Amazon Personalize は、エクスポートするデータに応じてレコードを含めたり削除したりします。

- 一括データと増分データの両方をエクスポートする場合、Amazon Personalize は同じ ID の最新のアイテムのみをエクスポートし (アイテムデータセットのエクスポートの場合)、同じ ID のユーザーのみをエクスポートします (ユーザーデータセットのエクスポートの場合)。アイテムインタラクションデータセットの場合、Amazon Personalize はすべてのアイテムインタラクションデータをエクスポートします。
- 増分データのみをエクスポートする場合、Amazon Personalize は、同じ ID のアイテムやユーザーを含め、個別にインポートしたすべてのアイテム、ユーザー、またはアイテムインタラクションデータをエクスポートします。すべてのフィールドに完全に一致するレコードのみが除外されます。
- バルクデータのみをエクスポートする場合、Amazon Personalize は、同じ ID のアイテムやユーザーを含め、一括でインポートしたすべてのアイテム、ユーザー、またはアイテムインタラクションデータを含めます。すべてのフィールドに完全に一致するレコードのみが除外されます。

データセットをエクスポートするには、データセットのエクスポートジョブを作成します。データセットのエクスポートジョブは、データセット内のレコードを Amazon S3 バケット内の 1 つ以上の CSV ファイルに出力するレコードエクスポートツールです。出力 CSV ファイルには、データセットのスキーマのフィールドと一致する列名を持つヘッダー行が含まれています。

トピック

データセットのエクスポートジョブの許可要件

データセットをエクスポートするには、Amazon Personalize は、Amazon S3 バケットにファイルを追加するための許可を必要とします。許可を付与するには、バケットで `PutObject` および `ListBucket` アクションを使用するための許可をロールに付与する、Amazon Personalize のサービスロールに新しい AWS Identity and Access Management (IAM) ポリシーをアタッチします。また、`PutObject` および `ListBucket` アクションを使用するための許可を Amazon Personalize のプリンシパルに付与する、出力 Amazon S3 バケットにバケットポリシーをアタッチします。

暗号化に AWS Key Management Service（ AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize の IAM サービスロールに、 キーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

データセットをエクスポートするためのサービスロールのポリシー

次のポリシー例は、`PutObject` および `ListBucket` アクションを使用するための許可を Amazon Personalize のサービスロールに付与します。`bucket-name` を出力バケットの名前に置き換えます。IAM のサービスロールにポリシーをアタッチする方法については、「Amazon Personalize サービスロールに対する、Amazon S3 ポリシーのアタッチ」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

データセットをエクスポートするための Amazon S3 バケットポリシー

次のポリシー例は、Amazon S3 バケットに対する PutObject および ListBucket アクションを使用するための許可を Amazon Personalize に付与します。bucket-name をバケットの名前に置き換えます。Amazon S3 バケットポリシーをバケットに追加する方法については、Amazon Simple Storage Service ユーザーガイドのAmazon S3 コンソールを使用したバケットポリシーの追加」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "personalize.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

データセットエクスポートジョブの作成

データセットのエクスポートジョブは、Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用して作成できます。

データセットのエクスポートジョブの作成 (コンソール)

データをデータセットにインポートし、出力 Amazon S3 バケットを作成したら、分析のためにデータをバケットにエクスポートできます。Amazon Personalize コンソールを使用してデータセットをエクスポートするには、データセットのエクスポートジョブを作成します。Amazon S3 バケットの作成の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの作成」を参照してください。

データセットをエクスポートする前に、Amazon Personalize のサービスロールが出力 Amazon S3 バケットにアクセスして書き込むことができることを確認してください。データセットのエクスポートジョブの許可要件 を参照してください。

データセットのエクスポートジョブを作成するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開きます。
2. ナビゲーションペインで、[Dataset groups] (データセットグループ) を選択します。
3. [Dataset groups] (データセットグループ) のページで、データセットグループを選択します。
4. ナビゲーションペインで、[Datasets] (データセット) を選択します。
5. Amazon S3 バケットにエクスポートするデータセットを選択します。
6. [Dataset export jobs] (データセットのエクスポートジョブ) で、[Create dataset export job] (データセットのエクスポートジョブを作成) を選択します。
7. [Dataset export job details] (データセットのエクスポートジョブの詳細) の [Dataset export job name] (データセットのエクスポートジョブ名) に、エクスポートジョブの名前を入力します。
8. [IAM サービスロール]で、「Amazon Personalize 向けの IAM ロールの作成」で作成した Amazon Personalize のサービスロールを選択します。
9. [Amazon S3 data output path] (Amazon S3 データ出力パス) で、宛先 Amazon S3 バケットを入力します。次の構文を使用します。

   ```
   s3://<name of your S3 bucket>/<folder path>
   ```

10. 暗号化 AWS KMS に を使用している場合は、KMS キー ARN にキーの Amazon リソースネーム (ARN) を入力します AWS KMS 。

11. [Export data type] (データタイプをエクスポート) で、最初にデータをインポートした方法に基づいて、エクスポートするデータのタイプを選択します。

    - データセットのインポートジョブを使用して一括でインポートしたデータのみをエクスポートするには、[Bulk] (一括) を選択します。
    - コンソールまたはPutEvents、PutUsers、または PutItems 操作を使用して個別にインポートしたデータのみをエクスポートするには、[増分] を選択します。
    - データセット内のすべてのデータをエクスポートするには、[Both] (両方) を選択します。
12. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。
13. [Create dataset export job] (データセットのエクスポートジョブを作成) を選択します。

    [Dataset overview] (データセットの概要) のページの [Dataset export jobs] (データセットのエクスポートジョブ) で、ジョブが [Export job status] (エクスポートジョブのステータス) で一覧表示されます。ステータスが ACTIVE になると、データセットのエクスポートジョブが完了します。その後、出力 Amazon S3 バケットからデータをダウンロードできます。Amazon S3 バケットからオブジェクトをダウンロードする方法については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのダウンロード」を参照してください。

データセットのエクスポートジョブの作成 (AWS CLI)

データをデータセットにインポートし、出力 Amazon S3 バケットを作成したら、分析のためにデータセットをバケットにエクスポートできます。を使用してデータセットをエクスポートするには AWS CLI、 create-dataset-export-job AWS CLI コマンドを使用してデータセットのエクスポートジョブを作成します。Amazon S3 バケットの作成の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの作成」を参照してください。

データセットをエクスポートする前に、Amazon Personalize のサービスロールが出力 Amazon S3 バケットにアクセスして書き込むことができることを確認してください。データセットのエクスポートジョブの許可要件 を参照してください。

次に create-dataset-export-job AWS CLI コマンドの例を示します。ジョブに名前を付け、dataset arn をエクスポートするデータセットの Amazon リソースネーム (ARN) に、role ARN を「Amazon Personalize 向けの IAM ロールの作成」で作成した Amazon Personalize のサービスロールの ARN に、それぞれ置き換えます。でs3DataDestination、 にはkmsKeyArnオプションで AWS KMS キーの ARN を指定し、 には出力 Amazon S3 バケットへのパスpathを指定します。

ingestion-mode で、次のオプションからエクスポートするデータを指定します。

- データセットのインポートジョブを使用して一括でインポートしたデータのみをエクスポートするように BULK を指定します。
- コンソールPUT、、、または PutItemsオペレーションを使用して個別にインポートしたデータのみをエクスポートするにはPutEvents PutUsers、 を指定します。
- データセット内のすべてのデータをエクスポートするように ALL を指定します。

詳細については、「CreateDatasetExportJob」を参照してください。

```
aws personalize create-dataset-export-job \
  --job-name job name \
  --dataset-arn dataset ARN \
  --job-output "{\"s3DataDestination\":{\"kmsKeyArn\":\"kms key ARN\",\"path\":
\"s3://bucket-name/folder-name/\"}}" \
  --role-arn role ARN \
  --ingestion-mode PUT
```

データセットのエクスポートジョブ ARN が表示されます。

```
{
  "datasetExportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-export-job/
DatasetExportJobName"
}
```

DescribeDatasetExportJob 操作を使用してステータスを確認します。

```
aws personalize describe-dataset-export-job \
  --dataset-export-job-arn dataset export job ARN
```

データセットエクスポートジョブの作成 (AWS SDKs)

データをデータセットにインポートし、出力 Amazon S3 バケットを作成したら、分析のためにデータセットをバケットにエクスポートできます。 AWS SDKsCreateDatasetExportJobオペレーションを使用してデータセットのエクスポートジョブを作成します。Amazon S3 バケットの作成の詳細については、Amazon Simple Storage Service ユーザーガイドの「バケットの作成」を参照してください。

次のコードは、SDK for Python (Boto3) または SDK for Java 2.x SDK を使用してデータセットのエクスポートジョブを作成する方法を示しています。

データセットをエクスポートする前に、Amazon Personalize のサービスロールが出力 Amazon S3 バケットにアクセスして書き込むことができることを確認してください。データセットのエクスポートジョブの許可要件 を参照してください。

SDK for Python (Boto3)

次の create_dataset_export_job を使用して、データセット内のデータを Amazon S3 バケットにエクスポートします。ジョブに名前を付け、dataset arn をエクスポートするデータセットの Amazon リソースネーム (ARN) に、role ARN を「Amazon Personalize 向けの IAM ロールの作成」で作成した Amazon Personalize のサービスロールの ARN に、それぞれ置き換えます。でs3DataDestination、 にはkmsKeyArnオプションで AWS KMS キーの ARN を指定し、 には出力 Amazon S3 バケットへのパスpathを指定します。

ingestionMode で、次のオプションからエクスポートするデータを指定します。

- データセットのインポートジョブを使用して一括でインポートしたデータのみをエクスポートするように BULK を指定します。
- コンソールPUT、、、または PutItemsオペレーションを使用して個別にインポートしたデータのみをエクスポートするにはPutEvents PutUsers、 を指定します。
- データセット内のすべてのデータをエクスポートするように ALL を指定します。

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_export_job(
    jobName = 'job name',
    datasetArn = 'dataset ARN',
    jobOutput = {
      "s3DataDestination": {
        "kmsKeyArn": "kms key ARN",
        "path": "s3://bucket-name/folder-name/"
      }
    },
    roleArn = 'role ARN',
    ingestionMode = 'PUT'
)
```

```
dsej_arn = response['datasetExportJobArn']

print ('Dataset Export Job arn: ' + dsej_arn)

description = personalize.describe_dataset_export_job(
    datasetExportJobArn = dsej_arn)['datasetExportJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['datasetExportJobArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

次の createDatasetExportJob メソッドを使用して、データセットのエクスポートジョブを作成します。パラメータとして PersonalizeClient、、エクスポートジョブの名前、エクスポートするデータセットの ARN、取り込みモード、出力 Amazon S3 バケットのパス、 AWS KMS キーの ARN を渡します。

ingestionMode は、次のいずれかのオプションになります。

- データセットのインポートジョブを使用して一括でインポートしたデータのみをエクスポートするように IngestionMode.BULK を使用します。
- コンソールまたは 、、または PutItemsオペレーションを使用して個別にインポートしたデータのみをエクスポートIngestionMode.PUTするにはPutEvents PutUsers、 を使用します。
- データセット内のすべてのデータをエクスポートするように IngestionMode.ALL を使用します。

```
public static void createDatasetExportJob(PersonalizeClient personalizeClient,
                                           String jobName,
                                           String datasetArn,
                                           IngestionMode ingestionMode,
                                           String roleArn,
                                           String s3BucketPath,
                                           String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {
        S3DataConfig exportS3DataConfig = S3DataConfig.builder()
```

```
            .path(s3BucketPath)
            .kmsKeyArn(kmsKeyArn)
            .build();

        DatasetExportJobOutput jobOutput = DatasetExportJobOutput.builder()
            .s3DataDestination(exportS3DataConfig)
            .build();

        CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
            .datasetArn(datasetArn)
            .ingestionMode(ingestionMode)
            .jobOutput(jobOutput)
            .roleArn(roleArn)
            .build();

        String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

        DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
            .datasetExportJobArn(datasetExportJobArn)
            .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetExportJob datasetExportJob =
personalizeClient.describeDatasetExportJob(describeDatasetExportJobRequest)
                .datasetExportJob();

            status = datasetExportJob.status();
            System.out.println("Export job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
```

```
        }
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

データセットのスキーマを置き換えて新しい列を追加する

アイテムまたはユーザーデータセットを作成したら、そのスキーマを新規または既存のスキーマに置き換えることができます。データセットの作成後にデータ構造が変更された場合は、データセットのスキーマを置き換えることができます。例えば、Amazon Personalize にトレーニング中に考慮してほしいアイテムメタデータの新しい列が存在する場合があります。あるいは、レコメンデーションをフィルタリングする場合にのみ使用するデータ列を追加したい場合もあります。

データセットのスキーマを置き換えるときは、以前のスキーマのフィールドをすべて保持しなければならず、データ型や属性を変更することはできません。データセットのスキーマを置き換えると、Amazon Personalize は既存のレコメンダーやカスタムソリューションのトレーニングから新しい列を自動的に除外します。その他のガイドラインと要件については、「ガイドラインと要件」を参照してください。

データセットのスキーマは、Amazon Personalize コンソール、 AWS Command Line Interface（AWS CLI）、および AWS SDKsに置き換えることができます。

トピック

- ガイドラインと要件
- データセットのスキーマの置き換え (コンソール)
- データセットのスキーマの置き換え (AWS CLI)
- データセットのスキーマの置き換え (AWS SDKs)

ガイドラインと要件

データセットのスキーマを置き換える前に、次のガイドラインと要件に注意してください。

- アイテムインタラクションデータセット、アクションインタラクションデータセット、またはアクションデータセットのスキーマを置き換えることはできません。
- 置換スキーマには新しいフィールドを追加できますが、すべてのフィールドは以前のスキーマに残しておく必要があります。また、データ型や属性は変更できません。例えば、以前のスキーマに力

テゴリ文字列データ用の MEMBERSHIP_STATUS フィールドが含まれている場合、使用する新しいスキーマには、これらの属性とデータ型を含む MEMBERSHIP_STATUS フィールドを含める必要があります。

- 現在のスキーマに名前を変更したいフィールドがある場合や、そのデータ型や属性を変更したい場合は、新しい名前を付けてタイプや属性を変更した新しいフィールドを追加できます。次に、新しいフィールドをトレーニングに含め、古いフィールドを除外します。新しいフィールドは null データをサポートする必要があります。古いフィールドが NULL データをサポートしていなかった場合は、データをインポートするときに、プレースホルダーデータを使用してインポートがスキーマと一致することを確認できます。レコメンダーが使用する列の設定については、「レコメンダーの更新」を参照してください。ソリューションによって使用される列の設定については、「トレーニング時に使用する列の設定」を参照してください。
- 新しいフィールドは null データをサポートする必要があります。フィールドに NULL タイプを追加する方法については、「スキーマのデータ型」を参照してください。
- データセットのスキーマを置き換えると、Amazon Personalize は既存のレコメンダーやカスタムソリューションのトレーニングから新しい列を自動的に除外します。変更したデータセットを使用するには、以下のアクションが必要です。
  - トレーニングで新しい列を使用するには、新しいスキーマに合うデータをインポートします。次に、レコメンダーを更新して新しい列を使用するか、新しいカスタムソリューションを作成して、トレーニング時に使用する列を設定します。

    レコメンダーが使用する列の更新については、「レコメンダーの更新」を参照してください。ソリューションによって使用される列の設定については、「トレーニング時に使用する列の設定」を参照してください。
  - フィルタリング時にのみ列を使用するには、新しいスキーマと一致するデータをインポートし、新しいデータを使用するフィルターを作成して、そのフィルターをレコメンデーションリクエストに適用します。レコメンダーを更新したり、カスタムリソースを作成または更新したりする必要はありません。

データセットのスキーマの置き換え (コンソール)

データセットのスキーマを Amazon Personalize コンソールに置き換えるには、変更するデータセットを選択し、新しいスキーマに置き換えるか、既存のスキーマを使用するかを選択します。

データセットのスキーマを置き換えるには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。
3. ナビゲーションペインで、データセット を選択し、変更するデータセットのラジオボタンを選択します。
4. [アクション] を選択し、[スキーマの置換] を選択します。
5. [スキーマの詳細] で、新しいスキーマと置き換えるか、以前に作成したスキーマに置き換えるかを選択します。
6. 使用する新しいスキーマを指定します。以下を選択した場合:
   - 新しいスキーマに置き換え、次にスキーマに名前を付け、[スキーマ定義] で JSON スキーマに変更を加えます。
   - 以前に作成したスキーマを使用する。次に [以前に作成したスキーマ] では、使用するスキーマを選択します。対象となるスキーマのみが表示されます。スキーマ要件については、「ガイドラインと要件」を参照してください。
7. [置換] を選択します。データセットがアクティブになると、新しいスキーマに沿ったデータのインポートを開始できます。詳細については、「ステップ 2: データの準備とインポート」を参照してください。

データセットのスキーマの置き換え (AWS CLI)

データセットのスキーマを に置き換えるには AWS CLI、 `update-dataset` コマンドを使用して、更新するデータセットの Amazon リソースネーム (ARN) と、使用する新しいスキーマの ARN を指定します。アイテムインタラクションデータセット、アクションインタラクションデータセット、またはアクションデータセットのスキーマを更新することはできません。

次のコードは、 AWS CLIでデータセットのスキーマを更新する方法を示しています。データセットのスキーマを新しいものに置き換えるには、まず `create-schema` コマンドを使用します。そして、次のコードを使用して現在のスキーマを新しいスキーマに置き換えます。を使用してスキーマを作成する方法については、 AWS CLI「」を参照してくださいデータセットとスキーマの作成 (AWS CLI)。データセットとスキーマの要件については、「 スキーマ」を参照してください。

```
aws personalize update-dataset \
--dataset-arn Dataset ARN \
```

```
--schema-arn New schema ARN
```

データセットがアクティブになると、新しいスキーマに沿ったデータのインポートを開始できます。詳細については、「ステップ 2: データの準備とインポート」を参照してください。データセットの最新の更新については、DescribeDataset オペレーションを使用できます。

データセットのスキーマの置き換え (AWS SDKs)

データセットのスキーマを AWS SDKsオペレーションを使用します。 UpdateDataset更新するデータセットの Amazon リソースネーム (ARN) と使用する新しいスキーマを指定します。アイテムインタラクションデータセット、アクションインタラクションデータセット、またはアクションデータセットのスキーマを更新することはできません。

次のコードは、データセットのスキーマを SDK for Python (Boto3) で置き換える方法を示しています。データセットのスキーマを新しいスキーマに置き換えるには、まず CreateSchema オペレーションを使用します。おして、次のコードを使用して現在のスキーマを新しいスキーマに置き換えます。 AWS SDKs「」を参照してくださいデータセットとスキーマ (AWS SDK) の作成。データセットとスキーマの要件についての詳細は、「 スキーマ」を参照してください。

```
import boto3

personalize = boto3.client('personalize')

update_dataset_response = personalize.update_dataset(
    datasetArn = 'dataset_arn',
    schemaArn = 'new_schema_arn'
)

print(update_dataset_response)
```

データセットがアクティブになると、新しいスキーマに沿ったデータのインポートを開始できます。詳細については、「ステップ 2: データの準備とインポート」を参照してください。データセットの最新の更新については、DescribeDataset オペレーションを使用できます。

データ削除ジョブによるユーザーとそのデータの削除

データをインポートしたら、メタデータやインタラクションデータを含むユーザーとそのデータをデータセットグループから削除できます。コンプライアンスプログラムの一環としてユーザーデータ

を削除したり、ユーザー削除リクエストに対処したり、ユーザーベースの変更に応じてデータを最新の状態に保つことができます。

ユーザーを削除すると、Amazon Personalize はデータに関するトレーニングを行わず、ユーザーセグメントの生成時にユーザーを考慮しなくなります。

データセットグループの Amazon Personalize データセットおよびモデル内のユーザーへの参照を削除するには、次の手順を実行します。

1. USER_ID 列で削除するユーザーの userIdsを一覧表示する CSV ファイルを作成します。
2. CSV ファイルを Amazon S3 バケットにアップロードします。Amazon Personalize サービスロールには、このバケットへのアクセス許可が必要です。
3. データ削除ジョブを作成します。データ削除ジョブは、データセットグループ内のモデルとデータセットからユーザーとそのデータを削除するバッチジョブです。

トピック

ガイドラインと要件

ユーザーを削除するためのガイドラインと要件は次のとおりです。

- データ削除ジョブを作成する前に、トレーニングジョブ、バッチジョブ、一括または個別のインポートオペレーションなど、データセットを使用するジョブが進行中でないことを確認してください。また、データ削除ジョブの進行中は、このようなジョブを作成しないようにしてください。トレーニングやインポートが発生した場合、ユーザーのデータがモデルから削除されることは保証できないため、追加のデータ削除ジョブを作成することをお勧めします。
- データ削除ジョブでは、Amazon Personalize 以外のユーザーへの参照は削除されません。例えば、Amazon S3 バケットのバッチレコメンデーションから userId を削除することはありません。Amazon S3 これらのレコードは手動で削除する必要があります。
- ステータスが「保留中」のデータセットグループに対して、最大 5 つの削除ジョブを設定できます。
- データ削除入力ファイルの最大合計サイズは 50 MB です。削除ジョブを作成するときと同じ入力ファイルを再利用できます。

- 各データ削除ジョブは、データセットグループ 内のユーザーとそのインタラクションデータを削除します。すべてのデータセットグループのデータを削除するには、データセットグループごとにデータ削除ジョブを作成する必要があります。
- ジョブを作成した後、データセットやモデルからユーザーのデータを削除するまでに最大 1 日かかる場合があります。
- ジョブが完了したら、必ずカスタムリソースを更新してください。必ず新しいソリューションバージョンを作成し、必要に応じてキャンペーンを更新してください。自動トレーニングを使用する場合でも、新しいソリューションバージョンを手動で作成できます。
- Amazon Personalize サービスロールには、削除するユーザーのリストを含む Amazon S3 バケットへのアクセス許可が必要です。バケット`GetObject`とそのコンテンツには および アクセス`ListBucket`許可が必要です。これらのアクセス許可はデータのインポートと同じです。アクセス許可の付与とポリシーの例については、「」を参照してくださいAmazon Personalize に対する、Amazon S3 リソースへのアクセスの付与。
- 削除するユーザーの userIds のリストを保存する Amazon S3 バケットで独自の AWS Key Management Service キーを使用することはできません。
- アイテムがアイテムインタラクションデータセットにのみ表示され、削除しようとしているユーザーのみがこのアイテムとインタラクションしている場合、このアイテムはレコメンデーションに表示されなくなります。

削除するユーザーのリストの準備

Amazon Personalize からユーザーを削除する前に、CSV ファイルで削除するユーザーのリストを準備し、Amazon S3 にアップロードする必要があります。

削除してアップロードするユーザーのリストを準備するには

1. 削除するユーザーの userIdsを一覧表示する CSV ファイルを作成します。以下は、CSV ファイルのフォーマット方法を示しています。

```
USER_ID
abc
2a
5basc
ab35
123f
a55d
0v22
```

```
441fa
efg
```

2. CSV ファイルを Amazon Simple Storage Service (Amazon S3) バケットにアップロードします。Amazon S3 へのファイルのアップロードの詳細については、Amazon Simple Storage Service ユーザーガイドの「ドラッグアンドドロップを使用したファイルとフォルダのアップロード」を参照してください。
3. Amazon Personalize にバケットと CSV ファイルへのアクセスを許可します。Amazon Personalize には、バケットとそのコンテンツに対して GetObjectおよび ListBucket アクションを実行するアクセス許可が必要です。これらのアクセス許可はデータのインポートと同じです。アクセス許可の付与とポリシーの例については、「」を参照してくださいAmazon Personalize に対する、Amazon S3 リソースへのアクセスの付与。

データ削除ジョブの作成

を完了すると削除するユーザーのリストの準備、データ削除ジョブを持つユーザーを削除する準備が整います。

データ削除ジョブは、データセットグループ内のモデルとデータセットからユーザーとそのデータを削除するバッチジョブです。ユーザーを削除すると、Amazon Personalize はデータに関するトレーニングを行わず、ユーザーセグメントの生成時にユーザーを考慮しなくなります。

データ削除ジョブを作成するときは、削除するユーザーリストの Amazon S3 の場所を指定します。

- データが 1 つのファイルにある場合は、Amazon S3 の場所に対して次の構文を使用します。

  s3://<name of your S3 bucket>/<folder path>/<CSV filename>.csv
- CSV ファイルが Amazon S3 バケット内のフォルダにある場合は、フォルダへのパスを指定できます。データ削除ジョブでは、Amazon Personalize は フォルダと任意のサブフォルダにファイル拡張子を持つすべての.csvファイルを使用します。他のタイプのファイルは無視されます。フォルダ名の後に / を付けて次の構文を使用します。

  s3://<name of your S3 bucket>/<folder path>/

使用するロールには、Amazon S3 バケットとそのコンテンツに対して GetObjectおよび ListBucket アクションを実行するアクセス許可が必要です。アクセス許可の付与とポリシーの例については、「」を参照してくださいAmazon Personalize に対する、Amazon S3 リソースへのアクセスの付与。

データ削除ジョブは、Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用して作成できます。

データ削除ジョブの作成 (コンソール)

Amazon Personalize コンソールでユーザーを削除するには、名前、IAM サービスロール、および Amazon S3 のデータの場所を使用してデータ削除ジョブを作成します。

レコードを削除するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループ] のページで、データセットグループを選択します。データセットグループの 概要 が表示されます。
3. ナビゲーションペインで、[データセット] を選択します。
4. データ削除ジョブ で、ジョブの作成 を選択します。
5. ジョブの詳細 で、ジョブに名前を付けます。
6. S3 入力ソース の S3 ロケーション で、削除するユーザーの userIds Amazon S3 の場所を指定します。このファイルは で準備しました削除するユーザーのリストの準備。
7. [IAM ロール] で、新しいロールを作成するか、既存のロールを使用するかを選択します。Amazon Personalize のロールを作成し、このロールに Amazon S3 バケットへのアクセス権を付与するための前提条件を満たしている場合は、既存のサービスロールを使用するを選択し、 で作成したロールを指定しますAmazon Personalize 向けの IAM ロールの作成。

   使用するロールには、Amazon S3 バケットとそのコンテンツに対して `GetObject`および `ListBucket` アクションを実行するアクセス許可が必要です。これらのアクセス許可はデータのインポートと同じです。アクセス許可の付与とポリシーの例については、「」を参照してくださいAmazon Personalize に対する、Amazon S3 リソースへのアクセスの付与。
8. [タグ] には、オプションで任意のタグを追加します。Amazon Personalize リソースのタグ付けについての詳細は、「Amazon Personalize リソースのタグ付け」を参照してください。
9. [ジョブの作成] を選択します。ジョブが開始され、詳細ページが表示されます。

   ジョブを作成した後、データセットやモデルからユーザーのデータを削除するのに約 1 日かかります。ジョブが完了するまで、Amazon Personalize はトレーニング時に引き続きデータを使用します。また、ユーザーはユーザーセグメントに表示される場合があります。

ステータスが COMPLETED と表示されると、データの削除は完了です。何らかの理由でジョブが失敗した場合は、別のデータ削除ジョブを作成することをお勧めします。ジョブが完了したら、必ずカスタムリソースを更新してください。必ず新しいソリューションバージョンを作成し、必要に応じてキャンペーンを更新してください。自動トレーニングを使用する場合でも、新しいソリューションバージョンを手動で作成できます。

データ削除ジョブの作成 (AWS CLI)

でユーザーを削除するには AWS CLI、 `create-data-deletion-job` コマンドを使用します。このコマンドは CreateDataDeletion API オペレーションを使用します。次のコードは、データ削除ジョブを作成する方法を示しています。コードを使用するには、コードを更新して、ジョブ名、 で作成した IAM ロールAmazon Personalize 向けの IAM ロールの作成、およびデータの Amazon S3 の場所を指定します。このファイルは で準備しました削除するユーザーのリストの準備。

```
aws personalize create-data-deletion-job \
--job-name deletion job name \
--dataset-group-arn dataset group ARN \
--data-source dataLocation=s3://bucketname/filename.csv \
--role-arn roleArn
```

ジョブを作成した後、データセットやモデルからユーザーのデータを削除するのに約 1 日かかります。ジョブが完了するまで、Amazon Personalize はトレーニング時に引き続きデータを使用します。また、ユーザーはユーザーセグメントに表示される場合があります。

ステータスが COMPLETED になると、ジョブは完了です。`describe-data-deletion-job` コマンドを使用してステータスを確認し、データ削除ジョブ ARN を指定します。API オペレーションの詳細については、「」を参照してくださいDescribeDataDeletionJob。作成時刻でソートされたデータ削除ジョブの履歴を表示するには、 ListDataDeletionJobs API オペレーションを使用します。

何らかの理由でジョブが失敗した場合は、別のデータ削除ジョブを作成することをお勧めします。ジョブが完了したら、必ずカスタムリソースを更新してください。必ず新しいソリューションバージョンを作成し、必要に応じてキャンペーンを更新してください。自動トレーニングを使用する場合でも、新しいソリューションバージョンを手動で作成できます。

データ削除ジョブの作成 (AWS SDKs)

AWS SDKsオペレーションを使用します。 CreateDataDeletionJob次のコードは、データ削除ジョブを作成する方法を示しています。コードを使用するには、コードを更新して、ジョブ名、 で作成し

た IAM ロールAmazon Personalize 向けの IAM ロールの作成、およびデータの Amazon S3 の場所を指定します。このファイルは で準備しました削除するユーザーのリストの準備。

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_data_deletion_job(
    jobName = 'Deletion job name',
    datasetGroupArn = 'Dataset Group ARN',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn'
)

deletion_job_arn = response['dataDeletionJobArn']

print ('Deletion Job arn: ' + deletion_job_arn)

description = personalize.describe_data_deletion_job(
    dataDeletionJobArn = deletion_job_arn)['dataDeletionJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['dataDeletionJobArn'])
print('Status: ' + description['status'])
```

ジョブを作成した後、データセットやモデルからユーザーのデータを削除するのに約 1 日かかります。ジョブが完了するまで、Amazon Personalize はトレーニング時に引き続きデータを使用します。また、ユーザーはユーザーセグメントに表示される場合があります。

ステータスが COMPLETED になると、ジョブは完了です。DescribeDataDeletionJob オペレーションを使用してステータスを確認し、データ削除ジョブ ARN を指定します。作成時刻でソートされたデータ削除ジョブの履歴を表示するには、 ListDataDeletionJobs API オペレーションを使用します。

何らかの理由でジョブが失敗した場合は、別のデータ削除ジョブを作成することをお勧めします。ジョブが完了したら、必ずカスタムリソースを更新してください。必ず新しいソリューションバージョンを作成し、必要に応じてキャンペーンを更新してください。自動トレーニングを使用する場合でも、新しいソリューションバージョンを手動で作成できます。

データセットを削除してすべてのデータを削除する

データセット内のすべてのデータを削除するには、データセットを削除します。データセットのインポートジョブまたはソリューションバージョンが CREATE PENDING または IN PROGRESS 状態にある場合、データセットを削除することはできません。User-Personalization-v2、User-Personalization、Next-Best-Action レシピ、または Top Picks for you and Recommended for you ユースケースを使用する場合、データセットを削除すると、関連するソリューションバージョンまたはレコメンダーの自動更新が停止します。

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用してデータセットを削除できます。

トピック

データセットの削除 (コンソール)

Amazon Personalize コンソールでデータセットを削除するには、データセットの詳細ページに移動して [削除] を選択します。

データセットを削除するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開きます。
2. ナビゲーションペインで、[Dataset groups] (データセットグループ) を選択します。
3. [Dataset groups] (データセットグループ) のページで、データセットグループを選択します。
4. ナビゲーションペインで、[Datasets] (データセット) を選択します。
5. データセットを選択して、詳細ページを開きます。
6. データセットの詳細ページで、データセットの削除と削除の確認を選択します。

データセットの削除 (AWS CLI)

次のコードは、 AWS CLI および DeleteDatasetオペレーションを使用してデータセットを削除する方法を示しています。

```
aws personalize delete-dataset --dataset-arn dataset-arn
```

データセットの削除 (AWS SDK)

次のコードは、 AWS SDKs と DeleteDatasetオペレーションを使用してデータセットを削除する方法を示しています。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.delete_dataset(
    datasetArn = 'dataset ARN'
)
```

SDK for Java 2.x

```
public static void deleteDataset(PersonalizeClient personalizeClient,
                                 String datasetArn) {

    try {
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                .datasetArn(datasetArn)
                .build();

        int responseCode =
 personalizeClient.deleteDataset(deleteRequest).sdkHttpResponse().statusCode();
        System.out.println(responseCode);
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

レコメンデーションとユーザーセグメントのフィルタリング

ドメインレコメンダーまたはカスタムキャンペーンでレコメンデーションを取得する場合、カスタム基準に基づいて結果をフィルタリングできます。例えば、ユーザーがすでに購入した商品はレコメンドしたくないとか、特定の年齢層向けのアイテムだけを勧めたいことがあるかもしれません。

同様に、USER_SEGMENTATION レシピでは、特定のタイプのユーザーをユーザーセグメントに含めたくない場合があります。結果をフィルタリングすることにより、ユーザーに推奨されるアイテムまたはユーザーセグメントに含まれるユーザーを制御できます。

Amazon Personalize コンソール、 (AWS CLI) 、および SDK を使用して、フィルターを作成、編集、 AWS Command Line Interface 削除、適用できます。 AWS SDKs

- リアルタイムのレコメンデーションについては、、、または GetPersonalizedRanking オペレーションを呼び出すときに、フィルターを適用し GetRecommendations GetActionRecommendations、フィルターパラメータ値を指定します。コンソールでキャンペーンまたはレコメンダーからレコメンデーションを取得するときにもフィルターを適用することができます。

  パーソナライズされたアイテムあるいは関連アイテムレシピまたはユースケースでリアルタイムのアイテムレコメンデーションを取得するときには、リクエストにプロモーションを指定できます。プロモーションでは、フィルターを使用して設定可能なおすすめアイテムのサブセットに適用される追加のビジネスルールを定義します。詳細については、レコメンデーション内のアイテムのプロモーションを参照してください。
- バッチワークフローでは、入力 JSON に任意のフィルターパラメータ値を含めます。次に、バッチ推論ジョブまたはバッチセグメントジョブを作成するときに、フィルターの Amazon リソースネーム (ARN) を指定します。詳細については、バッチレコメンデーションとユーザーセグメントのフィルタリング (カスタムリソース)を参照してください。

新しいレコードのフィルタ更新

PutEvents または PutActionInteractions オペレーションでインポートしたデータの場合、Amazon Personalize はデータセットグループ内のすべてのフィルターをインポートから数秒以内に新しいデータで更新します。例えば、フィルターがレコメンデーションから購入したアイテムを削除し、PutEvents オペレーションを使用してユーザーの購入イベントを記録すると、このアイテムはイベントを記録してから数秒以内にこのユーザーの今後のレコメンデーションから削除されます。

一括で、または個別にインポートされたその他すべてのデータについて、Amazon Personalize は、最後のインポートから 20 分以内に、データセットグループ内のすべてのフィルターを新しいデータで更新します。

トピック

フィルタ式

フィルターを設定するには、適切にフォーマットされたフィルター式を使用する必要があります。フィルター式は、論理演算子、キーワード、および値とともに、`dataset.field` フォーマットのデータセットとフィールド識別子で構成されます。値については、固定値を指定するか、プレースホルダーパラメータを追加して、レコメンデーションを取得する際にフィルター基準を設定できます。

フィルター式を使用して、次のデータセットのデータに基づいて、レコメンデーションからアイテム、ユーザー、またはアクションをフィルタリングできます。

- アイテムインタラクション : フィルター式を使用して、インタラクションデータに基づいてアイテムまたはユーザーを含めたり除外したりできます。例えば、ユーザーがクリックした項目を除外したり (アイテムレコメンデーションの場合) 、評価された項目を持つユーザーのみを含めたり (Item-Affinity レシピの場合) できます。すべてのレシピタイプで、イベントタイプに基づいてのみフィルタリングできます。コンテキストメタデータなど、他のインタラクションメタデータに基づいてフィルタリングすることはできません。でアイテムインタラクションフィルターを使用することはできませんItem-Attribute-Affinity レシピ。

  Amazon Personalize は、イベントタイプごとに、ユーザー 1 人あたり最大 100 件の最新のインタラクションを考慮します。これは調整可能なクォータです。Service Quotas コンソールを使用してクォータの増加をリクエストできます。
- アクションインタラクション : フィルター式を使用して、イベントタイプに基づいてユーザーが操作したアクションを含めたり除外したりします。例えば、ユーザーが既に実行したアクションを除外できます。他のアクションインタラクションメタデータに基づいてフィルタリングすることはできません。

Amazon Personalize は、イベントタイプごとに、ユーザー 1 人あたり最大 300 件の最新のアクションインタラクションを考慮します。これは調整可能なクォータです。Service Quotas コンソールを使用してクォータの増加をリクエストできます。

- Items: フィルター式を使用して、特定のアイテム条件に基づいてアイテムを含めたり除外したりします。フィルターを使用して、商品の説明などの構造化されていないテキストのアイテムメタデータに基づいてアイテムを含めたり、除外したりすることはできません。Similar-Items レシピや More Like X ドメインユースケースなど、ドメインのユースケースやカスタムレシピが関連アイテムのレコメンデーションを生成する場合、レコメンデーションリクエストで指定したアイテムのプロパティに基づいて、フィルター式を使用してアイテムを含めたり除外したりできます。
- Users: アイテムおよびアクションレコメンデーションの場合、Users データセットがあれば、`CurrentUser` に基づいてアイテムやアクションを除外したり含めたりできます。パーソナライズされたレコメンデーション、人気アイテム、およびアクションレコメンデーションの場合、これは、レコメンデーションを受けるユーザーです。関連アイテムの場合、これはレコメンデーションリクエストで指定できるオプションのユーザーです。

  ユーザーセグメントについては、フィルター式を使用して、`Users.MEMBERSHIP_STATUS` などの属性に基づいてユーザーをユーザーセグメントに含めたり、ユーザーセグメントから除外したりできます。
- Actions: フィルター式を使用して、特定のアクション条件に基づいて、アクションを含めたり除外したりします。Amazon Personalize は、`Action expiration timestamp` および `Repeat frequency` データに基づくアクションを自動的に除外します。このデータに基づいてフィルタリングする追加のカスタムフィルターを作成することはできません。

フィルター式の要素の詳細なリストについては、「フィルター式の要素」を参照してください。フィルター式の例については、「フィルター式の例」を参照してください。

トピック

- ガイドラインと要件
- フィルター式の構造と要素
- フィルター式の例

ガイドラインと要件

フィルター式を作成するときは、以下のガイドラインと要件に注意してください。

- フィルターを使用して、商品の説明などの構造化されていないテキストのアイテムメタデータに基づいてアイテムを含めたり、除外したりすることはできません。
- アイテムまたはアクションインタラクションデータに基づいてフィルタリングする場合は、イベントタイプに基づいてのみフィルタリングできます。コンテキストメタデータなど、他のインタラクションメタデータに基づいてフィルタリングすることはできません。
- Amazon Personalize は、イベントタイプが一致する場合にのみ大文字と小文字の区別を無視します。
- アイテムインタラクションおよびアイテムデータセットを 1 つの式で使用することはできません。Interactions データセット、次に Items データセット (またはその逆) でフィルタリングするフィルターを作成するには、2 つ以上の式を連鎖する必要があります。詳細については、「複数の式の組み合わせ」を参照してください。
- アイテムインタラクションおよび Action データセットを 1 つの式で使用することはできません。アイテムインタラクションデータセットによってフィルタリングし、次に Action データセットでフィルタリングする (またはその逆の) フィルターを作成するには、2 つ以上の式を連鎖する必要があります。詳細については、「複数の式の組み合わせ」を参照してください。
- でアイテムインタラクションフィルターを使用することはできませんItem-Attribute-Affinity レシピ。
- スキーマでブール型の値を使用してフィルタリングするフィルター式を作成することはできません。ブール値に基づいてフィルタリングするには、String 型のフィールドを持つスキーマを使用し、データの `"True"` および `"False"` の値を使用します。または、int もしくは long 型と `0` と `1` の値を使用できます。
- 1 つの条件式でも、複数の式を連結した場合でも、1 つのフィルターで使用できる個別のデータセット項目の最大数は 5 です。データセットグループ内のすべてのフィルターで使用できる個別のデータセットフィールドの最大数は 10 です。
- フィルターを CurrentItem 要素で適用できるのは、ドメインのユースケースまたはカスタムレシピが Similar-Items レシピや More Like X ドメインのユースケースなどの関連アイテムのレコメンデーションを生成する場合のみです。
- NOT_IN 演算子を使用するフィルター式では、プレースホルダーパラメーターは使用できません。代わりに IN 演算子を使用し、その逆の Action を使用してください。例えば、Exclude の代わりに Include を使用してください (またはその逆)。
- `Action expiration timestamp` および `Repeat frequency` データに基づいてフィルタリングするフィルターを作成することはできません。Amazon Personalize は、このデータに基づくアクションレコメンデーションを自動的にフィルタリングします。

フィルター式の構造と要素

このセクションには、フィルター式の構造と要素に関する情報が含まれています。

トピック

フィルター式の構造

フィルター式の一般的な構造は次のとおりです。

```
EXCLUDE/INCLUDE ItemID/ActionID/UserID WHERE dataset type.field IN/NOT IN (value/
parameter)
```

手動でフィルター式を作成するか、コンソールの[Expression builder] (式ビルダー) を使用して式の構文と構造に関するサポートを利用できます。

フィルター式の要素

次の要素を使用して、フィルター式を作成します。

INCLUDE または EXCLUDE

フィルター基準を満たすアイテムのみにレコメンデーションを制限するために INCLUDE を使用するか、またはフィルター基準を満たす、すべてのアイテムを削除するために EXCLUDE を使用します。

ItemID/ActionID/UserID

これらの要素のいずれかを INCLUDE または EXCLUDE 要素の後で使用します。使用する要素は、アイテム (アイテムレコメンデーション)、アクション (アクションレコメンデーション)、またはユーザー (ユーザーセグメント) のどれをフィルタリングするかによって異なります。

WHERE

アイテム、アクション、またはユーザーの条件を確認するには、WHERE を使用します。WHERE 要素は、ItemID、ActionID、または UserID の後で使用する必要があります。

AND/OR

同じフィルター式内で複数の条件を連鎖させるには、AND または OR を使用します。AND または OR を使用して組み合わせられた条件は、最初の条件で使用されたデータセットのフィールドにのみ影響を及ぼすことができます。

Dataset.field

dataset.field 形式でレコメンデーションをフィルタリングするデータセットとメタデータフィールドを指定します。例えば、Items データセットの genres フィールドに基づいてアイテムレコメンデーションをフィルタリングするには、フィルター式で Items.genres を使用します。

IF 条件

IF 条件は、式の最後で 1 回だけ、CurrentUser の条件をチェックするためにのみ使用してください。ただし、AND を使用して IF 条件を拡張できます。

CurrentUser.attribute

レコメンデーションを取得しようとしているユーザーに基づいてアイテムのレコメンデーションをフィルタリングするには、IF 条件でのみ、CurrentUser を使用してユーザーフィールドを指定します。例えば CurrentUser.AGE です。

CurrentItem.attribute

関連アイテムのレシピとユースケースにのみには、CurrentItem.attribute を使用して、関連商品のレコメンデーションのリクエストで指定した商品の属性に基づいて商品を絞り込むことができます。例えば、CurrentItem.GENRE、CurrentItem.PRICE などです。

フィルターを CurrentItem 要素で適用できるのは、ドメインのユースケースまたはカスタムレシピが Similar-Items レシピや More Like X ドメインのユースケースなどの関連アイテムのレコメンデーションを生成する場合のみです。CurrentItem 要素を含むフィルターを初めて作成する場合、フィルタの作成には数分かかることがあります。暗号化 AWS KMS に を使用する場合、フィルターの作成には最大 15 分かかることがあります。

IN/NOT IN

IN または NOT IN を比較演算子として使用して、1 つ以上の文字列値の一致 (または不一致) に基づいてフィルタリングします。Amazon Personalize は、正確な文字列に対してのみフィルターを適用します。

比較演算子

=、 < 、<=、 > 、>=、!= 演算子を使用して、プレースホルダーパラメータに渡されたデータを含む数値データを等価性についてテストします。

アスタリスク (*) 文字

すべてのタイプのインタラクションを含めたり、除外したりするために * を使用します。Interactions データセットの EVENT_TYPE フィールドを使用するフィルター式にのみ * を使用します。

パイプ区切り文字

パイプ区切り文字 (|) を使用して、複数の式を連鎖します。詳細については、「複数の式の組み合わせ」を参照してください。

パラメータ

比較演算子または IN 演算子を使用する式の場合、ドル記号 ($) とパラメータ名を使用して、プレースホルダーパラメータを値として追加します。例えば $GENRES です。この例では、レコメンデーションを取得する際に、フィルタリングする 1 つまたは複数のジャンルを指定します。

> **Note**
>
> パラメータ名は、式に追加するときに定義します。パラメータ名はフィールド名と一致する必要はありません。フィールド名に類似し、かつ、覚えやすいパラメータ名を使用することをお勧めします。レコメンデーションリクエストにフィルターを適用するときに、パラメータ名 (大文字と小文字が区別されます) を使用します。 AWS SDKS の使用時にプレースホルダーパラメータを含むフィルターを適用する方法の例については、「」を参照してくださいフィルターの適用 (AWS SDKs)。

フィルター式の例

次のセクションのフィルター式を使用して、独自のフィルター式を作成する方法について説明します。

トピック

- アイテムレコメンデーションのフィルター式の例
- ユーザーセグメントのフィルター式
- アクションレコメンデーションのフィルター式の例
- 複数の式の組み合わせ

アイテムレコメンデーションのフィルター式の例

以下のフィルター式は、アイテムインタラクション、アイテムメタデータ、およびユーザーメタデータに基づいてアイテムレコメンデーションをフィルタリングする方法を示しています。それらはデータ型別に編成されています。

トピック

アイテムインタラクションデータ

次の式は、$EVENT_TYPE パラメータを使用してレコメンデーションを取得するときに指定する 1 つのイベントタイプ (クリックなど) または複数のイベントタイプに基づいてアイテムを除外します。

```
EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ($EVENT_TYPE)
```

次の式は、ユーザーがクリックまたはストリーミングしたアイテムを除外します。

```
EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("click", "stream")
```

次の式は、ユーザーがクリックしたアイテムのみを含みます。

```
INCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("click")
```

アイテムデータ

次の式は、$CATEGORY パラメータを使用してレコメンデーションを取得するときに指定する 1 つまたは複数のカテゴリに基づいてアイテムを除外します。

```
EXCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY)
```

次の条件式には、現在の商品 (関連商品レコメンドのリクエストで指定した商品) よりも安価で、現在の商品と同じスタジオで作成された商品のみが含まれます。フィルターを CurrentItem 要素で適用できるのは、ドメインのユースケースまたはカスタムレシピが関連アイテムのレコメンデーションを生成する場合のみです。

```
INCLUDE ItemID WHERE Items.PRICE < CurrentItem.PRICE AND Items.GENRE IN
 CurrentItem.GENRE
```

次の式は、カテゴリ別フィールドの複数レベルに基づいてアイテムを除外します。CATEGORY_L1 の値が shoe のアイテムでも、CATEGORY_L2 の値がboot ではないアイテムは除外されます

```
EXCLUDE ItemID WHERE Items.CATEGORY_L1 IN ("shoe") AND Items.CATEGORY_L2 NOT IN
 ("boot")
```

以下の式は、$PRICE パラメータを使用してレコメンデーションを取得する際に指定した価格以下の価格を持つアイテムのみを含めます。

```
INCLUDE ItemID WHERE Items.PRICE <= $PRICE
```

次の式には、レコメンデーションを取得するときに指定する (UNIX エポック時間) より前に作成されたアイテムのみが含まれます。

```
INCLUDE ItemID WHERE Items.CREATION_TIMESTAMP < $DATE
```

次の式は、$GENRE パラメータを使用してレコメンデーションを取得するときに指定した 1 つまたは複数のジャンルのアイテムのみを含みます。

```
INCLUDE ItemID WHERE Items.GENRE IN ($GENRE)
```

次の式には、現在のアイテムよりも高価で、指定したタイムスタンプ (UNIX エポック時間) よりも後に作成されたアイテムのみが含まれます。このフィルターは、関連商品のレコメンデーションを取得していて、価格やさまざまな作成日に基づいて特定のビジネスルールを適用したい場合に使用できます。

```
INCLUDE ItemID WHERE Items.PRICE < CurrentItem.PRICE AND Items.CREATION_TIMESTAMP >
 $DATE
```

ユーザーデータ

次の式は、$GENRE パラメータを使用してレコメンデーションを取得するときに指定する 1 つまたは複数のジャンルのアイテムを除外しますが、現在のユーザーの年齢が、$AGE パラメータを使用してレコメンデーションを取得するときに指定する値と等しい場合に限ります。

```
EXCLUDE ItemID WHERE Items.GENRE IN ($GENRE) IF CurrentUser.AGE = $AGE
```

次の式は、現在のユーザーの年齢が 18 を超えている場合、CATEGORY_L1 向けの watchと CATEGORY_L2 向けの luxuryを持つアイテムのみを含みます。

```
INCLUDE ItemID WHERE Items.CATEGORY_L1 IN ("watch") AND Items.CATEGORY_L2 IN ("luxury")
 IF CurrentUser.AGE > 18
```

ユーザーセグメントのフィルター式

以下のフィルター式は、アイテムインタラクションデータとユーザーメタデータに基づいてユーザーセグメントをフィルタリングする方法を示しています。それらはデータ型別に編成されています。

ユーザーデータ

次のフィルター式は、ユーザーセグメントを取得するときに指定した値と等しいメンバーシップステータスを持つユーザーのみを含みます。

```
INCLUDE UserID WHERE Users.MEMBERSHIP_STATUS IN ($MEMBERSHIP)
```

次のフィルター式は、ユーザーセグメントを取得するときに指定した値未満の AGE のユーザーを除外します。

```
EXCLUDE UserID WHERE Users.AGE < $AGE
```

アイテムインタラクションデータ

次のフィルター式は、アイテムをクリックまたは評価したユーザーのみを含みます。

```
INCLUDE UserID WHERE Interactions.EVENT_TYPE IN ("click", "rating")
```

以下のフィルター式は、ユーザーセグメントを取得するときに指定したイベントタイプとアイテムインタラクションしたユーザーセグメントからユーザーを除外します。

```
EXCLUDE UserID WHERE Interactions.EVENT_TYPE IN ($EVENT_TYPE)
```

アクションレコメンデーションのフィルター式の例

以下のフィルター式の例は、アクションインタラクションデータ、アクションデータ、およびユーザーデータに基づいてアクションをフィルタリングする方法を示しています。それらはデータ型別に編成されています。

トピック

アクションインタラクションデータ

次のフィルター式は、レコメンデーションを取得するときに指定したイベントタイプを持ち、ユーザーがインタラクションしたアクションのみをレコメンデーションに含めます。

```
INCLUDE ActionID WHERE Action_Interactions.EVENT_TYPE IN ($EVENT_TYPE)
```

次のフィルター式は、ユーザーが実行していないアクションをイベントタイプに基づいて除外します。

```
EXCLUDE ActionID WHERE Action_Interactions.EVENT_TYPE IN ("NOT_TAKEN")
```

アクションデータ

次の式は、$CATEGORY パラメータを使用してレコメンデーションを取得するときに指定する 1 つまたは複数のカテゴリに基づいてアクションを除外します。

```
EXCLUDE ActionID WHERE Actions.CATEGORY IN ($CATEGORY)
```

次の式は、レコメンデーションを取得するときに指定した値より大きい値を持つアクションのみを含めます。

```
INCLUDE ActionID WHERE Actions.VALUE > ($VALUE)
```

ユーザーデータ

次の式は、現在のユーザーがプレミアムメンバーシップを持っている場合、プレミアムメンバー向けのアクションのみを含めます。

```
INCLUDE ActionID WHERE Action.MEMBERSHIP_LEVEL IN ("Premium") IF CurrentUser.MEMBERSHIP
 = $PREMIUM
```

次の式は、現在のユーザーがプレミアムメンバーの場合、レコメンデーションを取得するときに指定した値より小さい VALUE を持つアクションを除外します。

```
EXCLUDE ActionID WHERE Actions.VALUE < ($VALUE) IF CurrentUser.MEMBERSHIP = $PREMIUM
```

複数の式の組み合わせ

パイプ区切り文字 (|) を使用して、複数の式を組み合わせます。1 つのフィルターを使用して、アイテムデータセットとアイテムインタラクションデータセット、またはアクションデータセットとアクションインタラクションデータセットをフィルタリングするときには、複数の式を組み合わせて使用します。各式は最初に独立して評価され、結果は 2 つの結果の和集合または共通部分のいずれかになります。以下の例は、Items データセットとアイテムインタラクションデータセット用の式を作成する方法を示していますが、Actions と Action interactions を扱う場合も同じルールが適用されます。

マッチング式の例

両方の式が EXCLUDE を使用するか、両方の式が INCLUDE を使用する場合、結果は次のように 2 つの結果の和集合になります (A と B は異なる式です)。

- `Exclude A | Exclude B` は `Exclude result from A or result from B` に等しい
- `Include A | Include B` は `Include result from A or result from B` に等しい

次の例は、INCLUDE を使用する 2 つの式を組み合わせる方法を示しています。最初の式には、$CATEGORY パラメータを使用してレコメンデーションを取得するときに指定する 1 つまたは複数のカテゴリを持つアイテムのみが含まれます。2 番目の式は、ユーザーが favorite としてマークしたアイテムを含みます。レコメンデーションには、ユーザーがお気に入りとしてマークしたアイテムとともに、指定したカテゴリのアイテムのみが含まれます。

```
INCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY) | INCLUDE ItemID WHERE
 Interactions.EVENT_TYPE IN ("favorite")
```

INCLUDE と EXCLUDE の例

1 つ以上の式が INCLUDE を使用し、もう 1 つの式が EXCLUDE を使用する場合、結果は次のように INCLUDE 式の結果から EXCLUDE 式の結果を減じたものとなります (A、B、C、および D は異なる式です)。

- `Include A | Exclude B` は `Include result from A - result from B` に等しい
- `Include A | Include B | Exclude C | Exclude D` は `Include (A or B) - (C or D)` に等しい

式の順序は関係ありません。EXCLUDE 式が INCLUDE 式の前にあっても、結果は同じです。

次の例は、INCLUDE 式と EXCLUDE 式を組み合わせる方法を示しています。最初の式には、$GENRE パラメータを使用してレコメンデーションを取得するときに指定する 1 つまたは複数のジャンルを持つアイテムのみが含まれます。2 番目の式は、ユーザーがクリックまたはストリーミングしたアイテムを除外します。レコメンデーションには、これまでにクリックまたはストリーミングされていない、指定したジャンルのアイテムのみが含まれます。

```
INCLUDE ItemID WHERE Items.GENRE IN ($GENRE) | EXCLUDE ItemID WHERE
 Interactions.EVENT_TYPE IN ("click", "stream")
```

リアルタイムレコメンデーションのフィルタリング

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用して、リアルタイムのレコメンデーションをフィルタリングできます。

パーソナライズされたアイテムレコメンデーションや類似のアイテムを取得するとき、リクエストでプロモーションを指定できます。プロモーションでは、フィルターを使用して設定可能なおすすめアイテムのサブセットに適用される追加のビジネスルールを定義します。詳細については、レコメンデーション内のアイテムのプロモーションを参照してください。

トピック

- リアルタイムレコメンデーションのフィルタリング (コンソール)
- リアルタイムレコメンデーションのフィルタリング (AWS CLI)
- リアルタイムレコメンデーションのフィルタリング (AWS SDKs)

リアルタイムレコメンデーションのフィルタリング (コンソール)

コンソールを使用してリアルタイムのレコメンデーションをフィルタリングするには、フィルターを作成してから、それをレコメンデーションのリクエストに適用します。

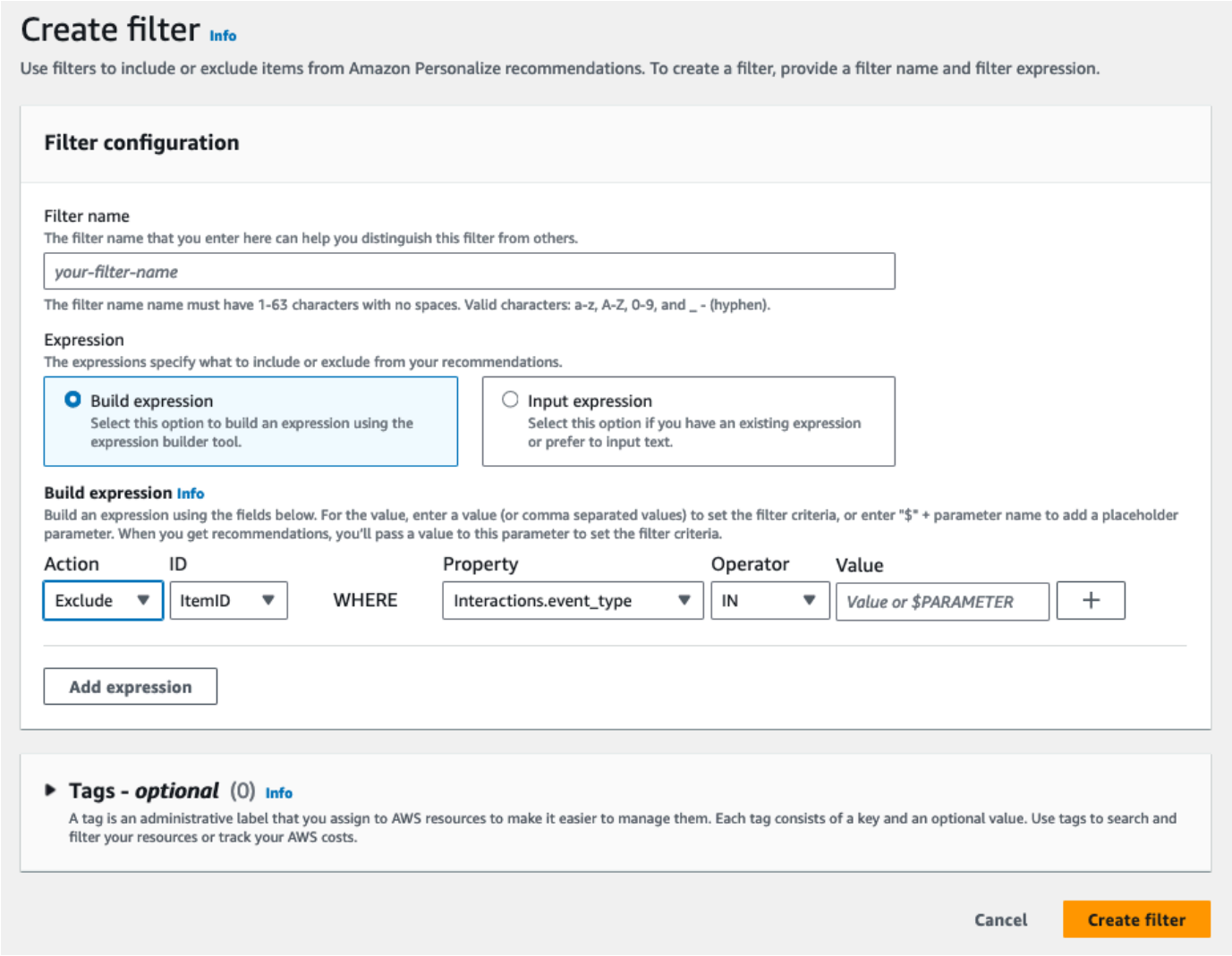
> **Note**
>
> 2020 年 11 月 10 日より前にデプロイされたパラメータとキャンペーンを持つフィルターを使用してレコメンデーションをフィルタリングするには、UpdateCampaign 操作を使用してキャンペーンを再デプロイするか、新しいキャンペーンを作成する必要があります。

フィルターの作成 (コンソール)

コンソールでフィルターを作成するには、フィルタリングされたレコメンデーションを取得するために使用するキャンペーンまたはレコメンダーを含むデータセットグループを選択します。次に、フィルター名とフィルター式を指定します。

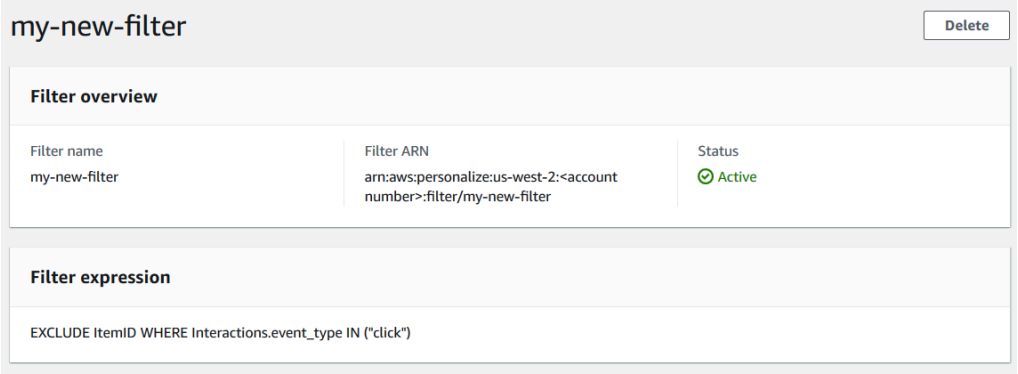
フィルターを作成するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. フィルタリングしたレコメンデーションを取得するために使用したいキャンペーンやレコメンデーションを含むデータセットグループを選択します。
3. ナビゲーションページで、[フィルター] を選択してから、[新しいフィルターの作成] を選択します。[Create filter] (フィルターの作成) ページが表示されます。



4. [Filter name] (フィルター名) で、フィルターの名前を入力します。レコメンデーションリクエストに適用するときに、この名前でフィルターを選択します。
5. [Expression] (式) で、[Build expression] (式を作成) または [Add expression manually] (式を手動で追加) を選択し、式を作成または挿入します。
   - 式ビルダーを使用するには、[Build expression] (式を作成) を選択します。式ビルダーは、正しくフォーマットされたフィルター式を作成するための構造、フィールド、およびガイドラインを提供します。詳細については、「フィルター式ビルダーの使用」を参照してください。
   - 独自の式を入力するには、[Add expression manually] (式を手動で追加) を選択します。詳細については、「フィルター式の要素」を参照してください。
6. [Finish] を選択します。フィルターの概要ページには、フィルターの Amazon リソースネーム (ARN)、ステータス、および完全なフィルター式が表示されます。フィルターを削除するに

は、[Delete] (削除) を選択します。概要ページから移動した後のフィルターの検索と削除については、「フィルターの削除 (コンソール)」を参照してください。



フィルターの適用 (コンソール)

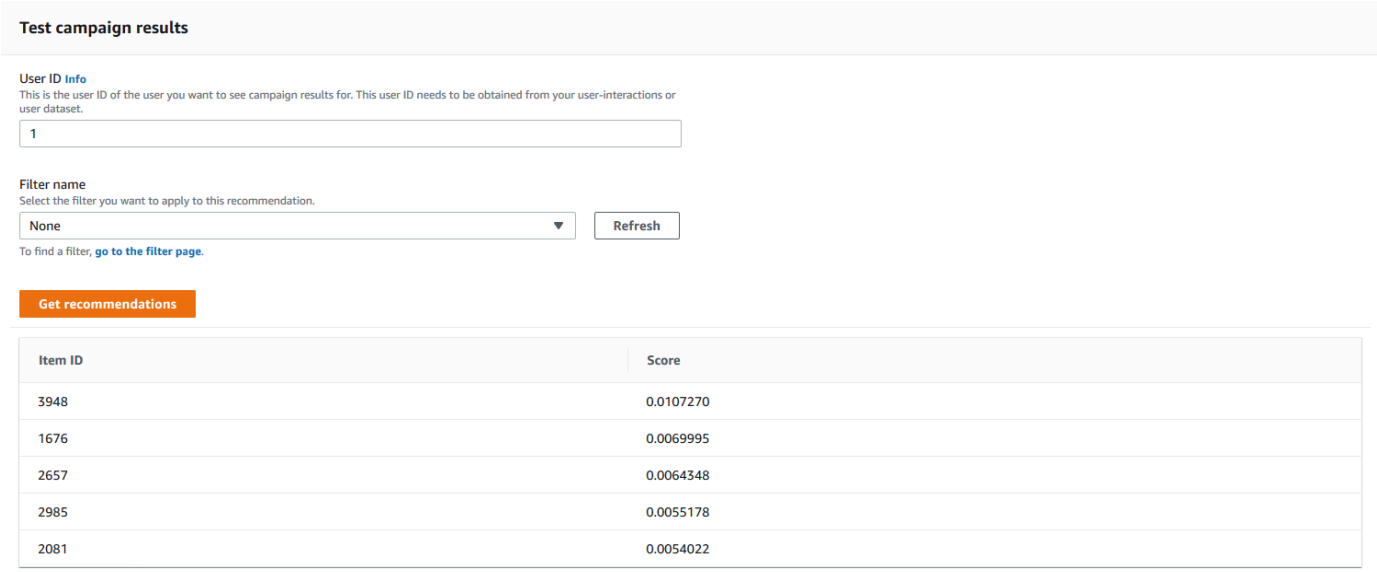
フィルターを適用するには、[レコメンダーのテスト] (レコメンダーの場合) または [キャンペーン結果のテスト] (カスタムキャンペーンの場合) で、フィルターを選択し、フィルターパラメータ値を入力します。その後、ユーザー向けのレコメンデーションを取得します。

> **Important**
>
> `INCLUDE` 要素を使用するフィルター式の場合、式で定義されているすべてのパラメータの値を指定する必要があります。`EXCLUDE` 要素を使用する式を持つフィルターの場合、`filter-values` を省略できます。この場合、Amazon Personalize は、式のその部分を使用してレコメンデーションをフィルタリングしません。

フィルターを適用するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. フィルタリングしたレコメンデーションを取得するために使用したいキャンペーンやレコメンデーションを含むデータセットグループを選択します。
3. データセットグループのタイプに応じて、次のいずれかを行います。
   a. ドメインデータセットグループの場合は、ナビゲーションペインで [レコメンダー] を選択します。
   b. Custom データセットグループまたはカスタムリソースの場合、ナビゲーションペインで [カスタムリソース]、[キャンペーン] の順に選択します。

4. レコメンダーまたはキャンペーンページで、ターゲットとなるレコメンダーまたはキャンペーンを選択します。
5. 比較のために、まず、フィルターを適用せずに、レコメンデーションを取得します。[レコメンダーのテスト] / [キャンペーン結果のテスト] で、レコメンデーションを取得するユーザーの ID または関連アイテムのアイテムの ID を入力し、[レコメンデーションを取得] を選択します。上位レコメンデーションを含むテーブルが表示されます。

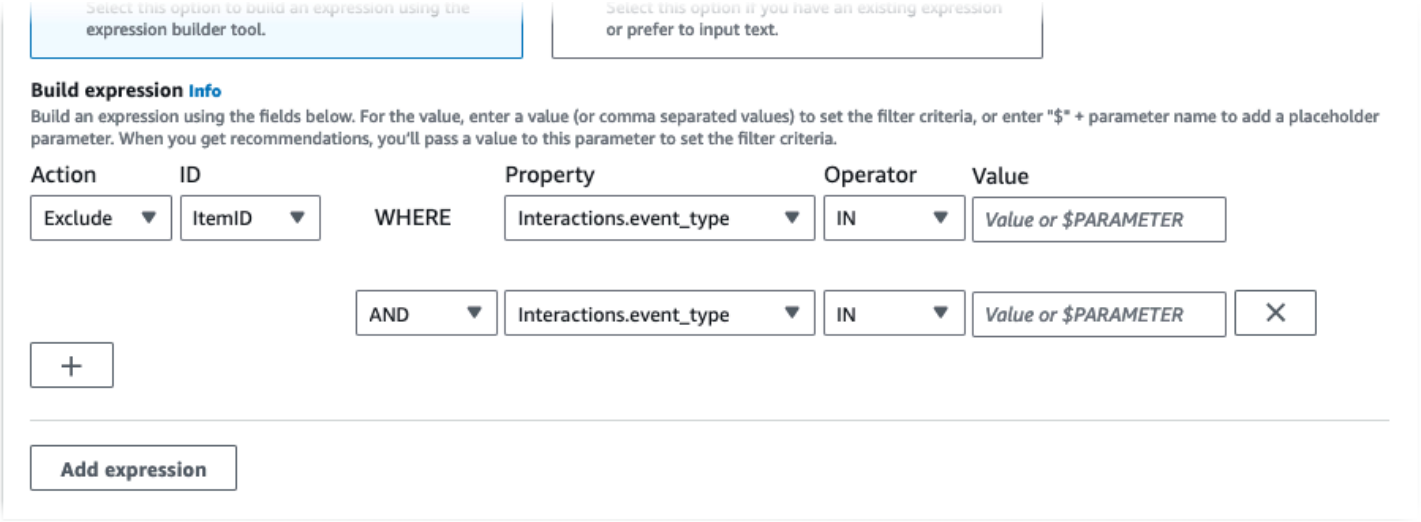


6. [Filter name] (フィルター名) のメニューから、作成したフィルターを選択します。フィルターにプレースホルダーパラメータがある場合は、各パラメータに関連付けられたフィールドが表示されます。
7. プレースホルダーパラメータとともにフィルターを使用している場合は、各パラメータに値を入力して、フィルター基準を設定します。1 つのパラメータに複数の値を使用するには、各値をコンマで区切ります。
8. 前のステップと同じ `User ID` または `Item ID` を使用して、[レコメンデーションを取得] を選択します。レコメンデーションテーブルが表示されます。

   例えば、ユーザーが推奨されたアイテムを既に購入していた場合、フィルターはレコメンデーションリストからそのアイテムを削除します。この例では、アイテム 2657、2985 が、ユーザーが購入しなかった最も適切なアイテム (アイテム 2641 および 1573) に置き換えられました。

フィルター式ビルダーの使用

フィルターの作成ページの式ビルダーには、正しくフォーマットされたフィルターを構築するための構造、フィールド、ガイドラインが表示されます。



フィルター式を作成するには

- [タイプ]、アクション、プロパティ、演算子、および 値 フィールドを使用して、式を作成します。

  [Value] (値) で、固定値を入力するか、レコメンデーションを取得する際にフィルター基準を設定するために、$ + パラメータ名を入力します。例えば $GENRES です。レコメンデーションを取得したら、フィルタリングする 1 つまたは複数の値を指定します。この例では、レコメンデーションを取得するときに、ジャンルまたはジャンルのリストを提供します。

  複数の非パラメータ値をコンマで区切ります。コンマ区切りのパラメータをフィルターに追加することはできません。

  > **Note**
  >
  > [Property] (プロパティ) (`dataset.field` 形式) を選択した後、AND または OR 条件によって連鎖された後続の行の [Property] (プロパティ) の値は、同じ `dataset` を使用する必要があります。

- + ボタンと X ボタンを使用して、行を式に追加したり、式から削除したりします。最初の行を削除することはできません。

- 新しい行については、[AND] メニューの AND、IF、または OR 演算子を使用して、連鎖条件を作成します。

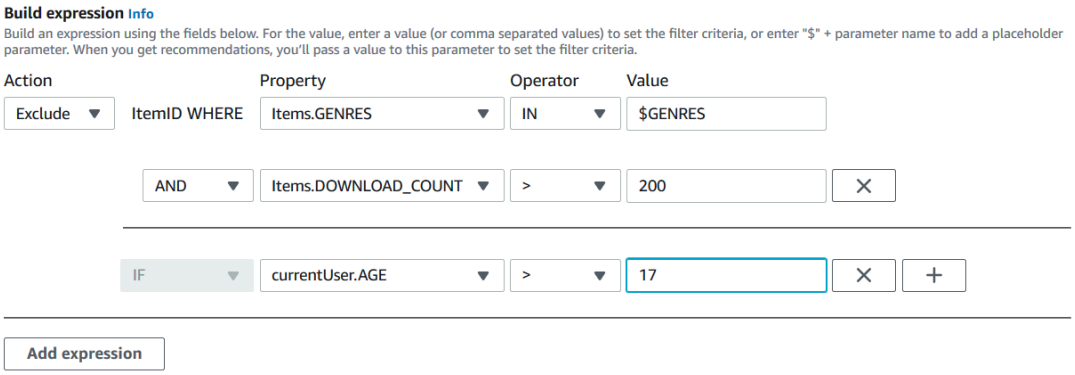
  IF 条件の場合

  - 各式には 1 つの IF アイテムのみを含めることができます。IF 条件を削除すると、式ビルダーはそれに続くすべての AND 条件を削除します。
  - IF 条件は、CurrentUser でフィルタリングする式にのみ使用できます。
- [式を追加] ボタンを選択して、追加のフィルター式を追加し、さらに正確なフィルタリングを行います。各式は最初に独立して評価され、結果は 2 つの結果の和集合になります。

> **Note**
>
> アイテムおよびアイテムインタラクションデータセットの両方、またはアクションおよびアクションインタラクションデータセットの両方を使用するフィルターを作成するには、複数の式を使用する必要があります。

式ビルダーの例

次の例は、レコメンデーションを取得するときに指定したジャンル ($GENRES プレースホルダーパラメータに留意してください) を持ち、アイテムを除外するフィルターを作成する方法を示しています。フィルターでは、200 以上の DOWNLOAD_COUNT を持つアイテムも除外されます。ただし、現在のユーザーの年齢が 17 より高い場合に限ります。



フィルターの削除 (コンソール)

フィルターを削除すると、データセットグループのフィルターのリストからフィルターが削除されます。

> **Important**
>
> バッチ推論ジョブの進行中は、フィルターを削除できません。

フィルターを削除するには (コンソール)

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [Dataset groups] (データセットグループ) のリストから、削除するフィルターを含むデータセットグループを選択します。
3. ナビゲーションペインで、[Filters] (フィルター) を選択します。
4. フィルターのリストから、削除するフィルターを選択し、[View Details] (詳細を表示) を選択します。フィルターの詳細ページが表示されます。
5. [Delete] (削除) を選択し、確認ダイアログボックスで削除を確認します。

リアルタイムレコメンデーションのフィルタリング (AWS CLI)

を使用してレコメンデーションをフィルタリングするには AWS CLI、フィルターを作成してから、GetRecommendationsまたは GetPersonalizedRankingリクエストでフィルター ARN を指定して適用します。

> **Important**
>
> 2020 年 11 月 10 日より前にデプロイされたパラメータとキャンペーンを使用してレコメンデーションを持つフィルターを使用してフィルタリングするには、UpdateCampaign コールを使用してキャンペーンを再デプロイするか、新しいキャンペーンを作成する必要があります。

フィルターの作成 (AWS CLI)

次の `create-filter` 操作を使用して、フィルターを作成し、フィルター式を指定します。

`Filter name` をフィルターの名前に、`Dataset group ARN` をデータセットグループの Amazon リソースネーム (ARN) に、それぞれ置き換えます。サンプル `filter-expression` を独自のフィルター式に置き換えます。

```
aws personalize create-filter \
  --name Filter name \
  --dataset-group-arn dataset group arn \
  --filter-expression "EXCLUDE ItemID WHERE Items.CATEGORY IN (\"$CATEGORY\")"
```

成功すると、フィルター ARN が表示されます。後で使用するために記録します。フィルターがアクティブであることを確認するには、フィルターを使用する前に DescribeFilter 操作を使用します。

API の詳細については、「CreateFilter」を参照してください。例を含むフィルター式の詳細については、「フィルター式の構造と要素」を参照してください。

フィルターの適用 (AWS CLI)

get-recommendations、get-action-recommendations、または get-personalized-ranking 操作を使用する場合は、filter-arn と任意のフィルター値をパラメータとして渡すことによってフィルターを適用します。

get-recommendations 動作の例を次に示します。Campaign ARN をキャンペーンの Amazon リソースネーム (ARN) に、User ID をレコメンデーションを取得しているユーザーの ID に、および Filter ARN をフィルターの ARN に、それぞれ置き換えます。キャンペーンではなくレコメンダーからレコメンデーションを受け取っている場合は、--campaign-arn の代わりに recommender-arn を使用して、そのレコメンダーの ARN を指定してください。

式にパラメータがある場合は、filter-values オブジェクトを含めます。フィルター式の各パラメータについて、パラメータ名 (大文字と小文字が区別されます) と値を指定します。例えば、フィルター式に $GENRE パラメータがある場合は、キーとして「GENRE」を指定し、値として 1 つまたは複数のジャンル ("Comedy" など) を指定します。複数の値はコンマで区切ります。例えば "\"comedy\",\"drama\",\"horror"\" です。

> **Important**
>
> INCLUDE 要素を使用してアイテムを含めるフィルター式について、式で定義されているすべてのパラメータの値を指定する必要があります。EXCLUDE 要素を使用してアイテムを除外する式を含むフィルターについては、filter-values を省略できます。この場合、Amazon Personalize は、式のその部分を使用してレコメンデーションをフィルタリングしません。

```
aws personalize-runtime get-recommendations \
  --campaign-arn Campaign ARN \
```

```
  --user-id User ID \
  --filter-arn Filter ARN \
  --filter-values '{
      "Parameter name": "\"value\"",
      "Parameter name": "\"value1\",\"value2\",\"value3\""
    }'
```

フィルターの削除 (AWS CLI)

フィルターを削除するには、次の delete-filter 操作を使用します。filter ARN をフィルターの ARN に置き換えます。

```
aws personalize delete-filter --filter-arn Filter ARN
```

リアルタイムレコメンデーションのフィルタリング (AWS SDKs)

AWS SDKs、フィルターを作成してから、 GetRecommendationsまたは GetPersonalizedRankingリクエストでフィルター ARN を指定して適用します。

> **Important**
>
> 2020 年 11 月 10 日より前にデプロイされたパラメータとキャンペーンを使用してレコメンデーションを持つフィルターを使用してフィルタリングするには、UpdateCampaign コールを使用してキャンペーンを再デプロイするか、新しいキャンペーンを作成する必要があります。

フィルターの作成 (AWS SDKs)

CreateFilter 操作で新しいフィルターを作成します。次のコードは、フィルターを作成する方法を示しています。フィルター名、データセットグループの Amazon リソースネーム (ARN) を指定し、フィルター式を指定します。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_filter(
    name = 'Filter Name',
```

```
    datasetGroupArn = 'Dataset Group ARN',
    filterExpression = 'EXCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY)'
)
filter_arn = response["filterArn"]
print("Filter ARN: " + filter_arn)
```

SDK for Java 2.x

```
public static String createFilter(PersonalizeClient personalizeClient,
                                  String filterName,
                                  String datasetGroupArn,
                                  String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .datasetGroupArn(datasetGroupArn)
                .filterExpression(filterExpression)
                .build();

        return personalizeClient.createFilter(request).filterArn();
    }
    catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  name: 'NAME', /* required */
  filterExpression: 'FILTER_EXPRESSION' /*required */
}
```

```
export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateFilterCommand(createFilterParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

後で使用するためにフィルター ARN を記録します。フィルターがアクティブであることを確認するには、フィルターを使用する前に DescribeFilter 操作を使用します。API の詳細については、「CreateFilter」を参照してください。例を含むフィルター式の詳細については、「フィルター式の構造と要素」を参照してください。

フィルターの適用 (AWS SDKs)

GetRecommendations、 GetActionRecommendations、または GetPersonalizedRanking オペレーションを使用する場合は、 `filterArn`および任意のフィルター値をパラメータとして渡してフィルターを適用します。

次のコードは、フィルタリングされた Amazon Personalize のユーザー向けアイテムレコメンデーションを取得する方法を示しています。レコメンデーションを取得させるユーザーの ID、キャンペーンの Amazon リソースネーム (ARN)、フィルターの ARN を指定します。キャンペーンではなくレコメンダーからレコメンデーションを受け取っている場合は、`recommenderArn` の代わりに `campaignArn` を使用して、そのレコメンダーの ARN を指定してください。

`filterValues` の場合、フィルター式のオプションの各パラメータについて、パラメータ名 (大文字と小文字が区別されます) と 1 つまたは複数の値を指定します。例えば、フィルター式に $GENRES パラメータがある場合は、キーとして「GENRES」を指定し、値として 1 つまたは複数のジャンル ("\"Comedy"\" など) を指定します。複数の値については、各値をコンマで区切ります。例えば "\"comedy\",\"drama\",\"horror\"" です。

> **Important**
>
> INCLUDE 要素を使用してアイテムを含めるフィルター式について、式で定義されているすべてのパラメータの値を指定する必要があります。EXCLUDE 要素を使用してアイテムを除外す

る式を含むフィルターについては、filter-values を省略できます。この場合、Amazon Personalize は、式のその部分を使用してレコメンデーションをフィルタリングしません。

SDK for Python (Boto3)

```
import boto3

personalize_runtime = boto3.client("personalize-runtime")

response = personalize_runtime.get_recommendations(
    campaignArn = "Campaign ARN",
    userId = "User ID",
    filterArn = "Filter ARN",
    filterValues = {
      "Parameter name": "\"value1\"",
      "Parameter name": "\"value1\",\"value2\",\"value3\""
      ....
    }
)
```

SDK for Java 2.x

次の例では、2 つのパラメータを使用しています。1 つは 2 つの値を持ち、もう 1 つは 1 つの値を持ちます。フィルター式に応じて、コードを変更して、parameterName フィールドおよび parameterValue フィールドを追加または削除します。

```
public static void getFilteredRecs(PersonalizeRuntimeClient
 personalizeRuntimeClient,
                                   String campaignArn,
                                   String userId,
                                   String filterArn,
                                   String parameter1Name,
                                   String parameter1Value1,
                                   String parameter1Value2,
                                   String parameter2Name,
                                   String parameter2Value){

    try {

        Map<String, String> filterValues = new HashMap<>();
```

```
        filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("\"%1$s\"",
                parameter2Value));

        GetRecommendationsRequest recommendationsRequest =
 GetRecommendationsRequest.builder()
                .campaignArn(campaignArn)
                .numResults(20)
                .userId(userId)
                .filterArn(filterArn)
                .filterValues(filterValues)
                .build();

        GetRecommendationsResponse recommendationsResponse =
 personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: 'CAMPAIGN_ARN', /* required */
  userId: 'USER_ID',      /* required */
  numResults: 15,    /* optional */
```

```
  filterArn: 'FILTER_ARN',   /* required to filter recommendations */
  filterValues: {
    "PROPERTY": "\"VALUE\""  /* Only required if your filter has a placeholder
 parameter */
  }
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

フィルターの削除 (AWS Python SDK)

フィルターを削除するには、次の delete_filter の方法を使用します。filter ARN をフィルターの ARN に置き換えます。

```
import boto3
personalize = boto3.client("personalize")

response = personalize.delete_filter(
  filterArn = "filter ARN"
)
```

バッチレコメンデーションとユーザーセグメントのフィルタリング (カスタムリソース)

バッチレコメンデーションとユーザーセグメントのフィルタリングは、リアルタイムレコメンデーションのフィルタリングとほぼ同じように機能します。バッチレコメンデーションとユーザーセグメント (カスタムリソース) で説明したのと同じワークフローに従います。バッチレコメンデーションまたはユーザーセグメントをフィルタリングするには、次の操作を行います。

1. リアルタイムのレコメンデーションの場合と同じようにフィルターを作成します。詳細については、リアルタイムレコメンデーションのフィルタリングを参照してください。
2. バッチレコメンデーション用の入力データを準備します。または ユーザーセグメントの入力データを準備しています。 の説明に従って、入力データを準備して Amazon S3 にアップロードします。フィルタでプレースホルダーパラメータを使用する場合は、`filterValues` オブジェクトを追加する必要があります。詳細については、「入力 JSON にフィルター値を指定します。」を参照してください。フィルターがプレースホルダーパラメーターを使用しない場合、入力データは バッチ推論ジョブの入力および出力 JSON の例 バッチセグメントジョブの入力および出力 JSON の例 の例のようになります。
3. フォルダまたは別の Amazon S3 バケットのいずれかで、出力データ用に個別の場所を作成します。
4. バッチ推論ジョブまたはバッチセグメントジョブを作成します。ジョブの作成時に、フィルターの Amazon リソースネーム (ARN) を指定します。
5. バッチ推論ジョブまたはバッチセグメントジョブが完了したら、Amazon S3 の出力場所からレコメンデーションまたはユーザーセグメントを取得します。

トピック

- 入力 JSON にフィルター値を指定します。
- バッチワークフローのフィルタリング (コンソール)
- バッチワークフローのフィルタリング (AWS SDK)

入力 JSON にフィルター値を指定します。

`$GENRE` などのプレースホルダーパラメータを持つフィルターについては、入力 JSON の `filterValues` オブジェクトでパラメータの値を指定します。`filterValues` オブジェクトの場合、各キーはパラメータ名です。各値は、パラメータとして渡す基準です。各値をエスケープ引用符で囲みます: `"filterValues":{"GENRES":"\"drama\""}` 複数の値については、各値をコンマ `"filterValues":{"GENRES":"\"horror\",\"comedy\",\"drama\""}` で区切ります。

バッチ推論ジョブの入力および出力 JSON の例

バッチ推論ジョブの JSON 入力ファイルの最初の数行の例を次に示します。この例には、`filterValues` オブジェクトが含まれています。`GENRES` キーは、フィルター式の `$GENRES` プレースホルダーに対応します。この例のジョブは User-Personalization レシピを

使用しています。RELATED_ITEMS レシピの場合は、userID の代わりに itemId を指定します。PERSONALIZED_RANKING レシピの場合は、userID と itemList を指定します。

```
{"userId": "5","filterValues":{"GENRES":"\"horror\",\"comedy\",\"drama\""}}
{"userId": "3","filterValues":{"GENRES":"\"horror\",\"comedy\""}}
{"userId": "34","filterValues":{"GENRES":"\"drama\""}}
```

レシピ別のバッチ推論ジョブ入力データのその他の例については、「バッチ推論ジョブの入力および出力 JSON の例」を参照してください。これらの例を出発点として使用し、上記の例から filterValues オブジェクトを追加できます。

バッチセグメントジョブの入力および出力 JSON の例

バッチセグメントジョブのフィルター値を持つ JSON 入力ファイルの最初の数行の例を次に示します。GENRES キーは、フィルター式の $GENRES プレースホルダーに対応します。

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action
\"","filterValues":{"COUNTRY":"\"Japan\""}}
{"itemAttributes": "ITEMS.genres = \"Horror\"","filterValues":{"COUNTRY":"\"United
 States\"\""}}
{"itemAttributes": "ITEMS.genres = \"Action\" AND ITEMS.genres = \"Adventure
\"","filterValues":{"COUNTRY":"\"England\""}}
```

レシピ別のバッチ推論ジョブ入力データのその他の例については、「バッチセグメントジョブの入力および出力 JSON の例」を参照してください。これらの例を出発点として使用し、上記の例からの filterValues オブジェクトを追加できます。

バッチワークフローのフィルタリング (コンソール)

Amazon Personalize コンソールでバッチワークフローをフィルタリングするには、フィルターを作成し、次にバッチ推論ジョブまたはバッチセグメントジョブを作成してフィルタを選択します。手順については、「バッチ推論ジョブの作成 (コンソール)」および「バッチセグメントジョブの作成 (コンソール)」を参照してください。

バッチワークフローのフィルタリング (AWS SDK)

AWS SDKsCreateBatchSegmentJobリクエストに FilterArnパラメータを含めます。CreateBatchInferenceJob

次のコードは、 AWS SDK for Python (Boto3)を使用するフィルターを持つバッチ推論ジョブを作成する方法を示しています。出力データには別の場所 (フォルダまたは別の Amazon S3 バケット) を使用することをお勧めします。すべてのフィールドの詳細な説明については、「バッチ推論ジョブの作成 (AWS SDKs)」を参照してください。

```
import boto3

personalize = boto3.client("personalize")

personalize_rec.create_batch_inference_job (
    solutionVersionArn = "Solution version ARN",
    jobName = "Batch job name",
    roleArn = "IAM role ARN",
    filterArn = "Filter ARN",
    jobInput =
        {"s3DataSource": {"path": "S3 input path"}},
    jobOutput =
        {"S3DataDestination": {"path": "S3 output path"}}
)
```

レコメンデーションの影響の測定

顧客がレコメンデーションに反応するにつれて、レコメンデーションが目標達成にどのように役立っているかを測定できます。視聴時間やクリック数が最も多いリソースなど、どのキャンペーンやレコメンダーが最も影響力があるかを特定できます。また、Amazon Personalize レコメンデーションのパフォーマンスを、サードパーティのサービスによって生成されたレコメンデーションと比較できます。

レコメンデーションのインパクトを測定するには、以下が役立ちます。

- メトリックス属性: Amazon Personalize のメトリクス属性は、指定したメトリクスと、インポートしたアイテムインタラクションとアイテムのデータに基づいてレポートを作成します。例えば、ユーザーが視聴した映画の合計時間や、クリックイベントの総数などです。
- A/B テスト: A/B テストでは、複数のバリエーションを試し、結果を比較します。A/B テストを使用すると、さまざまなレコメンデーション戦略を比較および評価し、レコメンデーションのインパクトを測定できます。

トピック

- メトリクス属性によるレコメンデーション効果の測定
- A/B テストを使用したレコメンデーションの影響の測定

メトリクス属性によるレコメンデーション効果の測定

アイテムレコメンデーションの影響を測定するために、メトリクス属性を作成できます。メトリクス属性は、インポートしたアイテムインタラクションとアイテムのデータ、および指定したメトリクスに基づいてレポートを作成します。例えば、ユーザーが視聴した映画の合計時間やクリックイベントの総数などです。Amazon Personalize は 15 分間のウィンドウで計算を集計します。 PutEvents および増分バルクデータの場合、Amazon Personalize は自動的にメトリクスレポートを Amazon に送信します CloudWatch。バルクデータの場合は、レポートを Amazon S3 バケットに発行することを選択できます。

インポートするインタラクションごとに、さまざまなキャンペーン、レコメンダー、サードパーティを比較するためのソースデータを含めます。ユーザーに見せたレコメンデーションのレコメンデーション ID や、サードパーティーなどのイベントソースを含めることができます。

例えば、2 つの Amazon Personalize レコメンダーからのおすすめ映画を表示するビデオストリーミングアプリがあるとします。どのレコメンダーが最も多くの視聴イベントを生成しているかを確認したい場合は、視聴イベントの総数を追跡するメトリクス属性を作成できます。そうすれば、ユーザーがレコメンデーションに反応した際の動画再生イベントを記録し、各イベントの `recommendationId` に組み込むことができます。Amazon Personalize `recommendationId` はを使用して各レコメンダーを識別します。イベントを記録すると、 で両方のレコメンダーについて 15 分ごとに集計された監視イベントの合計を表示できます CloudWatch。イベントに `recommendationId` または `eventAttributionSource` を含める方法を示すコードサンプルについては、「イベントメトリクスと属性レポート」を参照してください。

トピック

- ガイドラインと要件
- メトリクス属性の作成
- メトリクス属性の管理
- 結果の公開と表示

ガイドラインと要件

Amazon Personalize は、メトリクス属性を作成した後にのみ、レコメンデーションのインパクトの計算とレポートを開始します。最も完全な履歴を構築するには、インタラクションデータをインポートする前にメトリクス属性を作成することをお勧めします。Amazon Personalize コンソールでアイテムインタラクションデータセットのデータセットインポートジョブを作成する場合、新しいタブでメトリクス属性を作成するオプションがあります。その後、インポートジョブに戻って完了できます。

メトリクス属性を作成してイベントを記録するか、増分バルクデータをインポートすると、メトリクスごとに月額 CloudWatch コストが発生します。 CloudWatch 料金の詳細については、「Amazon の CloudWatch 料金」ページを参照してください。へのメトリクスの送信を停止するには CloudWatch、メトリクス属性 を削除します。

レコメンデーションのインパクトを長期的に確認するには、顧客がレコメンデーションに反応するたびにデータをインポートし続けてください。すでにデータをインポートしている場合でも、メトリクス属性を作成して、レコメンデーションの影響の測定を開始できます。ただし、Amazon Personalize は、作成前にインポートしたデータについてはレポートしません。

メトリクス属性を含むレポートを生成するためのガイドラインと要件は次のとおりです。

- にアクセスして にデータを配置するアクセス許可を Amazon Personalize に付与する必要があります CloudWatch。IAM ポリシーの例については、「Amazon Personalize に へのアクセスを許可する CloudWatch」を参照してください。
- Amazon S3 にメトリクスを発行するには、Amazon Personalize にバケットへの書き込み権限を付与します。また、メトリクス属性にバケットパスを指定する必要があります。IAM ポリシーの例については、「Amazon Personalize に対する、Amazon S3 リソースへのアクセス許可の付与」を参照してください。
- メトリクスを に発行するには CloudWatch、レコードが 14 日未満である必要があります。データが古い場合、これらのレコードは計算やレポートに含まれません。
- 重複するイベント (すべての属性が完全に一致するイベント) をインポートすると、メトリクスが不正確になるなど、予期しない動作が発生する可能性があります。インポートする前にバルクデータから重複レコードを削除し、`PutEvents` 操作で重複するイベントをインポートしないようにすることをお勧めします。
- アイテムインタラクションデータセットには `EVENT_TYPE` 列が必要です。
- Action インタラクションデータセット内のデータのメトリックレポートを作成することはできません。
- データセットグループごとに最大 1 つのメトリクス属性を作成できます。各メトリクス属性には最大 10 個のメトリクスを含めることができます。

ソースを比較するには、各インタラクションイベントに `recommendationId` または `eventAttributionSource` が含まれている必要があります。固有のイベント属性ソースは最大 100 個まで提供できます。`PutEvents` のコードの例については、「イベントメトリクスと属性レポート」を参照してください。

- `recommendationId` を指定すると、Amazon Personalize はソースキャンペーンまたはレコメンダーを自動的に判断し、レポートの EVENT_ATTRIBUTION_SOURCE 列で識別します。
- 両方の属性を指定した場合、Amazon Personalize は `eventAttributionSource` のみを使用します。
- ソースを指定しない場合、Amazon Personalize はレポートでソース `SOURCE_NAME_UNDEFINED` にラベルを付けます。

トピック

- Amazon Personalize に へのアクセスを許可する CloudWatch
- Amazon Personalize に対する、Amazon S3 リソースへのアクセス許可の付与

Amazon Personalize に へのアクセスを許可する CloudWatch

> **Important**
>
> アクセス許可を付与すると、Amazon Personalize は少量のデータを に配置して検証します CloudWatch。これには 0.30 USD 未満の 1 回限りのコストが発生します。 CloudWatch 料金の詳細については、「Amazon の CloudWatch 料金」ページを参照してください。

Amazon Personalize に へのアクセスを許可するには CloudWatch、 のPutMetricDataアクションを使用するアクセス許可をロールに付与する新しい AWS Identity and Access Management (IAM) ポリシーを Amazon Personalize サービスロールにアタッチします CloudWatch。以下のポリシーの例は、PutMetricData のアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon Personalize に対する、Amazon S3 リソースへのアクセス許可の付与

Amazon Personalize で Amazon S3 バケットへのアクセス許可を付与するには

- Amazon Personalize サービスロールに IAM ポリシーをアタッチして、バケットの PutObject アクションを使用するアクセス許可をロールに付与します。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

- PutObject アクションを使用するための許可を、Amazon Personalize プリンシパルに付与するバケットポリシーを出力の Amazon S3 バケットにアタッチします。

  暗号化に AWS Key Management Service（ AWS KMS) を使用する場合は、Amazon Personalize と Amazon Personalize の IAM サービスロールに、 キーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "personalize.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

メトリクス属性の作成

> **Important**
>
> メトリクス属性を作成してイベントを記録するか、増分バルクデータをインポートすると、メトリクスごとに月額 CloudWatch コストが発生します。 CloudWatch 料金の詳細については、「Amazon の CloudWatch 料金」ページを参照してください。へのメトリクスの送信を停止するには CloudWatch、メトリクス属性 を削除します。

メトリクスレポートの生成を開始するには、メトリクス属性を作成し、インタラクションデータをインポートします。メトリクス属性を作成する際は、レポートの対象となるイベントタイプのリストを指定します。イベントタイプごとに、Amazon Personalize がデータを収集するときに適用する関数を指定します。使用できる関数には、`SUM(DatasetType.COLUMN_NAME)` と `SAMPLECOUNT()` があります。

例えば、オンライン動画ストリーミングアプリを使用していて、レコメンデーションのクリック率と視聴された映画の合計時間の 2 つのメトリクスを追跡したいとします。ただし、アイテムデータセット内の各動画には `LENGTH` 属性が含まれます。メトリクス属性を作成し、それぞれにイベントタイプと機能を持つ 2 つのメトリクスを追加します。1 つ目は、`SAMPLECOUNT()` 関数付きの `Click` イベントタイプのものかもしれません。2 つ目は、`Watch` 関数付きの `SUM(Items.LENGTH)` イベントタイプのものかもしれません。

`SUM()` 関数はアイテムデータセットとアイテムインタラクションデータセットの数値列にのみ適用できます。アイテムデータセットの列に `SUM()` 関数を適用するには、まずアイテムメタデータをインポートする必要があります。

メトリクス属性は、Amazon Personalize コンソール AWS Command Line Interface、または AWS SDKS を使用して作成できます。

トピック

- メトリクス属性の作成 (コンソール)
- メトリクス属性 (AWS CLI) の作成
- メトリクス属性の作成 (AWS SDKs)

メトリクス属性の作成 (コンソール)

Amazon Personalize コンソールでメトリクス属性を作成するには、メトリクス属性ページに移動し、[メトリクス属性の作成] を選択します。メトリクス属性を作成するときは、オプションの Amazon S3 バケットパス、Amazon Personalize IAM サービスロール、およびレポートするメトリクスのリストを指定します。

Amazon Personalize コンソールでアイテムインタラクションデータセットのインポートジョブを作成する場合、新しいタブでメトリクス属性を作成するオプションがあります。その後、インポートジョブに戻って完了できます。すでに[メトリック属性の設定] ページを開いている場合は、ステップ 4 に進んでください。

メトリクス属性を作成するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. データセットグループを選択します。
3. ナビゲーションペインの、[カスタムリソース] で [メトリクス属性] を選択します。
4. [メトリクス属性の詳細] で [メトリクス属性を作成する] を選択します。
5. [メトリクス属性の設定] ページで、メトリクス属性に名前を付けます。
6. Amazon S3 のデータ出力パス用に Amazon S3 にメトリクスを発行する場合は、送信先の Amazon S3 バケットを入力します。これにより、データセットのインポートジョブを作成するたびにメトリクスを発行するオプションが有効になります。次の構文を使用します。

   s3://<name of your S3 bucket>/<folder> path>
7. 暗号化 AWS KMS に を使用している場合は、KMS キー ARN にキーの Amazon リソースネーム (ARN) AWS KMS を入力します。Amazon Personalize と Amazon Personalize IAM サービスロールにキーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。
8. [サービスロール] では、新しいサービスロールを作成するか、または既存のロールを使用するかを選択します。選択するロールには、 の`PutMetricData`アクセス許可が必要です CloudWatch。Amazon S3 に発行する場合、ロールには Amazon S3 バケットに対する `PutObject` アクセス許可が必要です。

   で作成したロールを使用するにはAmazon Personalize 向けの IAM ロールの作成、 CloudWatch と Amazon S3 のポリシーを追加する必要がある場合があります。

ポリシーの例については、「Amazon Personalize に へのアクセスを許可する CloudWatch」、「Amazon Personalize に対する、Amazon S3 リソースへのアクセス許可の付与」を参照してください。

9. [次へ] をクリックします。
10. [メトリクス属性の定義] ページで、メトリクスの定義方法を選択します。[メトリクス属性の作成 を選択してビルダーツールを使用します。JSON 形式でメトリクスを入力するには、[メトリクス属性の入力] を選択します。

    - [メトリクス属性の作成] を選択した場合は、メトリクスごとに名前とイベントタイプを指定し、関数を選択します。SUM( ) 関数の場合は、列名を選択します。[メトリクス属性を追加] を選択してメトリクスを追加します。
    - [メトリクス属性の入力] を選択した場合は、各メトリクスを JSON 形式で入力します。次に、メトリクスをフォーマットする方法を示します。

    ```
    {
        "EventType": "watch",
        "MetricName": "MinutesWatchedTracker",
        "MetricMathExpression": "SUM(Items.LENGTH)"
    }
    ```

11. [次へ] をクリックします。
12. [確認と作成] ページで新しいメトリクス属性の詳細を確認します。設定を変更するには、[戻る] を選択します。メトリクス属性を作成するには、[作成] を選択します。メトリクス属性が有効になっている場合は、データのインポートを開始して結果を表示できます。表示結果については、「結果の公開と表示」を参照してください。

メトリクス属性 (AWS CLI) の作成

次のコードは、 AWS Command Line Interfaceを使用してメトリクス属性を作成する方法を示しています。指定するロールには、Amazon S3 に発行する場合は、Amazon Amazon S3 バケットのPutObjectアクセスPutMetricData許可 CloudWatch が必要です。で作成したロールを使用するにはAmazon Personalize 向けの IAM ロールの作成、 CloudWatch と Amazon S3 のポリシーを追加する必要がある場合があります。すべてのポリシーの例については、「Amazon Personalize に へのアクセスを許可する CloudWatch」、「Amazon Personalize に対する、Amazon S3 リソースへのアクセス許可の付与」を参照してください。

メトリクスごとに、名前、イベントタイプ、式 (関数) を指定します。使用できる関数には SUM(DatasetType.COLUMN_NAME)と SAMPLECOUNT() があります。SUM() 関数では、データセットの種類と列名を指定します。例えば SUM(Items.LENGTH) です。各パラメータについては、「CreateMetricAttribution」を参照してください。

```
aws personalize create-metric-attribution \
--name metric attribution name \
--dataset-group-arn dataset group arn \
--metrics-output-config "{\"roleArn\": \"Amazon Personalize service role ARN\",
 \"s3DataDestination\":{\"kmsKeyArn\":\"kms key ARN\",\"path\":\"s3://bucket-
name/folder-name/\"}}" \
--metrics "[{
  \"eventType\": \"event type\",
  \"expression\": \"SUM(DatasetType.COLUMN_NAME)\",
  \"metricName\": \"metric name\"
}]"
```

メトリクス属性の作成 (AWS SDKs)

次のコードは、SDK for Python (Boto3) を使用してメトリクス属性を作成する方法を示しています。指定するロールには、Amazon S3 に発行する場合は、Amazon SAmazon S3 バケットのPutObjectアクセスPutMetricData許可 CloudWatch が必要です。で作成したロールを使用するにはAmazon Personalize 向けの IAM ロールの作成、 CloudWatch と Amazon S3 のポリシーを追加する必要がある場合があります。すべてのポリシーの例については、「Amazon Personalize にへのアクセスを許可する CloudWatch」、「Amazon Personalize に対する、Amazon S3 リソースへのアクセス許可の付与」を参照してください。

メトリクスごとに、名前、イベントタイプ、式 (関数) を指定します。使用できる関数には SUM(DatasetType.COLUMN_NAME)と SAMPLECOUNT() があります。SUM() 関数では、データセットの種類と列名を指定します。例えば SUM(Items.LENGTH) です。各パラメータについては、「CreateMetricAttribution」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

metricsList = [{
      "eventType": "event type",
      "expression": "SUM(DatasetType.COLUMN_NAME)",
```

```
      "metricName": "metric name"
}]

outputConfig = {
  "roleArn": "Amazon Personalize service role ARN",
  "s3DataDestination": {
    "kmsKeyArn": "key ARN",
    "path": "s3://<name of your S3 bucket>/<folder>"
  }
}
response = personalize.create_metric_attribution(
  name = 'metric attribution name',
  datasetGroupArn = 'dataset group arn',
  metricsOutputConfig = outputConfig,
  metrics = metricsList
)

metric_attribution_arn = response['metricAttributionArn']

print ('Metric attribution ARN: ' + metric_attribution_arn)

description = personalize.describe_metric_attribution(
    metricAttributionArn = metric_attribution_arn)['metricAttribution']

print('Name: ' + description['name'])
print('ARN: ' + description['metricAttributionArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

```
public static String createMetricAttribution(PersonalizeClient personalizeClient,
                                              String eventType,
                                              String expression,
                                              String metricName,
                                              String metricAttributionName,
                                              String roleArn,
                                              String s3Path,
                                              String kmsKeyArn,
                                              String datasetGroupArn) {
    String metricAttributionArn = "";

    try {
```

```
        MetricAttribute attribute = MetricAttribute.builder()
                .eventType(eventType)
                .expression(expression)
                .metricName(metricName)
                .build();

        ArrayList<MetricAttribute> metricAttributes = new ArrayList<>();
        metricAttributes.add(attribute);

        S3DataConfig s3DataDestination = S3DataConfig.builder()
                .kmsKeyArn(kmsKeyArn)
                .path(s3Path)
                .build();

        MetricAttributionOutput outputConfig = MetricAttributionOutput.builder()
                .roleArn(roleArn)
                .s3DataDestination(s3DataDestination)
                .build();

        CreateMetricAttributionRequest createMetricAttributionRequest =
 CreateMetricAttributionRequest.builder()
                .name(metricAttributionName)
                .datasetGroupArn(datasetGroupArn)
                .metrics(metricAttributes)
                .metricsOutputConfig(outputConfig)
                .build();
        CreateMetricAttributionResponse createMetricAttributionResponse =
 personalizeClient.createMetricAttribution(createMetricAttributionRequest);

        metricAttributionArn =
 createMetricAttributionResponse.metricAttributionArn();
        System.out.println("Metric attribution ARN: " + metricAttributionArn);
        return metricAttributionArn;
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateMetricAttributionCommand, PersonalizeClient } from
```

```
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the metric attribution param
export const createMetricAttributionParam = {
  name: "METRIC_ATTRIBUTION_NAME",            /* required */
  datasetGroupArn: "DATASET_GROUP_ARN",       /* required */
  metricsOutputConfig: {
    roleArn: "ROLE_ARN",                      /* required */
    s3DataDestination: {
      kmsKeyArn: "KEY_ARN",                                                  /*
 optional */
      path: "s3://<name of your output S3 bucket>/<folderName>/",    /* optional */
    },
  },
  metrics: [
    {
      eventType: "EVENT_TYPE",                      /* required for each metric */
      expression: "SUM(DatasetType.COLUMN_NAME)",   /* required for each metric */
      metricName: "METRIC_NAME",                    /* required for each metric */
    }
  ]
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateMetricAttributionCommand(createMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

メトリクス属性の管理

メトリクス属性を作成した後、更新または削除できます。メトリクス属性を削除する PutEvents と、Amazon Personalize は および への増分インポートに関連するレポートの送信を停止します CloudWatch。

トピック

メトリクス属性の更新

メトリクス属性を更新すると、メトリクスを追加および削除したり、出力設定を変更したりできます。メトリクス属性は、Amazon Personalize コンソール AWS Command Line Interface、または AWS SDKS を使用して更新できます。

トピック

メトリクス属性の更新 (コンソール)

Amazon Personalize コンソールでメトリクス属性を更新するには、[メトリクス属性]ページで変更を行います。

メトリクス属性を更新するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. データセットグループを選択します。
3. ナビゲーションペインで [メトリクス] を選択します。
4. 一番下のセクションで、[メトリクス属性] タブまたは [メトリクス属性設定] タブを選択し、変更を開始します。
    - メトリクスを追加または削除するには、[メトリクス属性] タブを選択し、[属性の編集] を選択します。[メトリクス属性の編集] ページで変更を加え、[更新] を選択して変更を保存します。

- Amazon S3 出力バケットまたは IAM サービスロールを変更するには、[メトリクス属性設定の編集] タブを選択し、[属性設定の編集] ページで変更を行います。[更新] を選択して変更を保存します。

メトリクス属性の更新 (AWS CLI)

メトリクス属性を作成したら、 AWS Command Line Interface （ AWS CLI) を使用してメトリクスを追加および削除し、出力設定を変更できます。次のコードは、update-metric-attribution コマンドを使用してメトリクスを削除する方法を示しています。

```
aws personalize update-metric-attribution \
--metric-attribution-arn metric attribution arn \
--remove-metrics metricName1 metricName2
```

次のコードは、メトリクスを追加して新しい出力の設定を指定する方法を示しています。

```
aws personalize update-metric-attribution \
--metric-attribution-arn metric attribution arn \
--metrics-output-config "{\"roleArn\": \"new role ARN\", \"s3DataDestination\":
{\"kmsKeyArn\":\"kms key ARN\",\"path\":\"s3://new-bucket-name/new-folder-name/\"}}" \
--add-metrics "[{
  \"eventType\": \"event type\",
  \"expression\": \"SUM(DatasetType.COLUMN_NAME)\",
  \"metricName\": \"metric name\"
}]"
```

成功した場合、Amazon Personalize は更新したメトリクス属性の ARN を返します。すべてのパラメータの詳細なリストについては、「UpdateMetricAttribution」を参照してください。

メトリクス属性の更新 (AWS SDK)

メトリクス属性を作成した後、メトリクスを追加または削除したり、出力の設定を変更したりできます。次のコードは、メトリクス属性からメトリクスを削除する方法を示しています。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

metricsToRemove = ["metricName1", "metricName2"]
```

```
response = personalize.update_metric_attribution(
  metricAttributionArn = "metric attribution ARN",
  removeMetrics = metricsToRemove
)
```

SDK for Java 2.x

```
public static void removeMetrics(PersonalizeClient client,
                                 String metricAttributionArn,
                                 String metric1Name,
                                 String metric2Name) {

    ArrayList<String> metricsToRemove = new ArrayList<>(Arrays.asList(metric1Name,
 metric2Name));

    try {

        UpdateMetricAttributionRequest request =
 UpdateMetricAttributionRequest.builder()
                .metricAttributionArn(metricAttributionArn)
                .removeMetrics(metricsToRemove)
                .build();

        UpdateMetricAttributionResponse response =
 client.updateMetricAttribution(request);
        System.out.println(response);

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import {UpdateMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});
```

```
// set the update request param
export const updateMetricAttributionParam = {
  metricAttributionArn: "METRIC_ATTRIBUTION_ARN",    /* required */
  removeMetrics: ["METRIC_NAME_1", "METRIC_NAME_2"]    /* specify list of names of
 metrics to delete */
};
export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new UpdateMetricAttributionCommand(updateMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

次のコードは、メトリクスを追加して新しい出力の設定を指定する方法を示しています。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

newMetrics = [{
      "eventType": "event type",
      "expression": "SUM(DatasetType.COLUMN_NAME)",
      "metricName": "metric name"
}]

newOutputConfig = {
  "roleArn": "Amazon Personalize service role ARN",
  "s3DataDestination": {
    "kmsKeyArn": "key ARN",
    "path": "s3://<name of your S3 bucket>/<folder>"
  }
}
```

```
response = personalize.update_metric_attribution(
  metricAttributionArn = "metric attribution arn",
  metricsOutputConfig = newOutputConfig,
  addMetrics = newMetrics
)
```

SDK for Java 2.x

```
public static void addMetricsAndUpdateOutputConfig(PersonalizeClient
 personalizeClient,
                                                    String metricAttributionArn,
                                                    String newMetric1EventType,
                                                    String newMetric1Expression,
                                                    String newMetric1Name,
                                                    String newMetric2EventType,
                                                    String newMetric2Expression,
                                                    String newMetric2Name,
                                                    String roleArn,
                                                    String s3Path,
                                                    String kmsKeyArn) {
    try {

        MetricAttribute newAttribute = MetricAttribute.builder()
                .eventType(newMetric1EventType)
                .expression(newMetric1Expression)
                .metricName(newMetric1Name)
                .build();

        MetricAttribute newAttribute2 = MetricAttribute.builder()
                .eventType(newMetric2EventType)
                .expression(newMetric2Expression)
                .metricName(newMetric2Name)
                .build();

        ArrayList<MetricAttribute> newAttributes = new
 ArrayList<>(Arrays.asList(newAttribute, newAttribute2));

        S3DataConfig newDataDestination = S3DataConfig.builder()
                .kmsKeyArn(kmsKeyArn)
                .path(s3Path)
                .build();

        MetricAttributionOutput newOutputConfig = MetricAttributionOutput.builder()
```

```
                .roleArn(roleArn)
                .s3DataDestination(newDataDestination)
                .build();

        UpdateMetricAttributionRequest request =
 UpdateMetricAttributionRequest.builder()
                .metricAttributionArn(metricAttributionArn)
                .metricsOutputConfig(newOutputConfig)
                .addMetrics(newAttributes)
                .build();

        UpdateMetricAttributionResponse response =
 personalizeClient.updateMetricAttribution(request);
        System.out.println("New metrics added!");
        System.out.println(response);

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import {UpdateMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

export const updateMetricAttributionParam = {
  metricAttributionArn: "METRIC_ATTRIBUTION_ARN",
  addMetrics: [
    {
      eventType: "EVENT_TYPE",                      /* required for each metric */
      expression: "SUM(DatasetType.COLUMN_NAME)",   /* required for each metric */
      metricName: "METRIC_NAME",                    /* required for each metric */
    }
  ],
  metricsOutputConfig: {
    roleArn: "ROLE_ARN",                       /* required */
```

```
    s3DataDestination: {
      kmsKeyArn: "KEY_ARN",                                                    /*
 optional */
      path: "s3://<name of your output S3 bucket>/<folderName>/",    /* optional */
    },
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new UpdateMetricAttributionCommand(updateMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

成功した場合、Amazon Personalize は更新したメトリクス属性の ARN を返します。すべてのパラメータの詳細なリストについては、「UpdateMetricAttribution」を参照してください。

メトリクス属性の削除

レポートが不要になった場合は、メトリクス属性を削除できます。メトリクス属性を削除すると、そのメトリクスと出力設定もすべて削除されます。

メトリクス属性を削除すると、Amazon Personalize は PutEvents および増分バルクデータに関連するレポートの送信を自動的に停止します CloudWatch。Amazon S3 にすでに送信 CloudWatch または公開されているデータは影響を受けません。メトリクス属性は、Amazon Personalize コンソール AWS Command Line Interface、または AWS SDKS を使用して削除できます。

トピック

- メトリクス属性の削除 (コンソール)
- メトリクス属性の削除 (AWS CLI)
- メトリクス属性の削除 (AWS SDKs)

メトリクス属性の削除 (コンソール)

メトリクス属性の概要ページでメトリクス属性を削除します。

メトリクス属性を削除するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. データセットグループを選択します。
3. ナビゲーションペインで [メトリクス] を選択します。
4. [削除] を選択し、削除を確定します。

メトリクス属性の削除 (AWS CLI)

を使用してメトリクス属性を削除するには AWS CLI、次のように `delete-metric-attribution` コマンドを使用します。

```
aws personalize delete-metric-attribution --metric-attribution-arn metric attribution
 ARN
```

メトリクス属性の削除 (AWS SDKs)

次のコードは、SDK for Python (Boto3) を使用してメトリクス属性を削除する方法を示しています。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.delete_metric_attribution(
  metricAttributionArn = 'metric attribution ARN'
)
```

SDK for Java 2.x

```
public static void deleteMetricAttribution(PersonalizeClient client, String
 metricAttributionArn) {

    try {
```

```
        DeleteMetricAttributionRequest request =
 DeleteMetricAttributionRequest.builder()
                .metricAttributionArn(metricAttributionArn)
                .build();

        DeleteMetricAttributionResponse response =
 client.deleteMetricAttribution(request);
        if (response.sdkHttpResponse().statusCode() == 200) {
            System.out.println("Metric attribution deleted!");
        }

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { DeleteMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

export const deleteMetricAttributionParam = {
  metricAttributionArn: "METRIC_ATTRIBUTION_ARN",
};
export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new DeleteMetricAttributionCommand(deleteMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

結果の公開と表示

Amazon Personalize は、各メトリクスのレポートを CloudWatch または Amazon S3 に送信します。

- PutEvents データおよび増分バルクデータの場合、Amazon Personalize は自動的にメトリクスをに送信します CloudWatch。でのレポートの表示と識別については、 CloudWatch「」を参照してくださいでのメトリクスの表示 CloudWatch。
- すべてのバルクデータについて、メトリクス属性の作成時に Amazon S3 バケットを指定すると、インタラクションデータのデータセットインポートジョブを作成するたびに、Amazon S3 バケットにメトリクスレポートを発行することを選択できます。

  Amazon S3 へのフローログの発行については、「メトリクスをAmazon S3 に発行する」を参照してください。

トピック

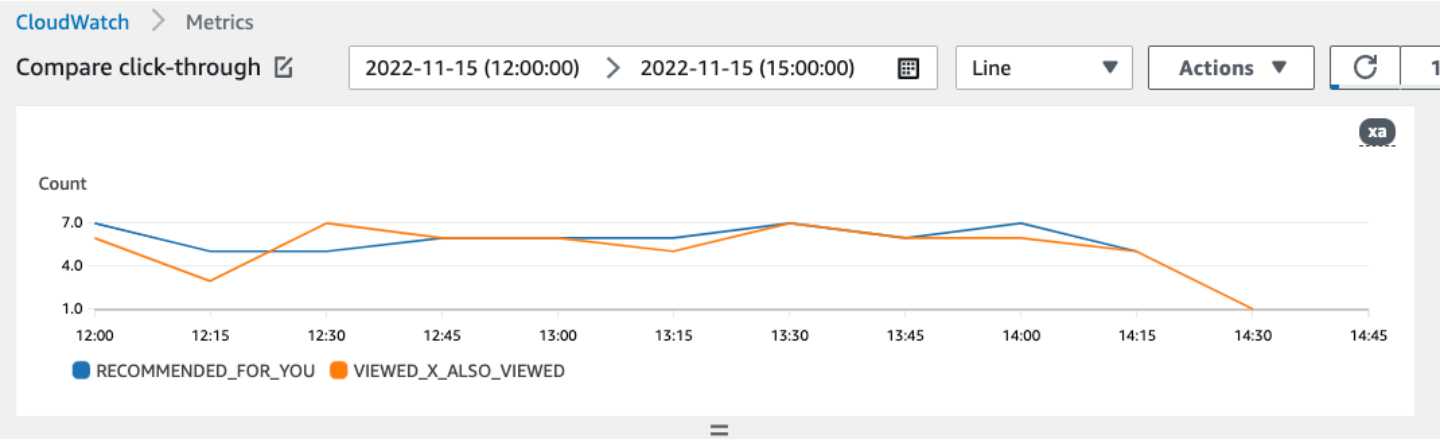
- でのメトリクスの表示 CloudWatch
- メトリクスをAmazon S3 に発行する

でのメトリクスの表示 CloudWatch

> **Important**
>
> メトリクス属性を作成してイベントを記録するか、増分バルクデータをインポートすると、メトリクスごとに月額 CloudWatch コストが発生します。 CloudWatch 料金の詳細については、「Amazon の CloudWatch 料金」ページを参照してください。へのメトリクスの送信を停止するには CloudWatch、メトリクス属性 を削除します。

でメトリクスを表示するには CloudWatch、メトリクス のグラフ化にある手順を完了します。グラフ化できる最小期間は 15 分です。検索語には、メトリクス属性を作成したときにメトリクスに付けた名前を指定します。

以下は、メトリクスが にどのように表示されるかの例です CloudWatch。このメトリクスには、2 人の異なるレコメンダーの 15 分ごとのクリックスルー率が表示されます。



メトリクスをAmazon S3 に発行する

Amazon S3 にメトリクスを発行するには、メトリクス属性で Amazon S3 バケットへのパスを指定します。次に、データセットのインポートジョブを作成する際に、レポートを Amazon S3 に発行します。

ジョブが完了すると、Amazon S3 バケット内のメトリクスを見つけることができます。メトリクスを発行するたびに、Amazon Personalize は Amazon S3 バケットに新しいファイルを作成します。ファイル名には、次のようなインポート方法と日付が含まれます。

`AggregatedAttributionMetrics - ImportMethod - Timestamp.csv`

次に、メトリクスレポートの CSV ファイルの最初の数行がどのように表示されるかの例を示します。この例のメトリクスは、2 人の異なるレコメンダーによる 15 分間隔での合計クリック数を報告しています。各レコメンデーションは、EVENT_ATTRIBUTION_SOURCE 列の Amazon リソースネーム (ARN) によって特定されます。

```
METRIC_NAME,EVENT_TYPE,VALUE,MATH_FUNCTION,EVENT_ATTRIBUTION_SOURCE,TIMESTAMP
COUNTWATCHES,WATCH,12.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender1Name,1666925124
COUNTWATCHES,WATCH,112.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender2Name,1666924224
COUNTWATCHES,WATCH,10.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender1Name,1666924224
COUNTWATCHES,WATCH,254.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender2Name,1666922424
COUNTWATCHES,WATCH,112.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender1Name,1666922424
COUNTWATCHES,WATCH,100.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender2Name,1666922424
```

```
......
.....
```

バルクデータのメトリクスを Amazon S3 (コンソール) に発行する

Amazon Personalize コンソールで Amazon S3 バケットにメトリクスを発行するには、データセットのインポートジョブを作成し、[イベントメトリクスを S3 に公開する] で [このインポートジョブのメトリクスを公開] を選択します。

step-by-step 手順については、「」を参照してくださいバルクレコードのインポート (コンソール)。

バルクデータのメトリクスを Amazon S3 に発行する (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して Amazon S3 バケットにメトリクスを発行するには、次のコードを使用してデータセットのインポートジョブを作成し、publishAttributionMetricsToS3フラグを指定します。特定のジョブのメトリクスを発行しない場合は、フラグを省略します。各パラメータについては、「CreateDatasetImportJob」を参照してください。

```
aws personalize create-dataset-import-job \
--job-name dataset import job name \
--dataset-arn dataset arn \
--data-source dataLocation=s3://bucketname/filename \
--role-arn roleArn \
--import-mode INCREMENTAL \
--publish-attribution-metrics-to-s3
```

Amazon S3 へのバルクデータのメトリクスの発行 (AWS SDKs)

SDK を使用して Amazon S3 バケットにメトリクスを発行するには、データセットのインポートジョブを作成し、 publishAttributionMetricsToS3 を true に設定します。 AWS SDKs 各パラメータについては、「CreateDatasetImportJob」を参照してください。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_import_job(
    jobName = 'YourImportJob',
```

```
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn',
    importMode = 'INCREMENTAL',
    publishAttributionMetricsToS3 = True
)

dsij_arn = response['datasetImportJobArn']

print ('Dataset Import Job arn: ' + dsij_arn)

description = personalize.describe_dataset_import_job(
    datasetImportJobArn = dsij_arn)['datasetImportJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['datasetImportJobArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
 personalizeClient,
                                                        String jobName,
                                                        String datasetArn,
                                                        String s3BucketPath,
                                                        String roleArn,
                                                        ImportMode importMode,
                                                        boolean publishToS3) {

  long waitInMilliseconds = 60 * 1000;
  String status;
  String datasetImportJobArn;

  try {
      DataSource importDataSource = DataSource.builder()
              .dataLocation(s3BucketPath)
              .build();

      CreateDatasetImportJobRequest createDatasetImportJobRequest =
 CreateDatasetImportJobRequest.builder()
              .datasetArn(datasetArn)
              .dataSource(importDataSource)
              .jobName(jobName)
```

```
                .roleArn(roleArn)
                .importMode(importMode)
                .publishAttributionMetricsToS3(publishToS3)
                .build();

        datasetImportJobArn =
    personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                .datasetImportJobArn();

        DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
    DescribeDatasetImportJobRequest.builder()
                .datasetImportJobArn(datasetImportJobArn)
                .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetImportJob datasetImportJob = personalizeClient
                    .describeDatasetImportJob(describeDatasetImportJobRequest)
                    .datasetImportJob();

            status = datasetImportJob.status();
            System.out.println("Dataset import job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return datasetImportJobArn;

      } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
      }
      return "";
    }
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateDatasetImportJobCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: {
    dataLocation: 's3://<name of your S3 bucket>/<folderName>/<CSVfilename>.csv'  /*
 required */
  },
  jobName: 'NAME',                        /* required */
  roleArn: 'ROLE_ARN',                    /* required */
  importMode: "FULL",                     /* optional, default is FULL */
  publishAttributionMetricsToS3: true     /* set to true to publish metrics to
 Amazon S3 bucket */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

A/B テストを使用したレコメンデーションの影響の測定

A/B テストを行うには、複数のバリエーションを使用して実験を行い、結果を比較します。Amazon Personalize レコメンデーションを使用して A/B テストを実行するには、さまざまなユーザーグルー

プにさまざまなタイプのレコメンデーションを表示し、結果を比較します。A/B テストを使用すると、さまざまなレコメンデーション戦略を比較および評価し、レコメンデーションの影響を測定できます。

例えば、A/B テストを使用して、Amazon Personalize のレコメンデーションがクリック率を高めるかどうかを確認できます。このシナリオをテストするために、注目の製品など、パーソナライズされていないレコメンデーションをあるユーザーグループを表示する場合があります。また、Amazon Personalize によって生成されたパーソナライズされたレコメンデーションを別のグループに表示する場合があります。顧客が商品を操作すると、その結果を記録して、どの戦略が最もクリックスルー率が高かったかを確認できます。

Amazon Personalize レコメンデーションを使用して A/B テストを実行するためのワークフローは次のとおりです。

1. 実験の計画 — 定量化可能な仮説を定義し、ビジネス目標を特定して、実験のバリエーションを定義して実験の期間を決定します。
2. ユーザーの分割 — ユーザーを 2 つ以上のグループに分割し、コントロールグループと 1 つ以上の実験グループに分けます。
3. 実験の実行 — 実験グループのユーザーに変更されたレコメンデーションを表示します。コントロールグループのユーザーにはレコメンデーションを変更せずに表示します。レコメンデーションの操作を記録し、結果を追跡します。
4. 結果の評価 — 実験結果を分析して、変更によって実験グループに統計的に有意な差が生じたかどうかを判断します。

Amazon CloudWatch Evidently を使用して、Amazon Personalize のレコメンデーションの A/B テストを実行できます。CloudWatch Evidently を使用すると、実験を定義し、主要業績評価指標 (KPI) を追跡して、レコメンデーションリクエストのトラフィックを関連する Amazon Personalize リソースにルーティングし、テスト結果を評価できます。詳細については、「CloudWatch Evidently を使用した A/B テスト」を参照してください。

トピック

- A/B テストのベストプラクティス
- CloudWatch Evidently を使用した A/B テスト

A/B テストのベストプラクティス

以下のベストプラクティスを使用し、Amazon Personalize のレコメンデーションの A/B テストを設計および維持に役立てます。

- 定量化可能なビジネス目標を特定します。比較するさまざまなレコメンデーションがこのビジネス目標と一致していて、異なる目標や定量化できない目標に関連していないことを確認します。
- ビジネス目標に合致する定量化可能な仮説を定義します。例えば、独自のカスタムメイドのコンテンツをプロモーションで、これらの商品からのクリック数が 20% 増えると予測する場合があります。その仮説によって、実験グループに対して行う変更が決まります。
- 仮説に関連する主要業績評価指標 (KPI) を定義します。KPI を使用して実験の結果を測定します。これらは次のとおりになります。
  - クリックスルー率
  - 総再生時間
  - 合計料金
- 実験に参加するユーザーの総数が、仮説によっては統計的に有意な結果に達するのに十分な数であることを確認します。
- 実験を開始する前に、トラフィック分割戦略を定義します。実験の実行中に、トラフィックの分割を変更しないでください。
- 実験に関連する変更 (モデルなど) を除いて、実験グループとコントロールグループの両方でアプリケーションまたはウェブサイトのユーザーエクスペリエンスを同じにします。UI やレイテンシーなどのユーザーエクスペリエンスのばらつきは、誤解を招く結果につながる可能性があります。
- 祝日、現行のマーケティングキャンペーン、ブラウザの制限などの外部要因を制御します。これらの外部要因は、誤解を招く結果につながる可能性があります。
- 仮説やビジネス要件に直接関係する場合を除き、Amazon Personalize のレコメンデーションを変更しないでください。フィルターの適用や順序の手動による変更などの変更は、誤解を招く結果につながる可能性があります。
- 結果を評価するときは、結論を出す前に結果が統計的に有意であることを確認します。業界標準は有意水準 5% です。統計的有意性の詳細については、「統計的有意性についての再確認」を参照してください。

CloudWatch Evidently を使用した A/B テスト

レコメンダーを作成するか、キャンペーンを含むカスタムソリューションバージョンをデプロイしたら、Amazon Personalize レコメンデーションと Amazon CloudWatch Evidently を使用して A/B テストを実行できます。以下の動画では、CloudWatch Evidently を使用して Amazon Personalize のレコメンデーションに基づく A/B テストを実行するプロセスについて説明しています。手順については、「CloudWatch Evidently で A/B テストを実行する」を参照してください。

Amazon Personalize と CloudWatch Evidently を使用して AB テストを実行する

トピック

- CloudWatch Evidently で A/B テストを実行する
- サンプル実装

CloudWatch Evidently で A/B テストを実行する

Amazon Personalize と Amazon CloudWatch Evidently を使用して A/B テストを実行するには、CloudWatch Evidently プロジェクトを作成し、機能とそのバリエーションを定義して、実験をサポートするようにアプリケーションを更新してから、実験を作成して実行します。実験が実行されると、結果が CloudWatch Evidently で表示されます。

Amazon Personalize と CloudWatch Evidently で A/B テストを実行するには

1. 新しい CloudWatch Evidently プロジェクトを作成します。プロジェクトは、CloudWatch リソースを論理的にグループ化したものです。プロジェクト内では、テストまたは起動をするバリエーションがある機能を作成します。詳細な手順については、Amazon CloudWatch ユーザーガイドの「新しいプロジェクトの作成」を参照してください。
2. プロジェクトに機能を追加し、そのバリエーションを定義します。この実験では、機能はクリック率など、テストしたいレコメンデーションのシナリオを表す必要があります。

   機能を追加する際は、シナリオのさまざまなバリエーションを Amazon Personalize レコメンダーまたはカスタムキャンペーンにマッピングするための識別子を指定します。バリエーションごとに、文字列などのバリエーションタイプを指定し、バリエーションに名前を付けて値を指定します。

   実験を実行すると、アプリケーションはバリエーションの値を使用して、レコメンデーションに使用する Amazon Personalize リソースを決定します。例えば、2 つの VIDEO_ON_DEMAND レコメンダーをテストする場合、1 つはユーザーのおすすめユースケース用に作成し、もう 1 つ

は現在のトレンドユースケース用に作成して、次の JSON を各バリエーションの値として設定できます。

```
{"type":"top-picks-recommendations","arn":"arn:aws:personalize:us-west-2:<acct-id>:recommender/top-picks-recommender"}
```

```
{"type":"trending-recommendations","arn":"arn:aws:personalize:us-west-2:<acct-id>:recommender/trending-now-recommender"}
```

アプリケーションがその識別子を使用して関連するリソースを識別できるのであれば、どのような識別子を指定してもかまいません。例えば、レコメンダーまたはキャンペーンの名前のみを指定し、アプリケーションのリソースの Amazon リソースネーム (ARN) を作成できます。

機能を追加する詳細な手順については、Amazon CloudWatch ユーザーガイドの「プロジェクトに機能を追加する」を参照してください。

3. 実験をサポートするようにアプリケーションを更新します。

- 機能評価 — CloudWatch Evidently `EvaluateFeature` API オペレーションを使用して、各ユーザーセッションにバリエーションを割り当てます。`EvaluateFeature`レスポンスには、前のステップで指定したバリエーション値が含まれます。この場合は、レコメンダーのタイプを持つ JSON オブジェクトで、レコメンダーの ARN です。レコメンデーションリクエストコードを更新して、このリソースからレコメンデーションを取得します。

  機能の評価については、Amazon CloudWatch ユーザーガイドの「EvaluateFeature の使用」を参照してください。

- 結果を記録する — ユーザーによるレコメンデーションと操作の結果を追跡するコードをアプリケーションに追加します。

  CloudWatch Evidently での実験のメトリクスを追跡するには、CloudWatch Evidently `PutProjectEvents` API オペレーションを使用して各ユーザーの結果を記録します。例えば、実験に参加しているユーザーがレコメンデーションをクリックすると、このイベントの詳細が CloudWatch Evidently に送信されます。

  CloudWatch Evidently へのイベントの送信については、Amazon CloudWatch ユーザーガイドの「PutProjectEvents の使用」を参照してください。

  Amazon Personalize レコメンデーションの関連性を高めるために、Amazon Personalize `PutEvents` API オペレーションを使用して結果イベントを記録できます。ドメインのユース

ケースまたはカスタムレシピでレコメンデーションのリアルタイム更新がサポートされている場合、Amazon Personalize はユーザーの最新のアクティビティから学習し、ユーザーがアプリケーションを使用するごとにレコメンデーションを更新します。更新をサポートしていない場合、Amazon Personalize は次回のモデル再トレーニング時にこのデータを使用するため、レコメンデーションに影響します。

Amazon Personalize へのイベントのストリーミングについては、「イベントの記録」を参照してください。

4. 実験を作成および開始します。実験を作成する際には、以下を指定できます。

   - 機能 — 実験でテストする機能を選択します。
   - 対象者 — 参加するユーザーの数と、機能のバリエーション間でトラフィックを分割する方法を設定します。
   - 指標 — 実験の成功を左右する指標を指定します。例えば、クリック数があります。

   実験の作成が完了したら、その期間を指定して実験を開始します。CloudWatch Evidently で実験を作成して開始する詳細な手順については、Amazon CloudWatch ユーザーガイド の「実験を作成する」を参照してください。

5. 実験を実行すると、CloudWatch Evidently 実験ダッシュボードに結果が表示されます。実験結果の表示については、Amazon CloudWatch ユーザーガイドの「ダッシュボードに実験結果を表示する」を参照してください。

サンプル実装

以下のサンプル実装は、CloudWatch Evidently を使用して A/B テストを実装する方法を示しています。

- A/B テストを実装するためのソースコードを含むリアルタイム API の完全な例については、AWS サンプル GitHub リポジトリ の「リアルタイム パーソナライゼーション API」を参照してください。
- パーソナライゼーションと A/B テストに関するワークショップを含むサンプル小売ウェブアプリケーションについては、AWS サンプル GitHub リポジトリの「小売デモ店舗」を参照してください。CloudWatch Evidently と小売デモ店舗を使用して A/B テストを作成する方法を説明するノートブックについては、「小売デモ店舗実験ワークショップ - CloudWatch Evidently」を参照してください。

- CloudWatch Evidently で A/B テストを使用する方法を説明するチュートリアルとサンプル React アプリケーションについては、Amazon CloudWatch ユーザーガイド の「チュートリアル: Evidently のサンプルアプリケーションを使用した A/B テスト」を参照してください。

から検索結果をパーソナライズする OpenSearch

Amazon Personalize を使用して、オープンソース OpenSearch または Amazon OpenSearch Service の結果をユーザー向けにパーソナライズできます。

OpenSearch は、Apache 2.0 ライセンスに基づくセルフマネージド型のオープンソース検索サービスです。Amazon OpenSearch Service は、 AWS クラウドでの OpenSearch リソースのデプロイ、運用、スケーリングを支援するマネージドサービスです。Amazon OpenSearch Service を使用すると、 は結果 OpenSearch を取得してランク付けします。

クエリ結果をランク付けするときに、BM-25 と呼ばれる確率的ランキングフレームワーク OpenSearch を使用して関連性スコアを計算します。固有のキーワードがドキュメントに頻繁に出現する場合、BM-25 はそのドキュメントに高い関連性スコアを割り当てます。 OpenSearch ランク付けでは、クリックスルーデータなどのユーザーの動作は考慮されません。

で Amazon Personalize を使用すると OpenSearch、Amazon Personalize はユーザーの過去の動作、項目に関するメタデータ、およびユーザーに関するメタデータに基づいて OpenSearch 結果を再ランク付けします。 OpenSearch その後、検索レスポンスをアプリケーションに返す前に再ランク付けが組み込まれます。Amazon Personalize を OpenSearch 結果に適用するときに、Amazon Personalize のランキングにどの程度の重み OpenSearch が与えられるかを制御します。

この再ランク付けにより、結果がより魅力的になり、ユーザーの興味に関連性の高いものになります。これにより、アプリケーションのクリックスルー率とコンバージョン率が向上する可能性があります。パーソナライズした検索によって eコマースアプリケーションの結果がどのように改善されるかを説明するユースケース例については、「ユースケースの例」を参照してください。

OpenSearch 結果のパーソナライズを開始する前に、「」に記載されている要件を確認してくださいガイドラインと要件。

トピック

- ユースケースの例
- パーソナライズされた検索ワークフロー
- Amazon Personalize Search Ranking プラグインの仕組み
- 追加情報
- ガイドラインと要件
- プラグインのセットアップ OpenSearch とインストール
- プラグインの設定

- OpenSearch クエリへのプラグインの適用
- OpenSearch 結果をプラグインの結果と比較する
- プラグインの監視

ユースケースの例

Amazon Personalize を使用して OpenSearch 結果を再ランク付けすると、検索結果がユーザーにより関連しやすくなります。例えば、自動車を販売する eコマースアプリケーションがあるとします。ユーザーがトヨタ車のクエリを入力し、結果をパーソナライズしない場合、 OpenSearch はデータ内のキーワードに基づいてトヨタ製の自動車のリストを返します。このリストは、すべてのユーザーに対して同じ順序でランク付けされます。

ただし、Amazon Personalize を使用して結果をパーソナライズする場合、 は、クリックなどの動作に基づいて、特定のユーザーにとって関連性がある順にこれらの車 OpenSearch を再ランク付けします。ユーザーがクリックする可能性が最も高い車が最初にランク付けされます。

OpenSearch 結果をパーソナライズするときは、Amazon Personalize からランク付けする重 OpenSearch み (強調) を制御します。この例を続けると、ユーザーが特定の年の特定のタイプの自動車 (2008 年式トヨタプリウスなど) を検索する場合、 の元のランキングにさらに重点を置きたい場合があります OpenSearch。

ただし、結果が広範囲に及ぶ一般的なクエリ (トヨタの全車両を検索する場合など) では、パーソナライゼーションに重点を置くことがあります。これにより、リスト上部にある車が、特定のユーザーにとってより関連性の高いものになります。

パーソナライズされた検索ワークフロー

OpenSearch 結果をパーソナライズするには、以下を実行します。

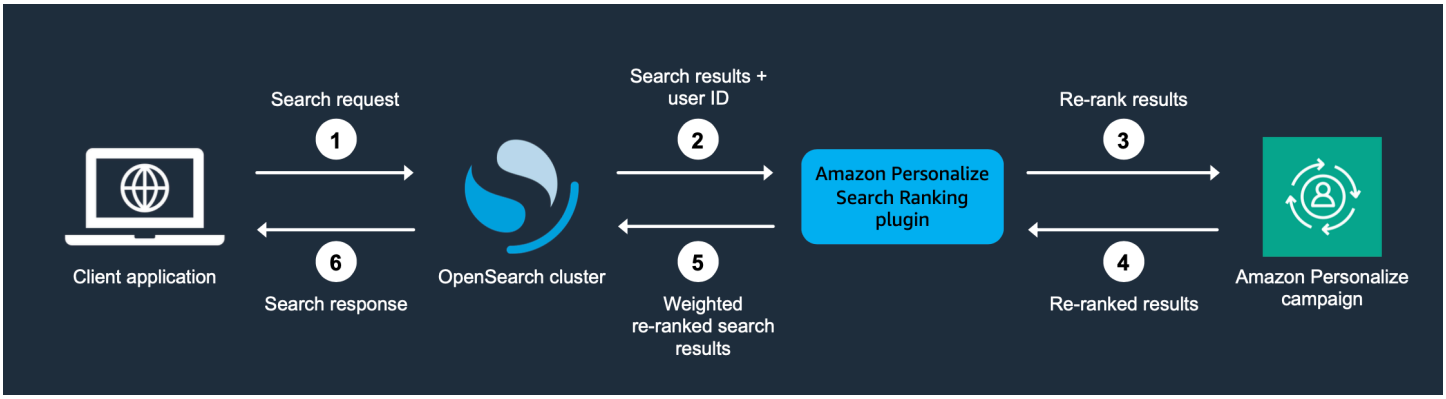
1. Amazon Personalize のセットアップ — まだセットアップをしていない場合は、Amazon Personalize の設定 のステップを実行して認証情報を設定し、Amazon Personalize のアクセス権限を設定します。 OpenSearch 結果をパーソナライズするために AWS SDKsを設定する必要はありません。
2. Amazon Personalize ワークフローの完了 — Amazon Personalize ワークフローを完了すると、データのインポート、Personalized-Ranking レシピを使ったソリューションの作成、カスタムソリューションバージョンのトレーニング、キャンペーンへのデプロイが可能になります。使用できるのは Personalized-Ranking レシピだけです。アイテムインタラクションデータセットを作成

する必要があります。ユーザーデータセットとアイテムデータセットはオプションです。詳細については、「Amazon Personalize のワークフロー」を参照してください。

3. Amazon Personalize Search Ranking プラグインをセットアップ OpenSearch してインストールする – まだセットアップしていない場合は、 OpenSearch サービスドメインまたはオープンソース OpenSearch クラスターをセットアップします。次に、Amazon Personalize Search Ranking プラグインをインストールします。このプラグインは Amazon Personalize との通信と結果の再ランク付けを処理します。詳細については、「プラグインのセットアップ OpenSearch とインストール」を参照してください。
4. Amazon Personalize Search Ranking プラグインの設定 — プラグインを設定するには、検索パイプラインを作成します。検索パイプラインは、リクエストプロセッサとレスポンスプロセッサのセットです。プラグインのパイプラインを作成するときは、`personalized_search_ranking` レスポンスプロセッサで Amazon Personalize リソースを指定します。また、プラグインが結果を再ランク付けするときに Amazon Personalize からの結果に与える重みも設定します。詳細については、「プラグインの設定」を参照してください。
5. Amazon Personalize Search Ranking プラグインを OpenSearch クエリに適用する – OpenSearch インデックス のすべてのクエリとレスポンスに Amazon Personalize Search Ranking プラグインを適用できます。プラグインを個々の OpenSearch クエリに適用することもできます。詳細については、「 OpenSearch クエリへのプラグインの適用」を参照してください。
6. 結果の比較 — Amazon Personalize Search Ranking プラグインは、 OpenSearch クエリレスポンスの検索結果を再ランク付けします。Amazon Personalize のランキングと のランキングの両方を考慮します OpenSearch。結果がどのように再ランク付けされるかを理解するには、パーソナライゼーションを使用するクエリと使用しないクエリの結果を比較できます。詳細については、「 OpenSearch 結果をプラグインの結果と比較する」を参照してください。
7. Amazon Personalize Search Ranking プラグインの監視 — Amazon Personalize Search Ranking プラグインを検索クエリに適用すると、検索パイプラインのメトリクスを取得してプラグインを監視できます。詳細については、「プラグインの監視」を参照してください。

Amazon Personalize Search Ranking プラグインの仕組み

次の図は、Amazon Personalize Search Ranking プラグインの動作を示しています。



1. 顧客のクエリを OpenSearch サービスドメインまたはオープンソース OpenSearch クラスターに送信します。
2. OpenSearch は、クエリレスポンス (クエリに関連する項目のリスト) とユーザーの ID を Amazon Personalize Search Ranking プラグインに送信します。
3. プラグインは、レスポンス内のアイテムとユーザーを Amazon Personalize キャンペーンに送信してランキングを求めます。検索パイプライン内のレシピとキャンペーンの Amazon リソースネーム (ARN) の値を使用して、ユーザーごとにカスタマイズされたランキングを取得します。レコメンデーションには GetPersonalizedRanking API オペレーションを使用します。リクエストでは、クエリを実行しているユーザーの userId と、 の OpenSearch クエリから返された項目を渡しますinputList。
4. Amazon Personalize は、再ランク付けされた結果をプラグインに返します。
5. プラグインは検索結果を再配置して、 OpenSearch サービスドメインまたはオープンソース OpenSearch クラスターに返します。Amazon Personalize キャンペーンからの反応と、設定時に指定したパーソナライゼーションの強調に基づいて、結果が再ランク付けされます。
6. オープンソース OpenSearch クラスターまたは OpenSearch サービスドメインは、アプリケーションに最終結果を返します。

追加情報

以下のリソースは、 の使用に関する追加情報を提供します OpenSearch。

- オープンソース の開始方法については OpenSearch、「クイックスタート」を参照してください。
- OpenSearch サービスの開始方法については、Amazon OpenSearch Service デベロッパーガイドの「Amazon OpenSearch Service の開始方法」を参照してください。

- Amazon Personalize の Personalized-Ranking レシピについては、「Personalized-Ranking レシピ」を参照してください。

ガイドラインと要件

このセクションには、Amazon Personalize Search Ranking プラグインを使用するための要件が含まれています。また、Amazon OpenSearch Service またはオープンソース のアクセス許可を設定する方法についても説明します OpenSearch。

トピック

- プラグインの要件
- Amazon OpenSearch Service のアクセス許可の設定
- オープンソース OpenSearch のアクセス許可の設定

プラグインの要件

から結果のパーソナライズを開始する前に OpenSearch、Amazon Personalize Search Ranking プラグインに関する以下のガイドラインと要件に注意してください。

- OpenSearch バージョン 2.9.0 以降を使用する必要があります。Amazon OpenSearch Service を使用する場合、ドメインはバージョン 2.9 以降を使用する必要があります。
- お持ちでない場合は、アクセス許可のセットアップ の手順を完了して Amazon Personalize にアクセスする許可をユーザーに与え、Amazon Personalize のリソースにアクセスする許可を Amazon Personalize に与えます。
- OpenSearch サービスドメインまたはオープンソース OpenSearch クラスターから Amazon Personalize リソースにアクセスできる必要があります。
  - OpenSearch サービスドメインへのアクセス許可の付与については、「」を参照してくださいAmazon OpenSearch Service のアクセス許可の設定。
  - OpenSearch クラスターへのアクセス許可の付与については、「」を参照してくださいオープンソース OpenSearch のアクセス許可の設定。
- Amazon Personalize のカスタムリソースのみを使用できます。ドメインデータセットグループを作成した場合でも、カスタムリソースを追加できます。
- カスタムレシピである Personalized-Ranking のみを使用できます。このレシピについての詳細は、「Personalized-Ranking レシピ」を参照してください。

- Amazon Personalize でアイテムインタラクションデータセットを作成する必要があります。アイテムデータセットとユーザーデータセットはオプションです。
- Amazon Personalize Search Ranking プラグインを使用している場合、Amazon Personalize フィルターを適用することはできません。
- デフォルトでは、プラグインは、 のインデックス付きドキュメント_idの が Amazon Personalize データ内の itemId OpenSearch と一致することを前提としています。 OpenSearch データが Amazon Personalize itemIds に対応する別のフィールドを使用している場合は、プラグインを設定するときにフィールドの名前を指定する必要があります。
- クエリを実行するユーザーに使用するユーザー userId は、Amazon Personalize にインポートするデータのuserId と一致する必要があります。
- プラグインは、 の検索結果の上位 500 件のみを再ランク付けします OpenSearch。残りのアイテムは再ランク付けされず、リスト末尾に表示されます。

Amazon OpenSearch Service のアクセス許可の設定

Amazon OpenSearch Service を使用する場合は、 OpenSearch サービスドメインから Amazon Personalize リソースにアクセスできる必要があります。

アクセス許可を設定するには

1. リソースが同じアカウントにあるか異なるアカウントにあるかに応じて、リソースへのアクセス許可を持つ 1 つ以上の IAM サービスロールを作成します。
   - OpenSearch サービスと Amazon Personalize リソースが同じアカウントにある場合は、OpenSearch サービスの IAM サービスロールを作成し、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するアクセス許可を付与します。詳細については、「リソースが同じアカウントにある場合のアクセス許可の設定」を参照してください。
   - OpenSearch サービスと Amazon Personalize リソースが別々のアカウントにある場合は、2 つの IAM サービスロールを作成します。 OpenSearch サービスリソースを使用してアカウントに 1 つ作成し、 OpenSearch サービスリソースへのアクセスを許可します。また、Amazon Personalize リソースを使用してアカウントに 1 つを作成し、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するアクセス許可を付与します。詳細については、「リソースが異なるアカウントにある場合のアクセス許可の設定」を参照してください。
2. Service 用に作成した IAM OpenSearch サービスロールの OpenSearch Service ドメインアクセスPassRole許可にアクセスするユーザーまたはロールに付与します。詳細については、「Amazon OpenSearch Service ドメインセキュリティの設定」を参照してください。

トピック

- リソースが同じアカウントにある場合のアクセス許可の設定
- リソースが異なるアカウントにある場合のアクセス許可の設定
- Amazon OpenSearch Service ドメインセキュリティの設定

リソースが同じアカウントにある場合のアクセス許可の設定

OpenSearch サービスと Amazon Personalize のリソースが同じアカウントにある場合は、OpenSearch サービスの IAM サービスロールを作成する必要があります。このロールには、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するためのアクセス許可が必要です。Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するためのアクセス許可を OpenSearch サービスロールに付与するには、以下が必要です。

- ロールの信頼ポリシーは、 OpenSearch サービスにアクセス`AssumeRole`許可を付与する必要があります。信頼ポリシーの例については、「信頼ポリシーの例」を参照してください。
- ロールには、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得する許可が必要です。ポリシーの例については、「アクセス許可ポリシーの例」を参照してください。

IAM ロールの作成方法の詳細については、「IAM ユーザーガイド」の「IAM ロールを作成する」を参照してください。ロールへの IAM ポリシーのアタッチについては、「IAM ユーザーガイド」の「IAM ID のアクセス許可の追加および削除」を参照してください。

OpenSearch サービスの IAM サービスロールを作成したら、サービスサービスロールの OpenSearch サービスドメインアクセス`PassRole`許可にアクセスするユーザーまたは OpenSearch ロールに付与する必要があります。詳細については、「Amazon OpenSearch Service ドメインセキュリティの設定」を参照してください。

トピック

- 信頼ポリシーの例
- アクセス許可ポリシーの例

信頼ポリシーの例

次の信頼ポリシーの例では、 OpenSearch サービスにアクセス`AssumeRole`許可を付与します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "",
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Principal": {
            "Service": [
                "es.amazonaws.com"
            ]
        }
    }]
}
```

アクセス許可ポリシーの例

次のポリシー例では、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するための最低限のアクセス許可をロールに付与しています。`Campaign ARN` には、Amazon Personalize キャンペーンの Amazon リソースネーム (ARN) を指定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:GetPersonalizedRanking"
            ],
            "Resource": "Campaign ARN"
        }
    ]
}
```

リソースが異なるアカウントにある場合のアクセス許可の設定

OpenSearch サービスと Amazon Personalize リソースが別々のアカウントにある場合は、各アカウントに IAM ロールを作成し、そのロールにアカウントのリソースへのアクセスを許可します。

複数のアカウントのアクセス許可を設定するには

1. Amazon Personalize キャンペーンが存在するアカウントで、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するアクセス許可を持つ IAM ロールを作成します。プラグインを設定するときは、`personalized_search_ranking`レスポンスプロセッサ

の external_account_iam_role_arnパラメータでこのロールの ARN を指定します。詳細については、「プラグインの設定」を参照してください。

ポリシーの例については、「アクセス許可ポリシーの例」を参照してください。

2. OpenSearch サービスドメインが存在するアカウントで、 OpenSearch サービスAssumeRoleアクセス許可を付与する信頼ポリシーを持つロールを作成します。プラグインを設定するときは、personalized_search_rankingレスポンスプロセッサの iam_role_arnパラメータでこのロールの ARN を指定します。詳細については、「プラグインの設定」を参照してください。

   信頼ポリシーの例については、「信頼ポリシーの例」を参照してください。

3. 各ロールを変更して、他のロールのAssumeRoleアクセス許可を付与します。例えば、Amazon Personalize リソースにアクセスできるロールの場合、その IAM ポリシーは、 OpenSearch サービスドメインを持つアカウントのロールに、次のようにロールの継承アクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "",
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::<Account number for role with access to
 OpenSearch Service domain>:role/roleName"

    }]
}
```

4. OpenSearch サービスドメインが存在するアカウントで、先ほど作成した OpenSearch サービスサービスロールのサービスドメインにアクセスしているユーザーまたは OpenSearch ロールに、アクセスPassRole許可を付与します。詳細については、「Amazon OpenSearch Service ドメインセキュリティの設定」を参照してください。

Amazon OpenSearch Service ドメインセキュリティの設定

Service でプラグインを使用するには OpenSearch 、ドメインにアクセスするユーザーまたはロールに、先ほど作成した OpenSearch Service の IAM サービスロールに対するPassRoleアクセス許可が必要です。また、ユーザーまたはロールには、es:ESHttpGet および es:ESHttpPut アクションを実行するためのアクセス許可が必要です。

OpenSearch サービスへのアクセスの設定の詳細については、Amazon OpenSearch Service デベロッパーガイドの「Amazon Service のセキュリティ」を参照してください。 OpenSearch IAM ポリシーの例については、「 OpenSearch サービスユーザーまたはロールのポリシー例」を参照してください。

OpenSearch サービスユーザーまたはロールのポリシー例

次の IAM ポリシーの例では、 で OpenSearch サービス用に作成した IAM サービスロールのアクセスPassRole許可をユーザーまたはロールに付与しますリソースが同じアカウントにある場合のアクセス許可の設定。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "OpenSearch Service role ARN"
        }
    ]
}
```

次の IAM ポリシーは、 OpenSearch サービスでパイプラインを作成し、クエリを検索するための最小限のアクセス許可を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "es:ESHttpGet",
                "es:ESHttpPut"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:ResourceTag/environment": [
                        "production"
                    ]
                }
```

```
            }
        }
    ]
}
```

オープンソース OpenSearch のアクセス許可の設定

オープンソース を使用する場合は OpenSearch、オープン検索クラスターから Amazon Personalize リソースにアクセスできる必要があります。アクセス許可を付与するには、次の手順を実行します。

- ゼロ OpenSearch からセットアップする場合は、クイックスタート bash スクリプトを使用して Docker コンテナで OpenSearch クラスターを実行できます。このスクリプトは、 AWS プロファイルのデフォルトの認証情報を使用します。スクリプトを実行するときに、代替プロファイルを指定できます。

  これらの認証情報は、Amazon Personalize キャンペーンのアクションを実行する GetPersonalizedRanking権限を持つユーザーまたはロールに関連付ける必要があります。IAM ポリシーの例については、「IAM ポリシーの例」を参照してください。または、認証情報にはこれらのアクセス許可を持つロールを引き受けるアクセス許可が必要です。Amazon Personalize Search Ranking プラグインのパイプラインを作成するときに、このロールの Amazon リソースネーム (ARN) を指定できます。
- クイックスタート bash スクリプト を使用しない場合は、 OpenSearch キーストアに認証情報を手動で追加できます。これらの認証情報は、Amazon Personalize キャンペーンの GetPersonalizedRanking アクションを実行する権限を持つユーザーまたはロールに対応している必要があります。

  AWS 認証情報を手動で OpenSearch キーストアに追加するには、 OpenSearch クラスターが実行されている (Docker コンテナなど) 次のコマンドを実行します。次に、各認証情報を入力します。セッショントークンを使用しない場合は、コマンドの最後の行を省略できます。

  ```
  opensearch-keystore add \
  personalized_search_ranking.aws.access_key \
  personalized_search_ranking.aws.secret_key \
  personalized_search_ranking.aws.session_token
  ```
- Amazon EC2 インスタンスで OpenSearch クラスターを実行する場合は、IAM インスタンスプロファイルを使用してアクセス許可を付与できます。ロールにアタッチされたポリシーは、Amazon Personalize キャンペーンの GetPersonalizedRanking アクションを実行するアクセス許可を付与

する必要があります。また、ロールを引き受けるためのアクセス許可を Amazon EC2 に付与する必要があります。

Amazon EC2 インスタンスプロファイルについては、「インスタンスプロファイルの使用」を参照してください。ポリシーの例については、「IAM ポリシーの例」を参照してください。

IAM ポリシーの例

次のポリシー例では、Amazon Personalize キャンペーンからパーソナライズされたランキングを取得するための最低限のアクセス許可をユーザーまたはロールに付与します。`Campaign ARN` には、Amazon Personalize キャンペーンの Amazon リソースネーム (ARN) を指定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:GetPersonalizedRanking"
            ],
            "Resource": "Campaign ARN"
        }
    ]
}
```

さらに、Amazon EC2 インスタンスで OpenSearch クラスターを実行し、IAM インスタンスプロファイルでアクセス許可を付与する場合、ロールの信頼ポリシーは、次のように Amazon EC2 アクセス`AssumeRole`許可を付与する必要があります。Amazon EC2 インスタンスプロファイルについては、「インスタンスプロファイルの使用」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

プラグインのセットアップ OpenSearch とインストール

Amazon Personalize Search Ranking プラグインは、 OpenSearch サービスドメインまたはオープンソース OpenSearch クラスターからの Amazon Personalize との通信を処理します。また、結果の再ランク付けを処理します。へのアクセス方法に応じて OpenSearch、プラグインを次のようにセットアップ OpenSearch してインストールします。

- Amazon OpenSearch Service を使用する場合は、 OpenSearch Service でドメインを作成し、データを取り込んで、プラグインをインストール OpenSearch して を設定します。
- オープンソースの を使用する場合は OpenSearch、 OpenSearch クラスターを作成し、データを取り込んで、プラグインをインストールします。

トピック

- Amazon OpenSearch Service のセットアップ
- オープンソースのセットアップ OpenSearch

Amazon OpenSearch Service のセットアップ

Amazon Personalize ワークフローを完了し、 に記載されている要件を満たしたらガイドラインと要件、Amazon OpenSearch Service をセットアップし、Amazon Personalize Search Ranking プラグインをインストールする準備が整います。

Amazon OpenSearch Service を設定するには、ドメインを作成し、データを取り込んで、プラグインをインストールします。すでにドメインを作成してデータを取り込んでいる場合は、ステップ 3 に進んでください。

OpenSearch サービスを設定するには

1. まだの場合は、「」のステップを完了Amazon OpenSearch Service のアクセス許可の設定して、 OpenSearch サービスドメインから Amazon Personalize リソースにアクセスできるようにします。
2. まだ作成していない場合は、 OpenSearch サービスドメインを作成します。OpenSearch サービスドメインは、オープンソース OpenSearch クラスターと同義です。ドメインは、指定した

設定、インスタンスタイプ、インスタンスカウント、およびストレージリソースを持つクラスターです。

- テストドメインを設定するための簡潔なチュートリアルについては、「Amazon OpenSearch Service デベロッパーガイド」の「開始方法」セクションの「ステップ 1: Amazon Service ドメインを作成する」を参照してください。 OpenSearch
- 詳細については、「Amazon OpenSearch Service ドメインの作成と管理」を参照してください。

3. まだの場合は、アイテムを OpenSearch サービスに取り込みます。

   - 少量のテストデータを OpenSearch サービスにアップロードするための簡潔なチュートリアルについては、Amazon OpenSearch Service デベロッパーガイドの「開始方法」セクションの「ステップ 2: Amazon Service にデータをアップロードしてインデックスを作成する」を参照してください。 OpenSearch
   - データの取り込みの詳細については、Amazon OpenSearch Service デベロッパーガイドの「Amazon Service でのデータのインデックス作成」を参照してください。 OpenSearch

4. `Amazon_Personalize_Search_Ranking_Plugin` プラグインをドメインに関連付けます。プラグインはプリインストールされており、Amazon S3 からインポートする必要はありません。プラグインは、 OpenSearch サービスパッケージを関連付けるのと同じ方法で関連付けます。

   OpenSearch サービスパッケージの関連付けについては、「Amazon OpenSearch Service のカスタムパッケージ」を参照してください。

ドメインを作成し、データを取り込み、Amazon Personalize Search Ranking プラグインをインストールしたら、プラグインを設定する準備が整います。検索パイプラインを作成し、`personalized_search_ranking` レスポンスプロセッサを指定して設定します。詳細については、「プラグインの設定」を参照してください。

オープンソースのセットアップ OpenSearch

Amazon Personalize ワークフローを完了し、ガイドラインと要件 に記載されている要件を満たしたら、オープンソースをセットアップし、Amazon Personalize Search Ranking プラグインをインストールする準備が整います。

OpenSearch クラスターが既に実行されている場合は、プラグインを手動でインストールできます。クラスターが実行されていない場合は、bash スクリプトを使用して OpenSearch とプラグインを最初からインストールできます。

トピック

- 既存の OpenSearch クラスターにプラグインを手動でインストールする
- クラスターをセットアップし、クイックスタートスクリプトを使用してプラグインをインストールする

既存の OpenSearch クラスターにプラグインを手動でインストールする

OpenSearch クラスターが既にある場合は、 OpenSearch GitHub リポジトリから直接クラスターにプラグインを手動でインストールできます。

プラグインを手動でインストールするには

1. クラスターを起動するには、次のコマンドを使用します OpenSearch 。

   ```
   bin/opensearch
   ```

2. カタログデータをまだクラスターにアップロードしていない場合は、 OpenSearch クラスターにアップロードします。データをアップロードするときは、 OpenSearch インデックスを作成し、フィールドマッピングを定義します。次に、そのインデックスにデータをアップロードします。例については、「サンプルデータを使用してインデックスとフィールドマッピングを作成する」を参照してください。
3. プラグインをインストールするには、次のコマンドを使用します。

   ```
   bin/opensearch-plugin install https://github.com/opensearch-project/search-
   processor/releases/download/2.9.0/opensearch-search-processor-2.9.0.0.zip
   ```

   プラグインのインストールについての詳細は、「プラグインのインストール」を参照してください。

Amazon Personalize Search Ranking プラグインをインストールしたら、設定する準備が整います。プラグインを設定するには、検索パイプラインを作成し、personalized_search_ranking レスポンスプロセッサを指定します。詳細については、「プラグインの設定」を参照してください。

クラスターをセットアップし、クイックスタートスクリプトを使用してプラグインをインストールする

OpenSearch クラスターを作成していない場合は、クイックスタート bash スクリプトを使用してクラスターを作成できます。このスクリプトは、Docker コンテナに OpenSearch クラスターを設定し、デフォルト AWS プロファイルを使用して認証情報を設定し、Amazon Personalize Search Ranking プラグインをインストールします。

OpenSearch クラスターの手動作成については、 OpenSearch ドキュメントの「クイックスタート手順」を参照してください。

クイックスタート Bash スクリプトを使用してプラグインをインストールするには

1. スクリプトを実行する前に、ご使用のオペレーティングシステム用の Docker Desktop をダウンロードしてインストールします。
2. クイックスタート bash スクリプトを からダウンロードします GitHub。
3. ワーキングディレクトリで、次のコマンドを使用してスクリプトを実行します。

   ```
   sh personalized_search_ranking_quickstart.sh
   ```

   このコマンドでは、スクリプトはデフォルト AWS プロファイルの認証情報を使用します。代替プロファイルを指定するには、デフォルトの `--profile` 引数を使用します。

   ```
   sh personalized_search_ranking_quickstart.sh --profile profile-name
   ```

   スクリプトを実行すると、スクリプトによって作成された固有のディレクトリにある README ファイルに、そのスクリプトに関する詳細情報が記載されます。このディレクトリには、スクリプトが使用する Dockerfile ファイルと docker-compose.yml ファイルが格納されます。例: `../opensearch-personalize-intelligent-ranking-docker.1234/README`。
4. カタログデータを OpenSearch クラスターにアップロードします。データをアップロードするときは、 OpenSearch インデックスを作成し、フィールドマッピングを定義します。次に、そのインデックスにデータをアップロードします。例については、「サンプルデータを使用してインデックスとフィールドマッピングを作成する」を参照してください。

Amazon Personalize Search Ranking プラグインをセットアップ OpenSearch してインストールしたら、設定する準備が整います。プラグインを設定するには、検索パイプラインを作成

し、personalized_search_ranking レスポンスプロセッサを指定します。詳細については、「プラグインの設定」を参照してください。

プラグインの設定

Amazon Personalize Search Ranking プラグインをインストールしたら、 OpenSearch 検索パイプラインを作成して設定する準備が整います。

検索パイプラインは、作成した順序で順番に実行されるリクエストプロセッサとレスポンスプロセッサのセットです。プラグインの検索パイプラインを作成するときは、personalized_search_ranking レスポンスプロセッサを指定します。検索パイプラインについて詳しくは、「検索パイプライン」を参照してください。

トピック

- personalized_search_ranking レスポンスプロセッサのフィールド
- Amazon OpenSearch Service を使用したパイプラインの作成
- オープンソースを使用したパイプラインの作成 OpenSearch

personalized_search_ranking レスポンスプロセッサのフィールド

personalized_search_ranking レスポンスプロセッサには、以下のフィールドを指定します。

- campaign_arn (必須) — 結果のパーソナライズに使用する Amazon Personalize キャンペーンの Amazon リソースネーム (ARN) を指定します。
- item_id_field (オプション) – のインデックス付きドキュメントの _idフィールド OpenSearch が Amazon Personalize itemIds に対応していない場合は、 が対応するフィールドの名前を指定します。デフォルトでは、プラグインは _id データが Amazon Personalize データの itemId と一致すると仮定します。
- レシピ (必須) — 使用する Amazon Personalize レシピの名前を指定します。aws-personalized-ranking のみを指定できます。
- 重み (必須) — レスポンスプロセッサが結果を再ランク付けする際にパーソナライゼーションを重視することを指定します。0.0 ~ 1.0 の範囲内の値を指定します。1.0 に近いほど、Amazon Personalize の結果が上位にランクされる可能性が高くなります。を指定した場合0.0、パーソナライゼーションは発生せず、 OpenSearch 優先されます。
- タグ (オプション) — プロセッサの識別子を指定します。

- iam_role_arn (OpenSearch サービスの場合は必須、オープンソースの場合はオプション OpenSearch) – OpenSearch サービスの場合は、Amazon Personalize リソースにアクセスするためのアクセス許可を OpenSearch サービスに設定するときに作成したロールの Amazon リソースネーム (ARN) を指定します。 OpenSearch サービスと Amazon Personalize リソースが異なるアカウントに存在する場合は、 OpenSearch サービスにアクセスAssumeRole許可を付与するロールを指定します。詳細については、「リソースが異なるアカウントにある場合のアクセス許可の設定」を参照してください。

  オープンソース の場合 OpenSearch、複数のロールを使用して組織内の異なるユーザーグループのアクセス許可を制限する場合は、Amazon Personalize へのアクセス許可を持つロールの ARN を指定します。 OpenSearch キーストアで認証情報のみを使用する場合は、このフィールドを AWS 省略できます。
- aws_region (必須) — Amazon Personalize キャンペーンを作成した AWS リージョン。
- ignore_failure (オプション) — プラグインがプロセッサ障害を無視するかどうかを指定します。値には、`true` または `false` を指定します。本番環境では、クエリ応答が中断されないように `true` を指定することをお勧めします。テスト環境では、プラグインが生成するエラーをすべて表示するように `false` を指定できます。
- external_account_iam_role_arn – OpenSearch サービスを使用し、Amazon Personalize と OpenSearch サービスリソースが異なるアカウントに存在する場合は、Amazon Personalize リソースへのアクセス許可を持つロールの ARN を指定します。このロールは、Amazon Personalize リソースと同じアカウントに存在する必要があります。詳細については、「リソースが異なるアカウントにある場合のアクセス許可の設定」を参照してください。

Amazon OpenSearch Service を使用したパイプラインの作成

次の Python コードを使用して、 OpenSearch サービスドメインのpersonalized_search_rankingレスポンスプロセッサで検索パイプラインを作成できます。`domain endpoint` をドメインのエンドポイント URL に置き換えます。例: `https://<domain name>.<AWS region>.es-staging.amazonaws.com`。

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4

domain_endpoint = 'domain endpoint'
pipeline_name = 'pipeline name'
url = f'{domain_endpoint}/_search/pipeline/{pipeline_name}'
auth = AWSSigV4('es')
```

```
headers = {'Content-Type': 'application/json'}

body = {
  "description": "A pipeline to apply custom re-ranking from Amazon Personalize",
  "response_processors": [
    {
      "personalized_search_ranking" : {
        "campaign_arn" : "Amazon Personalize Campaign ARN",
        "item_id_field" : "productId",
        "recipe" : "aws-personalized-ranking",
        "weight" : "0.3",
        "tag" : "personalize-processor",
        "iam_role_arn": "Role ARN",
        "aws_region": "AWS region",
        "ignore_failure": true
    }
  ]
}
try:
    response = requests.put(url, auth=auth, json=body, headers=headers, verify=False)
    print(response.text)
except Exception as e:
    print(f"Error: {e}")
```

personalized_search_ranking レスポンスプロセッサを使用して検索パイプラインを作成したら、 OpenSearch クエリへのプラグインの適用を開始する準備が整います。 OpenSearch インデックスまたは個々の OpenSearch クエリに適用できます。詳細については、「 OpenSearch クエリへのプラグインの適用」を参照してください。

オープンソースを使用したパイプラインの作成 OpenSearch

次の curl コマンドを使用して、オープンソース OpenSearch クラスターでpersonalized_search_rankingレスポンスプロセッサを使用して検索パイプラインを作成できます。

```
curl -X PUT "http://localhost:9200/_search/pipeline/pipeline-name" -ku 'admin:admin' --
insecure -H 'Content-Type: application/json' -d'
{
  "description": "A pipeline to apply custom re-ranking from Amazon Personalize",
  "response_processors" : [
    {
```

```
      "personalized_search_ranking" : {
        "campaign_arn" : "Amazon Personalize Campaign ARN",
        "item_id_field" : "productId",
        "recipe" : "aws-personalized-ranking",
        "weight" : "0.3",
        "tag" : "personalize-processor",
        "iam_role_arn": "Role ARN",
        "aws_region": "AWS region",
        "ignore_failure": true
      }
    }
  ]
}'
```

personalized_search_ranking レスポンスプロセッサを使用して検索パイプラインを作成したら、 OpenSearch クエリへのプラグインの適用を開始する準備が整います。 OpenSearch インデックスまたは個々の OpenSearch クエリに適用できます。詳細については、「 OpenSearch クエリへのプラグインの適用」を参照してください。

OpenSearch クエリへのプラグインの適用

personalized_search_ranking レスポンスプロセッサを使用して検索パイプラインを設定したら、Amazon Personalize Search Ranking プラグインを OpenSearch クエリに適用し、再ランク付けされた結果を表示する準備が整います。

プラグインを OpenSearch クエリに適用すると、検索パイプラインのメトリクスを取得してプラグインをモニタリングできます。詳細については、「プラグインの監視」を参照してください。

トピック

- Amazon OpenSearch Service クエリへのプラグインの適用
- オープンソースのクエリへのプラグインの適用 OpenSearch

Amazon OpenSearch Service クエリへのプラグインの適用

Amazon Personalize Search Ranking プラグインは、インデックスのすべてのクエリとレスポンスに適用できます。このプラグインは個々のクエリとレスポンスにも適用できます。

- 次の Python コードを使用して、検索パイプラインをインデックスに適用できます。この方法では、このインデックスを使用するすべての検索で、プラグインを使用して検索結果にパーソナライゼーションを適用します。

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4

domain_endpoint = 'domain endpoint'
index = 'index name'
url = f'{domain_endpoint}/{index}/_settings/'
auth = AWSSigV4('es')
headers = {'Content-Type': 'application/json'}
body = {
    "index.search.default_pipeline": "pipeline name"
}
try:
    response = requests.put(url, auth=auth, json=body, headers=headers)
    print(response.text)
except Exception as e:
    print(f"Error: {e}")
```

- 次の Python コードを使用して、トヨタブランドの自動車の個別のクエリに検索パイプラインを適用できます。

  コードを更新して、ドメインエンドポイント、 OpenSearch サービスインデックス、パイプラインの名前、クエリを指定します。user_id には、検索結果を取得するユーザーの ID を指定します。Amazon Personalize ソリューションバージョンの作成に使用したデータに、このユーザーが存在する必要があります。ユーザーが不在の場合、Amazon Personalize は人気に基づいてアイテムをランク付けします。

  context には、コンテキストメタデータを使用する場合は、デバイスタイプなどのユーザーのコンテキストメタデータを提供してください。context フィールドはオプションです。詳細については、「コンテキストメタデータを使用したレコメンデーションの関連性の向上」を参照してください。

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4

domain_endpoint = 'domain endpoint'
index = 'index name'
url = f'{domain_endpoint}/{index}/_search/'
```

```
auth = AWSSigV4('es')
headers = {'Content-Type': 'application/json'}
params = {"search_pipeline": "pipeline-name"}
body = {
    "query": {
        "multi_match": {
            "query": "Toyota",
            "fields": ["BRAND"]
        }
    },
    "ext": {
        "personalize_request_parameters": {
            "user_id": "USER ID",
            "context": { "DEVICE" : "mobile phone" }
        }
    }
}
try:
    response = requests.post(url, auth=auth, params=params, json=body,
 headers=headers, verify=False)
    print(response)
except Exception as e:
    print(f"Error: {e}")
```

オープンソースのクエリへのプラグインの適用 OpenSearch

Amazon Personalize Search Ranking プラグインは、 OpenSearch インデックスのすべてのクエリとレスポンスに適用できます。プラグインを個々の OpenSearch クエリとレスポンスに適用することもできます。

- 次の curl コマンドは、ローカルで実行されているオープンソース OpenSearch クラスターの OpenSearch インデックスに検索パイプラインを適用します。この方法では、このインデックスでのすべての検索がプラグインを使用して検索結果にパーソナライゼーションを適用します。

```
curl -XGET "https://localhost:9200/index/_settings" -ku 'admin:admin' --insecure -H
 'Content-Type: application/json' -d'
{
  "index.search.default_pipeline": "pipeline-name"
}
'
```

- 次の curl コマンドは、ローカルで実行されているオープンソース OpenSearch クラスターのインデックスにあるトヨタブランド車の個々のクエリに検索パイプラインを適用します。

  user_id には、検索結果を取得するユーザーの ID を指定します。Amazon Personalize ソリューションバージョンの作成に使用したデータに、このユーザーが存在する必要があります。ユーザーが不在の場合、Amazon Personalize は人気に基づいてアイテムをランク付けします。context には、コンテキストメタデータを使用する場合は、デバイスタイプなどのユーザーのコンテキストメタデータを提供してください。context フィールドはオプションです。詳細については、「コンテキストメタデータを使用したレコメンデーションの関連性の向上」を参照してください。

```
curl -XGET "http://localhost:9200/index/_search?search_pipeline=pipeline-name" -ku
 'admin:admin' --insecure -H 'Content-Type: application/json' -d'
{
  "query": {
    "multi_match": {
      "query": "Toyota",
      "fields": ["BRAND"]
    }
  },
  "ext": {
    "personalize_request_parameters": {
      "user_id": "USER ID",
      "context": { "DEVICE": "mobile phone" }
    }
  }
}
'
```

結果が再ランク付けされる方法を理解するには、 OpenSearch Dashboards を使用して、プラグインで OpenSearch 再ランク付けされた結果と結果を比較できます。詳細については、「 OpenSearch 結果をプラグインの結果と比較する」を参照してください。

プラグインを OpenSearch クエリに適用すると、パイプラインのメトリクスを取得してプラグインをモニタリングできます OpenSearch。詳細については、「プラグインの監視」を参照してください。

OpenSearch 結果をプラグインの結果と比較する

Amazon Personalize Search Ranking プラグインは、Amazon Personalize のランキングと のランキングの両方に基づいて検索結果を再配置します OpenSearch。プラグインが結果を再ランク付けする方法は、パイプラインで `personalized_search_ranking` レスポンスプロセッサをどのように設定したかによって異なります。

結果がどのように再ランク付けされるかを理解するには、パーソナライゼーションの有無にかかわらずクエリを実行し、結果を比較できます。

トピック

- Amazon OpenSearch Service との結果の比較
- 結果をオープンソースと比較する OpenSearch

Amazon OpenSearch Service との結果の比較

結果のランク付け方法を理解するには、パーソナライゼーションの有無にかかわらずクエリを実行し、結果を比較できます。次の Python コードを使用して 2 つの異なるクエリを実行し、結果を 2 つの JSON ファイルに出力できます。1 つ目の方法は、プラグインを使用して結果をランク付けし直すクエリを実行します。2 つ目は、パーソナライズせずに結果を生成するメソッドを実行します。

```
import json
import requests
from requests_auth_aws_sigv4 import AWSSigV4


# Returns re-ranked OpenSearch results using the Amazon Personalize Search Ranking
 plugin.
def get_personalized_results(pipeline_name):
    url = f'{domain}/{index}/_search/'
    auth = AWSSigV4('es')
    headers = {'Content-Type': 'application/json'}
    params = {"search_pipeline": pipeline_name}
    body = {
        "query": {
            "multi_match": {
                "query": "Toyota",
                "fields": ["BRAND"]
            }
        },
```

```
        "ext": {
            "personalize_request_parameters": {
                "user_id": "1"
            }
        }
    }
    try:
        response = requests.post(url, auth=auth, params=params, json=body,
 headers=headers, verify=False)
    except Exception as e:
        return f"Error: {e}"
    return response.text


# Returns OpenSearch results without personalization.
def get_opensearch_results():
    url = f'{domain}/{index}/_search/'
    auth = AWSSigV4('es')
    headers = {'Content-Type': 'application/json'}
    body = {
        "query": {
            "multi_match": {
                "query": "Toyota",
                "fields": ["BRAND"]
            }
        }
    }
    try:
        response = requests.post(url, auth=auth, json=body, headers=headers,
 verify=False)
    except Exception as e:
        return f"Error: {e}"
    return response.text


def print_results(file_name, results):
    results_file = open(file_name, 'w')
    results_file.write(json.dumps(results, indent=4))
    results_file.close()


# specify domain endpoint
domain = "DOMAIN_ENDPOINT"
```

```
# specify the region where you created your Amazon Personalize resources and Amazon
 OpenSearch domain
aws_region = "REGION"

# specify the name of the pipeline that uses the Amazon Personalize plugin
pipeline_name = "PIPELINE_NAME"

# specify your Amazon OpenSearch index
index = "INDEX"

# specify names for json files for comparison
personalized_results_file = "personalized_results.json"
opensearch_results_file = "opensearch_results.json"

# get personalized results
personalized_results = json.loads(get_personalized_results(pipeline_name))

# get OpenSearch results without personalization
opensearch_results = json.loads(get_opensearch_results())

# print results to files
print_results(personalized_results_file, personalized_results)
print_results(opensearch_results_file, opensearch_results)
```

結果をオープンソースと比較する OpenSearch

結果がどのように再ランク付けされるかを理解するために、2 つのブラウザウィンドウで開発ツールコンソールを使用してクエリを実行できます。そうすれば、パーソナライゼーションの有無にかかわらずクエリの結果を比較できます。

Dev Tools コンソールと結果を比較するには

1. まだの場合は、プラグインのセットアップ OpenSearch とインストール と プラグインの設定 の手順に従ってください。
2. OpenSearch Dashboards がインストールされていることを確認します。クイックスタート bash スクリプトは OpenSearch Dashboards をインストールします。スクリプトを使用しない場合、またはクラスターが既に実行されている場合は、 OpenSearch Dashboards をインストールする必要があります。詳細については、 OpenSearch 「ダッシュボードのインストール」を参照してください。

3. OpenSearch ダッシュボードを起動します。ブラウザhttp://localhost:5601から を開き、Dashboards に OpenSearchサインインします。デフォルトの認証情報は、ユーザー名は「admin」、パスワードは「admin」です。
4. OpenSearch Dashboards ホームページの管理メニューで開発ツールを選択します。
5. 別のブラウザウィンドウを開き、Dev Tools コンソールをもう一度開きます。前のウィンドウの URL を使用できます。
6. 1 つのウィンドウに、パーソナライゼーションのための再ランク付けを一切使用しないクエリを入力します。もう一方のウィンドウでは、personalized_search_ranking レスポンスプロセッサのパイプラインを使用する curl コマンドを入力します。curl コマンドをコンソールに直接貼り付けると、コマンドはコンソールが使用する形式に自動的に変換されます。コマンドの例については、「 OpenSearch クエリへのプラグインの適用」を参照してください。
7. 両方のクエリを実行し、結果を比較します。

プラグインの監視

OpenSearch サービスを使用する場合は、Amazon のメトリクスを使用してプラグインをモニタリングできます CloudWatch。詳細については、「Amazon OpenSearch Service ドメインのモニタリング」を参照してください。

Amazon Personalize Search Ranking プラグインを OpenSearch クエリに適用すると、検索パイプラインのメトリクスを取得してプラグインをモニタリングできます。パイプラインメトリックスには、personalized_search_ranking レスポンスプロセッサの失敗したリクエスト数などの統計が含まれます。

トピック

- Amazon OpenSearch Service によるプラグインのモニタリング
- オープンソースでプラグインをモニタリングする OpenSearch
- パイプラインメトリクスの例

Amazon OpenSearch Service によるプラグインのモニタリング

次の Python コードを使用して、すべてのパイプラインメトリクスを取得できます。パイプラインの例については、「パイプラインメトリクスの例」を参照してください。

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4
```

```
domain_endpoint = 'domain endpoint'
url = f'{domain_endpoint}/_nodes/stats/search_pipeline'

auth = AWSSigV4('es')
headers = {'Content-Type': 'application/json'}
try:
    response = requests.get(url, auth=auth, headers=headers, verify=False)
    print(response.text)
except Exception as e:
    print(f"Error: {e}")
```

オープンソースでプラグインをモニタリングする OpenSearch

次のコードを使用して、すべてのパイプラインのメトリクスを取得できます。レスポンスにはすべての検索パイプラインの統計が含まれます。パイプラインメトリクスの例については、「パイプラインメトリクスの例」を参照してください。

```
curl -XGET "https://localhost:9200/_nodes/stats/search_pipeline?pretty" -ku
 'admin:admin'
```

パイプラインメトリクスの例

次のコードは、 から返されるパイプラインメトリクスの抜粋を示しています OpenSearch。2 つの異なるパイプラインの統計を含む pipelines オブジェクトのみが表示されます。パイプラインごとに、Amazon Personalize Search Ranking プラグインのメトリックスが personalized_search_ranking レスポンスプロセッサリストに表示されます。すべてのメトリクスの詳細な例については、「検索パイプラインメトリクス」を参照してください。

```
{
....
....
  "pipelines": {
    "pipelineA": {
      "request": {
        "count": 0,
        "time_in_millis": 0,
        "current": 0,
        "failed": 0
      },
      "response": {
```

```
        "count": 6,
        "time_in_millis": 2246,
        "current": 0,
        "failed": 0
      },
      "request_processors": [],
      "response_processors": [
        {
          personalized_search_ranking": {
            "type": "personalized_search_ranking",
            "stats": {
              "count": <number of requests>,
              "time_in_millis": <time>,
              "current": 0,
              "failed": <number of failed requests>
            }
          }
        }
      ]
    },
    "pipelineB": {
      "request": {
        "count": 0,
        "time_in_millis": 0,
        "current": 0,
        "failed": 0
      },
      "response": {
        "count": 8,
        "time_in_millis": 2248,
        "current": 0,
        "failed": 0
      },
      "request_processors": [],
      "response_processors": [
        {
          "personalized_search_ranking": {
            "type": "personalized_search_ranking",
            "stats": {
              "count": <number of requests>,
              "time_in_millis": <time>,
              "current": 0,
              "failed": <number of failed requests>
            }
```

```
          }
        }
      ]
    }
  }
....
....
}
```

Amazon Personalize リソースのタグ付け

タグは、特定のタイプの Amazon Personalize AWS リソースなど、オプションで定義してリソースに関連付けるラベルです。リソースには、最大 50 個のタグを含めることができます。

タグを使用することで、目的、環境、その他の条件など、さまざまな方法でリソースを分類および管理できます。例えば、タグを使用して、収益をさまざまな部門に分配したり、さまざまなリソースの開発環境を特定したりできます。

Amazon Personalize リソースをタグで取得するには、Resource Groups Tagging API の GetResources オペレーションでフィルターを使用できます。詳細については、「Resource Groups Tagging API Reference guideGetResources」の「」を参照してください。

以下のタイプの Amazon Personalize リソースにタグを追加できます。

- バッチ推論ジョブ
- バッチセグメントジョブ
- キャンペーン
- データセット
- データセットグループ
- データセットのインポートとエクスポートのジョブ
- イベントトラッカー
- フィルター
- レコメンダー
- 解決方法
- ソリューションバージョン

トピック

ガイドラインと要件

タグはそれぞれ、1 つの必須タグキーとオプションの 1 つのタグ値で構成されており、どちらもお客様側が定義します。タグキーは、より具体的なタグ値のカテゴリのように動作する、一般的なラベルです。タグ値は、タグキーの記述子として機能します。

例えば、Amazon Personalize のデータセットグループに 2 つのバージョンがある場合 (1 つは内部テスト用、もう 1 つは本番用)、両方のプロジェクトに `Environment` タグキーを割り当てます。`Environment` タグキーの値は、場合によって、プロジェクトの 1 つのバージョンでは `Test` に、他のバージョンでは `Production` になる場合があります。

タグには以下の制限があります。

- リソースあたりのタグの最大数 - 50 件
- キーの最大長 - UTF-8 の 128 Unicode 文字
- 値の最大長 - UTF-8 の 256 Unicode 文字
- タグキーと値には、次のような文字、A-Z、a-z、0-9、space、 _ . : / = + @ – (ハイフン) を含むことができます。これは、タグをサポートするすべての AWS サービスで使用できる標準の文字セットです。一部のサービスでは追加の記号がサポートされています。
- タグのキーと値は大文字と小文字が区別されます。
- タグキーは、関連付けられたリソースごとにそれぞれ一意である必要があります。また、1 つのみのタグの値を持ちます。
- タグキーとタグ値は で始まることはできません`aws:`。 AWS サービスは で始まるタグを適用し`aws:`、それらのタグは変更できません。タグの上限にはカウントされません。
- タグのみに基づいて、リソースを更新または削除することはできません。使用する操作に応じて、Amazon リソースネーム (ARN) またはリソース ID も指定する必要があります。

追加情報

タグ付けの詳細については、次のリソースを参照してください。

- AWS 全般のリファレンスのAWS タグ付けの原則
- AWS タグ付け戦略 (ダウンロード可能な PDF)
- IAM ユーザーガイドのAWS 「アクセスコントロールAWS 」
- AWS Organizations ユーザーガイドAWS のタグ付けポリシー

Amazon Personalize リソースへのタグ追加

Amazon Personalize コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKs を使用して、Amazon Personalize リソースからタグキーと値を追加、表示、更新、削除できます。次の例では、Amazon Personalize データセットグループにタグを追加する方法を示しています。同じ方法で他の Amazon Personalize リソースにタグを追加できます。

トピック

タグの追加 (コンソール)

Amazon Personalize コンソールでリソースを作成するときに、Amazon Personalize コンソールでオプションのタグを追加できます。次の例では、データセットグループにタグを追加します。

新しいデータセットグループにタグを追加するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. [データセットグループの作成] を選択します。
3. [ロール名] に名前を入力します。
4. [ドメイン] では、ドメインを選択します。
5. [タグ] セクションで [新しいタグを追加する] を選択します。
6. [Key] (キー) と [Value] (値) で、適切な値を入力します。

   例えば、それぞれ **`Environment`** および **`Test`** などです。
7. タグをさらに追加するには、[Add new tag] (新しいタグを追加) を選択します。

   1 つのリソースに最大 50 個のタグを追加できます。
8. [Next] (次へ) を選択し、リソースの作成を続行します。

既存のリソースにタグを追加するのも同様です。リソースを選択し、「タグ」フィールドを使用してタグを追加します。

タグの追加 (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して、リソースの作成時にタグを追加したり、既存のリソースにタグを追加したりできます。

トピック

リソース作成時のタグの追加

新しいリソースを作成し、 を使用してタグを追加するには AWS CLI、リソースに適切なcreateコマンドを使用し、 tagsパラメータと値を含めます。例えば、次のコマンドは、ECOMMERCE ドメインに対して myDatasetGroup という名前の新しいドメインデータセットグループを作成し、次のタグを追加します。Environment タグキーには Test タグ値、Owner タグキーには xyzCorp 値を追加します。

```
aws personalize create-dataset-group \
--name myDatasetGroup \
--domain ECOMMERCE \
--tags tagKey=Environment,tagValue=Test tagKey=Owner,tagValue=xyzCorp
```

Amazon Personalize リソースの作成に使用できるコマンドの詳細については、「Amazon Personalize AWS CLI コマンドリファレンス」を参照してください。

既存のリソースにタグを追加する

既存のリソースにタグを追加するには、 tag-resource コマンドを使用します。リソースの ARN を指定し、 tagsパラメータにタグキーと値を指定します。

```
aws personalize tag-resource \
--resource-arn resource ARN \
--tags tagKey=key,tagValue=value
```

タグの追加 (AWS SDKs)

AWS SDKs、リソースの作成時にタグを追加したり、既存のリソースにタグを追加したりできます。

トピック

- リソース作成時のタグの追加
- 既存のリソースにタグを追加する

リソース作成時のタグの追加

AWS SDKs を使用して新しいリソースを作成し、そのリソースにタグを追加するには、適切な `create` メソッドを使用します。`tags` パラメータを使用して、各タグのキーと値のペアを指定します。例えば、次のコードは、ECOMMERCE ドメインに対して `myDatasetGroup` という名前の新しいドメインデータセットグループを作成し、次のタグを追加します。`Environment` タグキーには `Test` タグ値、`Owner` タグキーには `xyzCorp` 値を追加します。

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_group(
  name = 'myDatasetGroup',
  domain = 'ECOMMERCE',
  tags = [
    {
      'tagKey': 'Environment',
      'tagValue': 'Test'
    },
    {
      'tagKey': 'Owner',
      'tagValue': 'xyzCorp'
    }
  ]
)
dsg_arn = response['datasetGroupArn']

description = personalize.describe_dataset_group(datasetGroupArn = dsg_arn)
['datasetGroup']

print('Name: ' + description['name'])
print('ARN: ' + description['datasetGroupArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

```
public static String createDomainDatasetGroup(PersonalizeClient personalizeClient,
                                               String datasetGroupName,
                                               String domain) {

    try {

        ArrayList <Tag> tags = new ArrayList<>();

        Tag tag1 = Tag.builder()
                .tagKey("Environment")
                .tagValue("Test")
                .build();
        tags.add(tag1);
        Tag tag2 = Tag.builder()
                .tagKey("Owner")
                .tagValue("xyzCorp")
                .build();
        tags.add(tag2);

        CreateDatasetGroupRequest createDatasetGroupRequest =
 CreateDatasetGroupRequest.builder()
                .name(datasetGroupName)
                .domain(domain)
                .tags(tags)
                .build();
        return
 personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

既存のリソースにタグを追加する

次のコードは、既存の Amazon Personalize リソースにタグを追加する方法を示しています。タグを追加するリソースの Amazon リソースネーム (ARN) を指定し、タグごとにキーと値のペアを指定します。

SDK for Python (Boto3)

```
import boto3
personalize = boto3.client('personalize')

add_tags_response = personalize.tag_resource(
  resourceArn = "resourceArn",
  tags = [
    {
      'tagKey': 'Environment',
      'tagValue': 'Test'
    },
    {
      'tagKey': 'Owner',
      'tagValue': 'xyzCorp'
    }
  ]
)
```

SDK for Java 2.x

```
public static void tagResource(PersonalizeClient personalizeClient,
                                                String resourceArn,
                                                String domain) {

    try {

         ArrayList <Tag> tagList = new ArrayList<>();

          Tag tag1 = Tag.builder()
                  .tagKey("Environment")
                  .tagValue("Test")
                  .build();
          tags.add(tag1);
          Tag tag2 = Tag.builder()
                  .tagKey("Owner")
                  .tagValue("xyzCorp")
                  .build();
          tags.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn(resourceArn)
                .tags(tagList)
```

```
                .build();

        personalizeClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to "+ resourceArn);

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Amazon Personalize リソースからのタグの削除

Amazon Personalize リソースからタグを削除するには、Amazon Personalize コンソールを使用するか、 AWS Command Line Interface （ AWS CLI) または SDK を使用して UntagResource API オペレーションを実行します。 AWS SDKs 次の例は、Amazon Personalize データセットグループからタグを削除する方法を示しています。同じ方法で、他の Amazon Personalize リソースからタグを削除できます。

トピック

- タグの削除 (コンソール)
- タグの削除 (AWS CLI)
- タグの削除 (AWS SDKs)

タグの削除 (コンソール)

Amazon Personalize のリソースにタグを追加したら、Amazon Personalize コンソールを使用してタグを削除できます。次の例では、データセットグループからタグを削除します。

データセットグループからタグを削除するには

1. https://console.aws.amazon.com/personalize/home で Amazon Personalize コンソールを開き、アカウントにサインインします。
2. データセットグループを選択します。
3. ページの下部で、タグ タブを選択し、タグの管理 を選択します。
4. 削除するタグごとに、 の削除 を選択します。
5. 保存 を選択してタグを削除します。

タグの削除 (AWS CLI)

を使用して既存のリソースからタグを削除するには AWS CLI、次のuntag-resourceコマンドを使用します。にはresource-arn、リソースの Amazon リソースネーム (ARN) を指定します。にはtag-keys、削除するタグのキーを指定します。

```
aws personalize untag-resource \
--resource-arn resource ARN \
--tag-keys key1 key2
```

タグの削除 (AWS SDKs)

AWS SDKsオペレーションを使用します。 UntagResource次のコードは、SDK for Python (Boto3) を使用してデータセットグループから複数のタグを削除する方法を示しています。にはresourceArn、リソースの Amazon リソースネーム (ARN) を指定します。にはtagKeys、削除するタグのキーを指定します。

```
import boto3

personalize = boto3.client('personalize')

response = personalize.untag_resource(
    resourceArn="Resource ARN",
    tagKeys=["tag1Key", "tag2Key"]
)
```

IAMポリシーでタグを使用する

タグの実装を開始した後、タグベースのリソースレベルのアクセス許可を AWS Identity and Access Management (IAM) ポリシーと API オペレーションに適用できます。これには、リソースの作成時にリソースへのタグの追加をサポートするオペレーションが含まれます。この方法でタグを使用すると、 AWS アカウント内のどのグループとユーザーがリソースを作成およびタグ付けするアクセス許可を持っているか、どのグループとユーザーがタグをより一般的に作成、更新、削除するアクセス許可を持っているかをきめ細かく制御できます。

例えば、リソースの Owner タグの値がユーザー名となっているすべての Amazon Personalize rリソースに対して、ユーザーにフルアクセスを許可するポリシーを作成できます。

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "ModifyResourceIfOwner",
         "Effect": "Allow",
         "Action": "personalize:*",
         "Resource": "*",
         "Condition": {
            "StringEqualsIgnoreCase": {
               "aws:ResourceTag/Owner": "${aws:username}"
            }
         }
      }
   ]
}
```

次の例は、データセットの作成と削除を許可するポリシーを作成する方法を示しています。これらのオペレーションは、ユーザー名が johndoe の場合にのみ許可されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:CreateDataset",
                "personalize:DeleteDataset"
            ],
            "Resource": "arn:aws:personalize:*:*:dataset/*",
            "Condition": {
                "StringEquals": {"aws:username" : "johndoe"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "personalize:DescribeDataset",
            "Resource": "*"
        }
    ]
}
```

タグをベースにしてリソースレベルでアクセス許可を定義した場合、そのアクセス許可は即座に反映されます。つまり、リソースが作成されるとすぐにリソースの安全性が増し、新しいリソースにタ

グの使用をすぐに強制できるようになります。リソースレベルのアクセス許可を使用して、新しいリソースと既存のリソースに、どのタグキーと値を関連付けるかを制御することもできます。詳細については、AWS IAM ユーザーガイドの「タグを使用したアクセス制御」を参照してください。

トラブルシューティング

以下のトピックでは、Amazon Personalize で発生する可能性のあるエラーメッセージに関する一般的な質問への回答とトラブルシューティングのアドバイスを提供します。Amazon Personalize がお客様のユースケースに適合するかどうかを判断するのに役立つクイックリファレンスについては、Amazon Personalize サンプルリポジトリにある「Amazon Personalize チートシート」を参照してください。

トピック

- よくある質問
- エラーメッセージ

よくある質問

以下は、Amazon Personalize でのデータのインポート、トレーニング、モデルのデプロイ、レコメンデーション、フィルターに関するよくある質問への回答です。

その他の質問と回答については、Amazon Personalize サンプルリポジトリの「Amazon Personalize のチートシート」を参照してください

トピック

- データのインポートと管理
- カスタムソリューションとソリューションバージョンの作成
- モデルのデプロイ (カスタムキャンペーン)
- レコメンデーション
- レコメンデーションのフィルタリング

データのインポートと管理

バルクデータはどのような形式にすべきですか?

入力データをコンマ区切り値 (CSV) 形式である必要があります。CSV データセットファイルの最初の行には、ファイルヘッダーが含まれている必要があります。CSV ファイルの列ヘッダーは、データセットを作成するためにスキーマに対応している必要があります。データに ASCII でエンコードされていない文字が含まれている場合は、CSV ファイルを UTF-8 形式でエンコードする必要があ

ります。ヘッダーを引用符 (") で囲まないでください。TIMESTAMP および CREATION_TIMESTAMP データは UNIX エポック時間形式である必要があります。データのインポートの詳細については、「タイムスタンプのデータ」を参照してください。スキーマの詳細については、「 スキーマ」を参照してください。

データフォーマットのガイドラインの詳細については、「データ形式ガイドライン」を参照してください。データのフォーマット方法がわからない場合は、Amazon Data Wrangler (SageMaker データラングラー) を使用してデータを準備できます。詳細については、「Amazon SageMaker Data Wrangler を使用したデータの準備とインポート」を参照してください。

どの程度のトレーニングが必要ですか?

すべてのユースケース (ドメインデータセットグループ) とカスタムレシピにおいて、インタラクションデータには以下が必要です。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

質の高いレコメンデーションを行うには、1,000 人以上のユーザーからのアイテムインタラクションが少なくとも 50,000 件あり、それぞれ 2 回以上のアイテムインタラクションがあることが推奨されます。

空のアイテムインタラクションデータセットから始めて、十分なデータを記録したら、新しく記録されたイベントのみを使用して、レコメンダー (ドメインデータセットグループ) またはカスタムソリューションバージョンを作成できます。一部のレシピとユースケースには、追加のデータ要件がある場合があります。ユースケースの要件については、レシピ要件の詳細については、「レシピの選択」を参照してください。

アイテムやユーザーの属性を更新する方法を教えてください。

Amazon Personalize コンソール、PutItems または PutUsers オペレーションを使用して、同じアイテム ID を持つものの属性が変更されたアイテムまたはユーザーをインポートします。

アイテムまたはユーザーを削除する方法

Amazon Personalize は、特定のアイテムまたはユーザーの削除をサポートしていません。アイテムやユーザーがレコメンデーションに表示されないようにするには、フィルターを使用して商品を除外

します。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

スキーマを削除する方法

スキーマは DeleteSchema 操作でのみ削除できます。Amazon Personalize コンソールでは、スキーマを削除することはできません。

カスタムソリューションとソリューションバージョンの作成

どのレシピを使うべきですか?

使用する Amazon Personalize レシピは、ユースケースによって異なります。ユースケースとレシピを一致させる方法については、「レシピの選択」を参照してください。「Amazon Personalize チートシート」には、ユースケースとレシピの情報も含まれています。

どのくらいの頻度でトレーニングすればよいですか?

最低でも週1回のトレーニング頻度で自動トレーニングを行うことをおすすめします。自動トレーニングにより、推奨内容の関連性を維持しやすくなります。トレーニングの頻度は、ビジネス要件、使用するレシピ、データをインポートする頻度によって異なります。詳細については、「自動トレーニングの設定」を参照してください。関連性を維持する方法については、を参照してくださいレコメンデーションの関連性の維持。

AutoML を使うべきですか?

いいえ。代わりに、ユースケースをさまざまな Amazon Personalize レシピと一致させ、レシピを選択することをお勧めします。ユースケースとレシピを一致させる方法については、「レシピの選択」を参照してください。

モデルのデプロイ (カスタムキャンペーン)

キャンペーンの最小プロビジョンドTPSには何を設定すればよいですか ?

`minProvisionedTPS` の値を高く設定するとコストが増加します。最初は `minProvisionedTPS` に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、`minProvisionedTPS`必要に応じて増やします。

キャンペーンの費用をモニタリングする方法を教えてください。

Amazon Personalize Monitor プロジェクトは、Amazon Personalize CloudWatch キャンペーン用のダッシュボード、カスタムメトリックス、使用率アラーム、コスト最適化機能を提供しま

す。Amazon Personalizeのサンプルリポジトリにある「Amazon Personalize モニター」を参照してください。

キャンペーンの最大トランザクションスループットを設定する方法を教えてください。

キャンペーンの最小スループットのみを設定できます。Amazon Personalize のキャンペーンを作成する場合、アプリケーションユーザー向けのリアルタイムのレコメンデーションを作成するための専用のトランザクション容量を指定します。TPS が `minProvisionedTPS` を超えて増加した場合、Amazon Personalize はプロビジョンド容量を自動スケーリングしますが、`minProvisionedTPS` を下回ることはありません。詳細については、「1 秒あたりの最小プロビジョンドトランザクション数とオートスケーリング」を参照してください。

レコメンデーション

Amazon Personalize モデルが品質に関する推奨事項を生成しているかどうかはどうすればわかりますか?

オフラインとオンラインの指標 (「メトリクスを使用してソリューションバージョンを評価する」を参照) とオンラインテスト (A/B テストなど) を使用して、ソリューションバージョンのパフォーマンスを評価します。A/B テストの詳細については、を参照してくださいA/B テストを使用したレコメンデーションの影響の測定。

バッチ推論ジョブを削除する方法と、そのステータスが「アクティブ」なのはなぜですか?

バッチ推論ジョブは削除できません。バッチ推論ジョブのステータスがアクティブになると、ジョブは完了です。出力 Amazon S3 バケットまたはフォルダでレコメンデーションにアクセスできます。バッチ推論ジョブが完了しても、追加コストは発生しません。ただし、Amazon S3 などの他のサービスでは、入出力データのストレージに追加料金がかかる場合があります。

SIMS が支援するキャンペーンで、メタデータに基づいて類似していないアイテムが推奨されるのはなぜですか?

SIMS は、色や料金などのアイテムメタデータではなく、アイテムインタラクションデータセットを使用して類似性を判断します。SIMS は、Interactions データセットのユーザー履歴でアイテムの共起を識別して、類似アイテムを推奨します。詳細については、「SIMS の recipe」を参照してください。

1 GetRecommendations 回の API オペレーションで 500 を超えるアイテムを取得できますか?

500 個は、1 回の GetRecommendations で取得できるアイテムの最大数です。この値を増やすことはできません。

レコメンデーションのフィルタリング

おすすめ商品が期待どおりにフィルタリングされないのはなぜですか？

これは、さまざまな理由で発生する可能性があります。

- フィルター式の形式や構文に問題がある可能性があります。フィルター式の例については、「フィルター式の例」を参照してください。
- Amazon Personalize は、イベントタイプごとに、ユーザー 1 人あたり最大 100 件の最新のインタラクションを考慮します。これは調整可能なクォータです。Service Quotas コンソールを使用してクォータの増加をリクエストできます。

詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

購入済みの商品をレコメンデーションから削除する方法を教えてください。

ECOMMERCE ドメインデータセットグループでは、おすすめ または X を閲覧したお客様はこちらも閲覧しました ユースケースのレコメンダーを作成すると、Amazon Personalize は、指定した userId と `Purchase` イベントに基づいて、ユーザーが購入したアイテムを自動的にフィルタリングします。

他のドメインデータセットグループのユースケースやカスタムリソースでは、フィルターを使用して購入したアイテムを削除します。データに `Purchased` イベントタイプ属性を追加し、`PutItems`オペレーションと共に購入イベントを記録して、購入アイテムをレコメンデーションから削除するフィルターを作成します。例:

```
EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("purchased")
```

詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

エラーメッセージ

以下のセクションでは、Amazon Personalize の使用時に発生する可能性のある一部のメッセージの一覧と説明をします。

トピック

- データのインポートと管理
- ソリューションとソリューションバージョン (カスタムリソース) の作成
- モデルデプロイ (カスタムキャンペーン)
- レコメンダー (ドメインデータセットグループ)
- レコメンデーション
- レコメンデーションのフィルタリング

データのインポートと管理

エラーメッセージ: データの場所が無効です。

Amazon S3 バケットの場所用に正しい構文を使用していることを確認してください。データセットのインポートジョブでは、Amazon S3 内のデータの場所に次の構文を使用します。

s3://<name of your S3 bucket>/<folder path>/<CSVfilename>

CSV ファイルが S3 バケット内のフォルダにあり、1 つのデータセットのインポートジョブで複数の CSV ファイルをデータセットにアップロードする場合は、CSV ファイル名なしでこの構文を使用します。

エラーメッセージ: CreateDataSetImportJob オペレーションを呼び出しているときに、エラー (LimitLimitExceededException) が発生しました: 5 つ以上のリソースが PENDING または IN_PROGRESS の状態です。

1 つのリージョンにつき、保留中または進行中のデータセットのインポートジョブを 5 つまで作成できます。これは調整可能なクォータではありません。Amazon S3 アクセス許可の詳細なリストについては、「Amazon Personalize エンドポイントとクォータ」を参照してください。

エラーメッセージ: <dataset type> データセットのデータインポートジョブを作成できませんでした... Amazon S3 のデータにアクセスするための許可が不十分です。

アクセスポリシーを Amazon S3 バケットと Amazon Personalize のサービスロールにアタッチすることにより、Amazon S3 リソースへのアクセス許可を Amazon Personalize に付与します。「Amazon Personalize に対する、Amazon S3 リソースへのアクセスの付与」を参照してください。

暗号化に AWS Key Management Service (AWS KMS KMS) を使用している場合は、キーを使用するための許可を、IAM ユーザーと Amazon Personalize の IAM サービスロールに付与する必要があり

ます。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

エラーメッセージ: データインポートジョブ <dataset type> データセットを作成できませんでした... 入力 CSV に次の列がありません:[COLUMN_NAME, COLUMN_NAME]。

Amazon Personalize にインポートするデータ (属性名やデータ型を含む) は、インポート先データセットのスキーマと一致する必要があります。詳細については、「スキーマ」を参照してください。

エラーメッセージ: <COLLUMN_NAME> の長さは <character limit> 文字を超えないようにしてください。文字数制限を超える値がない場合は、データが https://docs.aws.amazon.com/personalize/latest/dg/data-prep-formatting.html に記載されているフォーマットガイドラインに従っていることを確認してください。

この列のすべての値が文字制限を超えていないことを確認してください。文字数制限を超える値がない場合は、先行するテキストフィールドに次の項目がないか確認してください。

- テキストデータはすべて二重引用符で囲まれていることを確認してください。\ 文字を使用して、データ内の二重引用符または \ 文字をエスケープします。
- CSV ファイル内の各レコードが 1 行になっていることを確認します。

ソリューションとソリューションバージョン (カスタムリソース) の作成

エラーメッセージ: 作成に失敗しました。それぞれ 2 つのインタラクションを持つデータセットのユーザーが 25 人未満です。

モデルをトレーニングする前に、さらにデータをインポートする必要があります。モデルをトレーニングするための最小データ要件は次のとおりです。

- カタログ内のアイテムを操作したユーザーからのアイテムインタラクションレコードが少なくとも 1000 件ある。これらのインタラクションは、一括インポート、ストリーミングイベント、あるいはその両方からのものである。
- それぞれに 2 回以上のアイテムインタラクションを伴う 25 個以上のユニークユーザー ID。

リアルタイムのレコメンデーションには、データセットインポートジョブでより多くのデータをインポートするか、イベントトラッカーと PutEvents 操作でユーザーのインタラクションイベントをより多く記録します。リアルタイムイベントの記録の詳細については、「イベントの記録」を参照してください。

バッチレコメンデーションについては、データがまだあるときにデータセットのインポートジョブを使用してデータをインポートします。バルクデータのインポートについての詳細は、「ステップ 2: データの準備とインポート」を参照してください。

モデルデプロイ (カスタムキャンペーン)

エラー: キャンペーンを作成できません。5 つ以上のリソースがアクティブ状態です。いくつか削除して、もう一度試してください。

1 つのデータセットグループにつき、合計 5 つの Amazon Personalize キャンペーンをアクティブにできます。調整可能なクォータの場合、Service Quotas コンソールを使用してクォータの引き上げをリクエストできます。Amazon Personalize の制限とクォータの詳細なリストについては、「Amazon Personalize エンドポイントとクォータ」を参照してください。

レコメンダー (ドメインデータセットグループ)

エラー: イベントタイプ: <event type> でフィルタリングした結果、データセットのインタラクション数が 1000 件未満

ユースケースが異なれば、必要なイベントタイプも異なります。データには、ユースケースに必要なタイプのイベントが少なくとも 1000 件含まれている必要があります。詳細については、「ユースケースの選択」を参照してください。

レコメンデーション

Batch 推論ジョブのエラーメッセージ: 無効な S3 入力パスまたは 無効な S3 出力パス

Amazon S3 の入力場所または出力場所用に正しい構文を使用していることを確認してください。また、出力場所が入力データとは異なることを確認してください。同じ Amazon S3 バケットまたは別のバケット内のフォルダである必要があります。

Amazon S3 の入力ファイルの場所には次の構文を使用します: **s3://<name of your S3 bucket>/<folder name>/<input JSON file name>**

Amazon S3 の出力フォルダには次の構文を使用します: **s3://<name of your S3 bucket>/<output folder name>/**

レコメンデーションのフィルタリング

エラーメッセージ: フィルターを作成できませんでした。入力記号: $parameterName が無効です。プレースホルダーは NOT_IN 演算子では使用できません。

NOT_IN 演算子を使用するフィルター式では、プレースホルダーパラメーターは使用できません。代わりに IN 演算子を使用し、その逆のアクションを使用してください。Exclude の代わりに Include を使用してください (またはその逆)。

例えば、`INCLUDE ItemID WHERE Items.GENRE NOT IN ($GENRE)` 使用したい場合に `EXCLUDE ItemID WHERE Items.GENRE IN ($GENRE)` を使用しても同じ結果が得られます。

フィルターの詳細については、「フィルター式の要素」を参照してください。

エラーメッセージ: フィルターを作成できませんでした。式が無効です... ブール型フィールドをフィルタリングする場合

スキーマでブール型の値を使用してフィルタリングするフィルター式を作成することはできません。ブール値に基づいてフィルタリングするには、`String` タイプのフィールドを持つスキーマを使用し、データの `True` および `False` の値を使用します。または、`int` または `long` タイプもしくは `0` または `1` の値を使用できます。

フィルターの詳細については、「フィルター式の要素」を参照してください。

AWS CloudFormation を使用してリソースを指定する

Amazon Personalize は AWS CloudFormation と統合されています。これは、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスです。(Amazon Personalize データセットグループなど) を指定できるすべての AWS リソースを記述するテンプレートを作成します。AWS CloudFormationそして、ユーザーに代わってこれらのリソースのプロビジョニングおよび設定を行います。

AWS CloudFormation を使用すると、テンプレートを再利用して Amazon Personalize リソースをいつでも繰り返しセットアップできます。リソースを一度記述するだけで、同じリソースを複数の AWS アカウント とリージョンで何度でもプロビジョニングできます。

トピック

- Amazon Personalize と AWS CloudFormation テンプレート
- Amazon Personalize リソースの AWS CloudFormation テンプレートの例
- AWS CloudFormation の詳細はこちら

Amazon Personalize と AWS CloudFormation テンプレート

Amazon Personalize および関連サービスのリソースをプロビジョニングして設定するには、AWS CloudFormation テンプレートについて理解しておく必要があります。テンプレートは、JSON または YAML でフォーマットされたテキストファイルです。これらのテンプレートには、AWS CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、AWS CloudFormation Designer を使えば、AWS CloudFormation テンプレートを使いこなすことができます。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation Designer とは」を参照してください。

Amazon Personalize では、AWS CloudFormation におけるデータセット、データセットグループ、データセットインポートジョブ、スキーマ、ソリューションの指定がサポートされています。詳細については、AWS CloudFormation ユーザーガイドの「Amazon Personalize リソースタイプのリファレンス」を参照してください。

Amazon Personalize リソースの AWS CloudFormation テンプレートの例

次の AWS CloudFormation テンプレートの例は、さまざまな Amazon Personalize リソースの指定方法を示しています。

トピック

CreateDatasetGroup

JSON

```
{
   "AWSTemplateFormatVersion":"2010-09-09",
   "Resources":{
      "MyDatasetGroup": {
            "Type": "AWS::Personalize::DatasetGroup",
            "Properties": {
               "Name": "my-dataset-group-name"
            }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyDatasetGroup:
    Type: 'AWS::Personalize::DatasetGroup'
    Properties:
      Name: my-dataset-group-name
```

CreateDataset

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "MyDataset": {
      "Type": "AWS::Personalize::Dataset",
      "Properties": {
        "Name": "my-dataset-name",
        "DatasetType": "Interactions",
        "DatasetGroupArn": "arn:aws:personalize:us-west-2:123456789012:dataset-
group/dataset-group-name",
        "SchemaArn": "arn:aws:personalize:us-west-2:123456789012:schema/schema-
name",
        "DatasetImportJob": {
          "JobName": "my-import-job-name",
          "DataSource": {
            "DataLocation": "s3://bucket-name/file-name.csv"
          },
          "RoleArn": "arn:aws:iam::123456789012:role/personalize-role"
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyDataset:
    Type: 'AWS::Personalize::Dataset'
    Properties:
      Name: my-dataset-name
      DatasetType: Interactions
      DatasetGroupArn: 'arn:aws:personalize:us-west-2:123456789012:dataset-group/
dataset-group-name'
      SchemaArn: 'arn:aws:personalize:us-west-2:123456789012:schema/schema-name'
      DatasetImportJob:
        JobName: my-import-job-name
        DataSource:
```

```
          DataLocation: 's3://bucket-name/file-name.csv'
        RoleArn: 'arn:aws:iam::123456789012:role/personalize-role'
```

CreateSchema

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MySchema": {
            "Type": "AWS::Personalize::Schema",
            "Properties": {
                "Name": "my-schema-name",
                "Schema": "{\"type\": \"record\",\"name\": \"Interactions\",
 \"namespace\": \"com.amazonaws.personalize.schema\", \"fields\": [ { \"name\":
 \"USER_ID\", \"type\": \"string\" }, { \"name\": \"ITEM_ID\", \"type\": \"string
\" }, { \"name\": \"TIMESTAMP\", \"type\": \"long\"}], \"version\": \"1.0\"}"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MySchema:
    Type: AWS::Personalize::Schema
    Properties:
      Name: "my-schema-name"
      Schema: >-
        {"type": "record","name": "Interactions", "namespace":
        "com.amazonaws.personalize.schema", "fields": [ { "name": "USER_ID",
        "type": "string" }, { "name": "ITEM_ID", "type": "string" }, { "name":
        "TIMESTAMP", "type": "long"}], "version": "1.0"}
```

CreateSolution

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MySolution": {
            "Type": "AWS::Personalize::Solution",
            "Properties": {
               "Name": "my-solution-name",
               "DatasetGroupArn": "arn:aws:personalize:us-
west-2:123456789012:dataset-group/my-dataset-group-name",
               "RecipeArn": "arn:aws:personalize:::recipe/aws-user-personalization",
               "SolutionConfig": {
                  "EventValueThreshold" : ".05"
                }
            }
         }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MySolution:
    Type: 'AWS::Personalize::Solution'
    Properties:
      Name: my-solution-name
      DatasetGroupArn: >-
        arn:aws:personalize:us-west-2:123456789012:dataset-group/my-dataset-group-
name
      RecipeArn: 'arn:aws:personalize:::recipe/aws-user-personalization'
      SolutionConfig:
        EventValueThreshold: '.05'
```

AWS CloudFormation の詳細はこちら

AWS CloudFormation の詳細については、以下のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- AWS CloudFormation API リファレンス
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

Amazon Personalize のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。 AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ — クラウドで AWS AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。Amazon Personalize は、データ暗号化を使用してデータを保護します。詳細については、「データ暗号化」を参照してください。コンプライアンスAWS プログラムコンプライアンスプログラム の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。Amazon Personalize に適用するコンプライアンスプログラムの詳細については、コンプライアンスプログラムによるAWS 対象範囲内のサービスのを参照してください。
- クラウドのセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon Personalize 使用時における責任共有モデルの適用法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目的を満たすように Amazon Personalize を設定する方法について説明します。また、Amazon Personalize リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

- Amazon Personalize でのデータ保護
- Amazon Personalize のための ID とアクセス管理
- Amazon Personalize でのログ記録とモニタリング
- Amazon Personalize のコンプライアンス検証
- Amazon Personalize の耐障害性
- Amazon Personalize のインフラストラクチャセキュリティ
- Amazon Personalize とインターフェイス VPC エンドポイント (AWS PrivateLink)

Amazon Personalize でのデータ保護

責任 AWS 共有モデル、Amazon Personalize でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon Personalize AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

データ暗号化

次の情報は、Amazon Personalize がデータを暗号化して保護する場所について説明しています。

保管中の暗号化

Amazon Personalize に保存されているデータはすべて、Amazon Personalize が管理する AWS Key Management Service（ AWS KMS) キーで保管時に常に暗号化されます。 AWS KMS リソースの作成時に独自のキーを指定すると、Amazon Personalize はそのキーを使用してデータを暗号化し、保存します。例えば、CreateDatasetGroup オペレーションで AWS KMS ARN を指定すると、Amazon Personalize はキーを使用して、インポートしたデータを暗号化し、そのデータセットグループで作成したデータセットに保存します。

Amazon Personalize と Amazon Personalize IAM サービスロールにキーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

Amazon S3 の暗号化の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「暗号化を使用したデータの保護」を参照してください。独自の AWS KMS キーの管理については、「 AWS Key Management Service デベロッパーガイド」の「キーの管理」を参照してください。

転送中の暗号化

Amazon Personalize は、 AWS 証明書で TLS を使用して、他の AWS サービスに送信されるデータを暗号化します。他のサービスとの通信は HTTPS 経由で AWS 行われ、Amazon Personalize エンドポイントは HTTPS 経由の安全な接続のみをサポートします。

Amazon Personalize は、アカウントからデータをコピーし、内部 AWS システムで処理します。データを処理するとき、Amazon Personalize は Amazon Personalize AWS KMS キーまたは指定した AWS KMS キーを使用してデータを暗号化します。

キー管理

AWS は、デフォルト AWS KMS キーをすべて管理します。所有する AWS KMS キーを管理するのはユーザーの責任です。Amazon Personalize と Amazon Personalize IAM サービスロールにキーを使用するためのアクセス許可を付与する必要があります。詳細については、「AWS KMS キーを使用するアクセス許可を Amazon Personalize に付与する」を参照してください。

独自の AWS KMS キーの管理については、「 AWS Key Management Service デベロッパーガイド」の「キーの管理」を参照してください。

Amazon Personalize のための ID とアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に Amazon Personalize リソースの使用を承認する (許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon Personalize で行う作業によって異なります。

サービスユーザー – ジョブを実行するために Amazon Personalize サービスを使用する場合は、管理者から必要な許可と認証情報が与えられます。さらに多くの Amazon Personalize 機能を使用して作業を行う場合は、追加の許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Amazon Personalize の機能にアクセスできない場合は、「Amazon Personalize アイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon Personalize リソースを担当している場合は、Amazon Personalize に対する完全なアクセス権があると思われます。サービスのユーザーがどの Amazon Personalize 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で Amazon Personalize と IAM を併用する方法の詳細については、「IAM が Amazon Personalize で機能する方法」を参照してください。

IAM 管理者 – IAM 管理者には、Amazon Personalize へのアクセスを管理するポリシーの作成方法の詳細を理解することが推奨されます。IAM で使用できる Amazon Personalize のアイデンティティ

ベースポリシーの例を確認するには、「Amazon Personalize のアイデンティティベースのポリシーの例」を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 にサインインする方法 AWS アカウントAWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストに

ついては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「IAM Identity Center とは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイド の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳

細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) AWS がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限によ

り、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プ

リンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーで IAM の AWS マネージドポリシーを使用することはできません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポ

リシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

IAM が Amazon Personalize で機能する方法

IAM を使用して Amazon Personalize へのアクセスを管理する前に、Amazon Personalize で使用できる IAM 機能について理解しておく必要があります。

Amazon Personalize で使用できる IAM の機能

IAM 機能	Amazon Personalize のサポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	No
ABAC (ポリシー内のタグ)	はい
一時的な認証情報	Yes
プリンシパル権限	Yes
サービスロール	あり

IAM 機能	Amazon Personalize のサポート
サービスリンクロール	いいえ

Amazon Personalize およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

Amazon Personalize のアイデンティティベースのポリシー

アイデンティティベースポリシーをサポートする	Yes

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon Personalize のアイデンティティベースのポリシーの例

Amazon Personalize のアイデンティティベースポリシーの例を確認するには、「Amazon Personalize のアイデンティティベースのポリシーの例」を参照してください。

Amazon Personalize のリソースベースのポリシー

リソースベースのポリシーのサポート	No

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があ

げられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。

Amazon Personalize のポリシーアクション

ポリシーアクションに対するサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon Personalize アクションのリストを確認するには、「サービス認証リファレンス」の「サービス認証リファレンス」を参照してください。

Amazon Personalize のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
personalize
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "personalize:action1",
      "personalize:action2"
         ]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "personalize:Describe*"
```

Amazon Personalize のアイデンティティベースポリシーの例を確認するには、「Amazon Personalize のアイデンティティベースのポリシーの例」を参照してください。

Amazon Personalize ポリシーリソース

ポリシーリソースに対するサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon Personalize リソースのタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Amazon Personalize で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon Personalize で定義されるアクション」を参照してください。

Amazon Personalize のアイデンティティベースポリシーの例を確認するには、「Amazon Personalize のアイデンティティベースのポリシーの例」を参照してください。

Amazon Personalize の条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定するか、1 つの `Condition` 要素に複数のキーを指定すると、 AWS は `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

Amazon Personalize での条件キーの一覧については、「サービス認証リファレンス」の「Amazon Personalize の条件キー」を参照してください。どのアクションおよびリソースと条件キーを使用できるかについては、「Amazon Personalize で定義されるアクション」を参照してください。

Amazon Personalize のアイデンティティベースポリシーの例を確認するには、「Amazon Personalize のアイデンティティベースのポリシーの例」を参照してください。

Amazon Personalize での ACL

ACL のサポート	No

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon Personalize での ABAC

ABAC のサポート (ポリシー内のタグ)	はい

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義するアクセス許可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、IAM ユーザーガイド の「ABAC とは?」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセス制御 (ABAC) を使用する」を参照してください。

Amazon Personalize リソースのタグ付けの詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。

リソースのタグに基づいてリソースへのアクセスを制限するためのアイデンティティベースポリシーの例を表示するには、「IAMポリシーでタグを使用する」を参照してください。

Amazon Personalize での一時的な認証情報の使用

一時的な認証情報のサポート	はい

一部の は、一時的な認証情報を使用してサインインすると機能 AWS のサービス しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。この際、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Amazon Personalize のクロスサービスプリンシパル許可

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon Personalize のサービスロール

サービスロールに対するサポート	あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、Amazon Personalize の機能が破損する可能性があります。Amazon Personalize が指示する場合以外は、サービスロールを編集しないでください。

Amazon Personalize のサービスリンクロール

サービスにリンクされたロールのサポート	いいえ

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の中から、Service-linked role (サービスにリンクされたロール) 列に Yes と記載されたサービスを見つけます。サービスリンクロールに関するドキュメントをサービスで表示するには、はい リンクを選択します。

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出

し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐために、 AWS には、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールが用意されています。

リソースポリシー内では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、Amazon Personalize が別のサービスに付与する、リソースへのアクセス許可を制限することをお勧めします。

Amazon Personalize が引き受けるロールで混乱した代理問題が発生するのを防ぐため、ロールの信頼ポリシーでの値を「aws:SourceArn ~ arn:aws:personalize:*region*:*accountNumber*:*」に設定します。ワイルドカード (*) は、すべての Amazon Personalize リソースに条件を適用します。

次の信頼関係ポリシーは、「混乱した代理」問題を防ぐために、Amazon Personalize にリソースへのアクセスを許可し、aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストを使用しています。Amazon Personalize (Amazon Personalize 向けの IAM ロールの作成) のロールを作成するときには、このポリシーを使用してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "personalize.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "accountNumber"
        },
        "StringLike": {
          "aws:SourceArn": "arn:aws:personalize:region:accountNumber:*"
        }
      }
    }
  ]
```

```
}
```

Amazon Personalize のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーおよびロールには Amazon Personalize リソースを作成または変更する許可はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

Amazon Personalize が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、サービス認証リファレンスの「Amazon Personalize のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- AWS マネージドポリシー
- Amazon Personalize コンソールの使用
- 自分の権限の表示をユーザーに許可する
- Amazon Personalize のリソースに対するフルアクセスの許可
- Amazon Personalize のリソースに対する読み取り専用アクセスの許可

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon Personalize リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カ

スタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。

- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素: 条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

AWS マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるポリシーです AWS。Amazon Personalize の使用時に使用できる AWS マネージドポリシーの例を次に示します。

AmazonPersonalizeFullAccess ポリシー

AWS 管理AmazonPersonalizeFullAccessポリシーを使用して、ユーザーに次のアクセス許可を付与できます。

- すべての Amazon Personalize リソースにアクセスする
- Amazon でのメトリクスの公開と一覧表示 CloudWatch
- バケット名に `Personalize` あるいは `personalize` が含まれる Amazon S3 バケット内のすべてのオブジェクトをリスト化、読み込み、書き込み、削除します
- Amazon Personalize にロールを渡す

`AmazonPersonalizeFullAccess` は必要以上のアクセス許可を提供します。必要な許可のみを付与する新しい IAM ポリシーを作成することをお勧めします (「Amazon Personalize にリソースへのアクセス許可を付与する」を参照)。

CloudWatchFullAccess

を使用して Amazon Personalize をモニタリングするアクセス許可をユーザーに付与するには CloudWatch、`CloudWatchFullAccess`ポリシーをロールにアタッチします。詳細については、「Amazon Personalize のモニタリング」を参照してください。

以下の `CloudWatchFullAccess` ポリシーは、次のアクションに対する許可を付与します。

- での Amazon Personalize メトリクスの公開と一覧表示 CloudWatch
- メトリクスとメトリクス統計を表示します。
- メトリクスベースのアラームを設定します。

Amazon Personalize コンソールの使用

Amazon Personalize コンソールにアクセスするには、許可の最小限のセットが必要です。これらのアクセス許可により、 の Amazon Personalize リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、

または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Personalize のリソースに対するフルアクセスの許可

次の例では、 AWS アカウントの IAM ユーザーに、すべての Amazon Personalize リソースとアクションへのフルアクセスを許可します。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Personalize のリソースに対する読み取り専用アクセスの許可

この例では、 AWS アカウントの IAM ユーザーに、Amazon Personalize データセット、データセットグループ、ソリューション、キャンペーンなどの Amazon Personalize リソースへの読み取り専用アクセスを許可します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:DescribeAlgorithm",
                "personalize:DescribeBatchInferenceJob",
                "personalize:DescribeBatchSegmentJob",
                "personalize:DescribeCampaign",
                "personalize:DescribeDataset",
                "personalize:DescribeDatasetExportJob",
                "personalize:DescribeDatasetGroup",
                "personalize:DescribeDatasetImportJob",
                "personalize:DescribeEventTracker",
                "personalize:DescribeFeatureTransformation",
                "personalize:DescribeFilter",
                "personalize:DescribeRecipe",
                "personalize:DescribeRecommender",
                "personalize:DescribeSchema",
                "personalize:DescribeSolution",
                "personalize:DescribeSolutionVersion",
                "personalize:GetSolutionMetrics",
                "personalize:ListBatchInferenceJobs",
                "personalize:ListBatchSegmentJobs",
                "personalize:ListCampaigns",
```

```
                "personalize:ListDatasetExportJobs",
                "personalize:ListDatasetGroups",
                "personalize:ListDatasetImportJobs",
                "personalize:ListDatasets",
                "personalize:ListEventTrackers",
                "personalize:ListFilters",
                "personalize:ListRecipes",
                "personalize:ListRecommenders",
                "personalize:ListSchemas",
                "personalize:ListSolutions",
                "personalize:ListSolutionVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Personalize アイデンティティとアクセスのトラブルシューティング

以下の情報を使用して、Amazon Personalize と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立てます。

トピック

- Amazon Personalize でアクションを実行する権限がない
- iam を実行する権限がありません。PassRole
- 自分の 以外のユーザーに Amazon Personalize リソース AWS アカウント へのアクセスを許可したい

Amazon Personalize でアクションを実行する権限がない

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なpersonalize:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 personalize:GetWidget on resource: my-example-widget
```

この場合、personalize:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon Personalize にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して Amazon Personalize でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon Personalize リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon Personalize がこれらの機能をサポートしているかどうかを確認するには、「IAM が Amazon Personalize で機能する方法」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部認証されたユーザーへのアクセスの提供 (ID フェデレーション) を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

Amazon Personalize でのログ記録とモニタリング

このセクションでは、Amazon Personalize を Amazon CloudWatch および でモニタリングおよびログ記録する方法について説明します AWS CloudTrail。

トピック

- Amazon Personalize のモニタリング
- CloudWatch Amazon Personalize の メトリクス
- を使用した Amazon Personalize API コールのログ記録 AWS CloudTrail

Amazon Personalize のモニタリング

Amazon では CloudWatch、Amazon Personalize に関連付けられたメトリクスを取得できます。これらのメトリクスのうち 1 つ以上が定義されたしきい値から外れるときに通知を行うようにアラームを設定できます。メトリクスを表示するには、Amazon CloudWatch、Amazon AWS Command Line Interface、または CloudWatch API を使用できます。

トピック

- Amazon Personalize の CloudWatch メトリクスの使用

- Amazon Personalize メトリクスへのアクセス
- アラームを作成する
- Amazon Personalize のサーバーレスモニタリングアプリケーションの例

Amazon Personalize の CloudWatch メトリクスの使用

メトリクスを使用するには、以下の情報を指定する必要があります。

- メトリクスの名前。
- メトリクスディメンション。ディメンションは、メトリクスを一意に識別するための名前と値のペアです。

Amazon Personalize のモニタリングデータは AWS Management Console、、 AWS CLI、または CloudWatch API を使用して取得できます。 AWS SDKsまたは CloudWatch API ツールのいずれかを使用して CloudWatch API を使用することもできます。コンソールには、 CloudWatch API の raw データに基づいて一連のグラフが表示されます。必要に応じて、コンソールに表示されるグラフまたは API から取得したグラフを使用できます。

以下のリストは、メトリクスの一般的な利用方法をいくつか示しています。ここで紹介するのは開始するための提案事項です。すべてを網羅しているわけではありません。

どうすればよいか?	関連するメトリクス
記録されたイベント数を追跡するにはどうすればよいですか。	`PutEventsRequests` メトリクスをモニタリングします。
DatasetImportJob エラーをモニタリングするにはどうすればよいですか？	`DatasetImportJobError` メトリクスを使用します。
`GetRecommendations` 呼び出しのレイテンシーをモニタリングするにはどうすればよいですか。	`GetRecommendationsLatency` メトリクスを使用します。

で Amazon Personalize をモニタリングするには、適切な CloudWatch アクセス許可が必要です CloudWatch。詳細については、「Amazon の認証とアクセスコントロール CloudWatch」を参照してください。

Amazon Personalize メトリクスへのアクセス

次の例は、コンソール、、 AWS CLIおよび CloudWatch API を使用して Amazon Personalize CloudWatch メトリクスにアクセスする方法を示しています。

メトリクスを表示するには (コンソール)

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. [Metrics] (メトリクス) を選択し、[All metrics] (すべてのメトリクス) タブを選択して、`AWS/Personalize` を選択します。
3. メトリクスディメンションを選択します。
4. リストから目的のメトリクスを選択して、グラフの期間を選択します。

一定期間中に受信したイベントのメトリクスを表示するには (CLI)。

- を開き AWS CLI 、次のコマンドを入力します。

```
aws cloudwatch get-metric-statistics \
  --metric-name PutEventsRequests \
  --start-time 2019-03-15T00:00:20Z \
  --period 3600 \
  --end-time 2019-03-16T00:00:00Z \
  --namespace AWS/Personalize \
  --dimensions Name=EventTrackerArn,Value=EventTrackerArn \
  --statistics Sum
```

この例では、指定するイベントトラッカー ARN に一定期間中に受信したイベントを示します。詳細については、「get-metric-statistics」を参照してください。

メトリクスにアクセスするには (CloudWatch API)

- `GetMetricStatistics` を呼び出します。詳細については、「Amazon CloudWatch API リファレンス」を参照してください。

アラームを作成する

CloudWatch アラームの状態が変更されたときに Amazon Simple Notification Service (Amazon SNS) メッセージを送信するアラームを作成できます。1 つのアラームで、指定した期間中、1 つのメトリクスを監視します。アラームは、指定された複数の期間にわたるしきい値とメトリクスの値の関係性に基づき、1 つ以上のアクションを実行します。アクションは、Amazon SNS のトピックまたは AWS Auto Scaling のポリシーに送信される通知です。

アラームは、持続する状態変化に対してのみアクションを呼び出します。 CloudWatch アラームは、特定の状態にあるという理由だけではアクションを呼び出しません。状態が変って、変わった状態が指定期間にわたって維持される必要があります。

アラームを設定するには (コンソール)

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Alarms] を選択し、[Create alarm] を選択します。これにより、[Create Alarm Wizard] が起動します。
3. [メトリクスの選択] を選択します。
4. [All metrics] (すべてのメトリクス) タブで、`AWS/Personalize` を選択します。
5. EventTrackerArn を選択し、PutEventsRequests メトリクスを選択します。
6. [グラフ化したメトリクス] タブを選択します。
7. [統計] で、[合計] を選択します。
8. [メトリクスの選択] を選択します。
9. [名前] と [説明] を入力します。[次の時] で、[>] を選択し、任意の最大値を入力します。
10. アラーム状態に達したときに E CloudWatch メールを送信する場合は、「このアラームが発生するたびに」で「状態は ALARM 」を選択します。既存の Amazon SNS トピックにアラームを送信するには、[通知の送信先:] で既存の SNS トピックを選択します。新しい E メールサブスクリプションリストの名前と E メールアドレスを設定するには、「新規リスト」を選択します。リスト CloudWatch を保存して フィールドに表示し、将来のアラームの設定に使用できます。

> **Note**
>
> [新しいリスト] を使用して新しい Amazon SNS トピックを作成する場合は、宛先に通知を送信する前にメールアドレスを検証する必要があります。Amazon SNS は、アラーム

がアラーム状態になったときにのみメールを送信します。アラーム状態になったときにメールアドレスの検証がまだ完了していない場合、宛先には通知が届きません。

11. [アラームを作成] を選択します。

アラームを設定するには (AWS CLI)

- を開き AWS CLI、次のコマンドを入力します。alarm-actions パラメータの値を変更して、作成済みの Amazon SNS トピックを参照します。

```
aws cloudwatch put-metric-alarm \
    --alarm-name PersonalizeCLI \
    --alarm-description "Alarm when more than 10 events occur" \
    --metric-name PutEventsRequests \
    --namespace  AWS/Personalize \
    --statistic Sum \
    --period 300 \
    --threshold 10 \
    --comparison-operator GreaterThanThreshold \
    --evaluation-periods 1 \
    --unit Count \
    --dimensions Name=EventTrackerArn,Value=EventTrackerArn \
    --alarm-actions SNSTopicArn
```

この例では、指定するイベントトラッカーで 5 分以内に 10 回以上発生したイベントへのアラームを作成する方法を示しています。詳細については、「put-metric-alarm」を参照してください。

アラームを設定するには (CloudWatch API)

- PutMetricAlarm を呼び出します。詳細については、「Amazon CloudWatch API リファレンス」を参照してください。

Amazon Personalize のサーバーレスモニタリングアプリケーションの例

Amazon Personalize のモニタリング、アラート、および最適化の機能を追加するアプリケーションの例については、Amazon Personalize サンプルリポジトリの「Amazon Personalize のモニタリング」を参照してください。

CloudWatch Amazon Personalize の メトリクス

このセクションでは、Amazon Personalize で使用できる Amazon CloudWatch メトリクスについて説明します。詳細については、「Amazon Personalize のモニタリング」を参照してください。

次の表には、Amazon Personalize のメトリクスが一覧表示されています。 GetRecommendations および GetPersonalizedRanking を除くすべてのメトリクスは、次の統計をサポートします。Average, Minimum, Maximum, Sum. GetRecommendations and GetPersonalizedRanking サポートSumのみ。

メトリクス	説明
DatasetImportJobRequests	成功した CreateDatasetImportJob API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn, DatasetImportJobArn
DatasetImportJobError	エラーが発生した CreateDatasetImportJob API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn, DatasetImportJobArn
DatasetImportJobExecutionTime	CreateDatasetImportJob API 呼び出しからオペレーション完了 (または失敗) までの時間。 ディメンション: DatasetGroupArn, DatasetArn, DatasetImportJobArn 単位: 秒
DatasetSize	データセットインポートジョブによってインポートされるデータのサイズ。 ディメンション: DatasetGroupArn, DatasetArn, DatasetImportJobArn 単位: バイト

メトリクス	説明
SolutionTrainingJobRequests	成功した CreateSolutionVersion API 呼び出しの数。 ディメンション: SolutionArn, SolutionVersionArn
SolutionTrainingJobError	エラーが発生した CreateSolutionVersion API 呼び出しの数。 ディメンション: SolutionArn, SolutionVersionArn
SolutionTrainingJobExecutionTime	CreateSolutionVersion API 呼び出しからオペレーション完了 (または失敗) までの時間。 ディメンション: SolutionArn, SolutionVersionArn 単位: 秒
GetPersonalizedRanking	API コールが成功したかどうか。GetPersonalizedRanking統計を使用して、成功した GetPersonalizedRanking API sum コールの合計数を表示します。このメトリクスは他の統計をサポートしていません。 ディメンション: CampaignArn
GetPersonalizedRanking4xxErrors	4xx HTTP 応答コードを返した GetPersonalizedRanking API 呼び出しの数。 ディメンション: CampaignArn
GetPersonalizedRanking5xxErrors	5xx HTTP 応答コードを返した GetPersonalizedRanking 呼び出しの数。 ディメンション: CampaignArn

メトリクス	説明
GetPersonalizedRankingLatency	GetPersonalizedRanking API 呼び出しを受信してからレコメンデーションを送信するまでの時間 (4xx および 5xx エラーを除く)。 ディメンション: CampaignArn 単位: ミリ秒
GetRecommendations	GetRecommendations API コールが成功したかどうか。統計を使用して、成功した GetRecommendations API sum コールの合計数を表示します。このメトリクスは他の統計をサポートしていません。 ディメンション: CampaignArn
GetRecommendations4xxErrors	4xx HTTP 応答コードを返した GetRecommendations API 呼び出しの数。 ディメンション: CampaignArn
GetRecommendations5xxErrors	5xx HTTP 応答コードを返した GetRecommendations 呼び出しの数。 ディメンション： CampaignArn
GetRecommendationsLatency	GetRecommendations API 呼び出しを受信してからレコメンデーションを送信するまでの時間 (4xx および 5xx エラーを除く)。 ディメンション: CampaignArn 単位: ミリ秒
PutEventsRequests	成功した PutEvents API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn, EventTrackerArn

メトリクス	説明
PutEvents4xxErrors	4xx HTTP 応答コードを返した PutEvents API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn, EventTrackerArn
PutEvents5xxErrors	5xx HTTP 応答コードを返した PutEvents 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn, EventTrackerArn
PutEventLatency	PutEvents API 呼び出しの完了にかかった時間 (4xx および 5xx エラーを除く)。 ディメンション: DatasetGroupArn, DatasetArn, EventTrackerArn 単位: ミリ秒
PutItemsRequests	成功した PutItems API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn
PutItems4xxErrors	4xx HTTP 応答コードを返した PutItems API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn
PutItems5xxErrors	5xx HTTP 応答コードを返した PutItems 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn

メトリクス	説明
PutItemsLatency	PutItems API 呼び出しの完了にかかった時間 (4xx および 5xx エラーを除く)。 ディメンション: DatasetGroupArn, DatasetArn 単位: ミリ秒
PutUsersRequests	成功した PutUsers API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn
PutUsers4xxErrors	4xx HTTP 応答コードを返した PutUsers API 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn
PutUsers5xxErrors	5xx HTTP 応答コードを返した PutUsers 呼び出しの数。 ディメンション: DatasetGroupArn, DatasetArn
PutUsersLatency	PutUsers API 呼び出しの完了にかかった時間 (4xx および 5xx エラーを除く)。 ディメンション: DatasetGroupArn, DatasetArn 単位: ミリ秒

を使用した Amazon Personalize API コールのログ記録 AWS CloudTrail

Amazon Personalize は AWS CloudTrail、Amazon Personalize のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。は、Amazon Personalize コンソールからの呼び出しや Amazon Personalize API へのコード呼び出しを含む、Amazon Personalize の API 呼び出しのサブセットをイベントとして CloudTrail キャプチャします。 APIs 証跡を作成する場合は、Amazon Personalize の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。 Amazon S3 証跡を設定しない場合でも、 CloudTrail コンソールのイベント履歴 で最新のイベントを表示できま

す。で収集された情報を使用して CloudTrail、Amazon Personalize に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の設定と有効化の方法など CloudTrail、 の詳細については、AWS CloudTrail 「 ユーザーガイド」を参照してください。

の Amazon Personalize 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。Amazon Personalize でサポートされているイベントアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴 を含む CloudTrail イベントの表示」を参照してください。

Amazon Personalize のイベントなど、 AWS アカウント内のイベントの継続的な記録については、証跡を作成します。証跡により CloudTrail 、 はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのリージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベントデータをより詳細に分析し、それに基づいて行動するように他の AWS サービスを設定できます。詳細については、以下をご覧ください。

- 証跡を作成するための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

Amazon Personalize は、すべてのアクション (API オペレーション) をイベントとして CloudTrail ログファイルに記録することをサポートしています。詳細については、「アクション」を参照してください。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。ID 情報は次の判断に役立ちます。

- リクエストが、ルートとユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーションユーザーの一時的なセキュリティ認証情報のどちらを使用して送信されたか。

- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、CloudTrail userIdentity 要素」を参照してください。

例: Amazon Personalize ログファイルエントリ

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、 ListDatasetGroups API オペレーションの アクションを含む CloudTrail ログエントリを示しています。 ListDatasetGroups API オペレーションは状態を変更しないアクションであるため、`responseElements`レスポンスは null であることに注意してください。 CloudTrail レコードの本文の詳細については、CloudTrail 「レコードの内容」を参照してください。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "principal-id",
        "arn": "arn:aws:iam::user-arn",
        "accountId": "account-id",
        "accessKeyId": "access-key",
        "userName": "user-name"
    },
    "eventTime": "2018-11-22T02:18:03Z",
    "eventSource": "personalize.amazonaws.com",
    "eventName": "ListDatasetGroups",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "source-ip-address",
    "userAgent": "aws-cli/1.11.16 Python/2.7.11 Darwin/15.6.0 botocore/1.4.73",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "request-id",
    "eventID": "event-id",
    "eventType": "AwsApiCall",
    "recipientAccountId": "recipient-account-id"
}
```

Amazon Personalize のコンプライアンス検証

サードパーティーの監査者は、複数の コンプライアンスプログラムの一環として Amazon Personalize のセキュリティと AWS コンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA などを含みます。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスAWS プログラムによる対象範囲内のサービス」を参照してください。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポート AWS Artifactのダウンロード」の」を参照してください。

Amazon Personalize を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用される法律および規制によって決まります。 AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を にデプロイする手順について説明します AWS。
- HIPAA セキュリティとコンプライアンスのアーキテクチャの設計ホワイトペーパー — を使用して、米国の医療保険の相互運用性と説明責任に関する法律 (HIPAA) で規制されている機密性の高いワークロード AWS を実行する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」のルールによるリソースの評価 — この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

Amazon Personalize の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。

アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

Amazon Personalize は、データの耐障害性のために AWS グローバルインフラストラクチャを活用します。 AWS リージョンに Amazon Personalize リソースを作成すると、Amazon Personalize は複数のアベイラビリティーゾーンにまたがるリソースの耐障害性とデータ冗長性を管理します。Amazon Personalize リソースを作成できる AWS リージョンのリストについては、「Amazon Web Services 全般のリファレンス」のAWS 「リージョンとエンドポイント」を参照してください。AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Amazon Personalize のインフラストラクチャセキュリティ

マネージドサービスである Amazon Personalize は AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Amazon Personalize にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Personalize とインターフェイス VPC エンドポイント (AWS PrivateLink)

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合、VPC と Amazon Personalize の間にプライベート接続を確立できます。この接続を使用すると、Amazon Personalize はパブリックインターネットを経由せずに、VPC のリソースと通信できます。

Amazon VPC は AWS のサービス 、ユーザーが定義した仮想プライベートクラウド (VPC) または仮想ネットワークで AWS リソースを起動するために使用する です。VPC を使用することで、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC エンドポイントでは、 AWS ネットワークは VPC と 間のルーティングを処理します AWS のサービス。

VPC を Amazon Personalize に接続するには、Amazon Personalize に対してインターフェイス VPC エンドポイントを定義します。インターフェイスエンドポイントは、サポートされる AWS のサービスを宛先とするトラフィックのエントリポイントとなるプライベート IP アドレスを持つ Elastic Network Interface です。エンドポイントは、Amazon Personalize への信頼性の高いスケーラブルな接続を提供します。インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、または VPN 接続は必要ありません。詳細については、「Amazon VPC ユーザーガイド」の「Amazon VPC とは」を参照してください。

インターフェイス VPC エンドポイントは によって有効になります AWS PrivateLink。この AWS テクノロジーにより、プライベート IP アドレスを持つ Elastic Network Interface AWS のサービス を使用して、 間のプライベート通信が可能になります。

> **Note**
>
> すべての Amazon Personalize 連邦情報処理規格 (FIPS) エンドポイントは、 でサポートされています AWS PrivateLink。

トピック

- Amazon Personalize 用のインターフェイス VPC エンドポイントの作成
- Amazon Personalize 用の VPC エンドポイントポリシーの作成

Amazon Personalize 用のインターフェイス VPC エンドポイントの作成

Amazon Personalize サービスの VPC エンドポイントは、Amazon VPC コンソールまたは AWS Command Line Interface () を使用して作成できますAWS CLI。詳細については、「Amazon VPC ユーザーガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

Amazon Personalize の VPC エンドポイントを作成するには、サービス用に次のいずれかを選択します。

- com.amazonaws.*region*.personalize
- com.amazonaws.*region*.personalize-events
- com.amazonaws.*region*.personalize-runtime

エンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名 (`personalize.us-east-1.amazonaws.com` など) を使用して、Amazon Personalize への API リクエストを実行できます。

Amazon Personalize 用の VPC エンドポイントポリシーの作成

VPC エンドポイントに Amazon Personalize へのアクセスをコントロールするエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

例:Amazon Personalize アクションと passRole アクションを許可する VPC エンドポイントポリシー

このポリシーは、エンドポイントにアタッチされると、すべての Amazon Personalize アクションと passRole アクションへのアクセスを許可します。

```
{
    "Statement": [
```

```
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "personalize:*",
                "iam:PassRole"
            ],
            "Resource": "*"
        }
    ]
}
```

例: Amazon Personalize ListDatasets アクションを許可する VPC エンドポイントポリシー

このポリシーは、エンドポイントにアタッチされると、リストされている Amazon Personalize ListDatasets アクションへのアクセスを許可します。

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "personalize:ListDatasets"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Personalize エンドポイントとクォータ

次のセクションには、Amazon Personalize のガイドライン、クォータ、およびエンドポイントに関する情報が含まれています。調整可能なクォータの場合、Service Quotas コンソールを使用してクォータの引き上げをリクエストできます。詳細については、「クォータ引き上げのリクエスト」を参照してください。

トピック

Amazon Personalize のエンドポイントとリージョン

リージョンごとの Amazon Personalize のエンドポイントのリストについては、アマゾン ウェブ サービスの全般のリファレンスの「AWS リージョンとエンドポイント」を参照してください。

コンプライアンス

Amazon Personalize のコンプライアンスプログラムの詳細については、「AWS コンプライアンス」、「AWS コンプライアンスプログラム」、および「コンプライアンスプログラムによる対象範囲内のAWS のサービス」を参照してください。

Service Quotas

AWS アカウントには、Amazon Personalize の次のクォータがあります。

リソース	クォータ
Item interactions	
ソリューションバージョンまたはレコメンダーの作成に必要な一意のアイテムインタラクションの最小数。カスタムソリューションの場合、	1,000

リソース	クォータ
トレーニング前にイベントタイプまたはイベント値でフィルタリングした後に、この数のレコードが必要です。	
User-Personalization-v2 および Personalized-Ranking-v2 レシピの場合、トレーニング中にモデルによって考慮されるアイテムインタラクションの最大数。	30 億
User-Personalization-v2 または Personalized-Ranking-v2 以外のすべてのドメインユースケースとカスタムレシピでは、トレーニング中にモデルによって考慮されるアイテムインタラクションの最大数。	5 億 (調整可能)
アイテムインタラクションデータセット内のオプションのメタデータ列の合計数と組み合わせた個別のイベントタイプの最大数。	10
アイテムインタラクションデータセット内の予約済みフィールドを除くメタデータ列の最大数。	5
カテゴリ別データとインプレッション値の最大文字数。	1,000
フルインポートモードでのデータセットインポートジョブあたりのバルクアイテムインタラクションデータの最大量。	100 GB (モデル で考慮されるアイテムインタラクションの増加に伴って 1TB に増加)
INCREMENTAL インポートモードでのデータセットインポートジョブあたりのバルクアイテムインタラクションデータの最大量。	1 GB
FULL または INCREMENTAL インポートモードのデータセットインポートジョブあたりのアイテムインタラクションレコードの最小数。	1,000

リソース	クォータ
Users	
ドメインレコメンダーまたはカスタムソリューションバージョンを作成するために必要な、アイテムインタラクションデータ内の最小ユニークユーザー数。各ユーザーには少なくとも 2 つのアイテムインタラクションが必要です。	25
ドメインレコメンダーまたはカスタムソリューションバージョンを作成する前に、少なくとも 2 つ以上のアイテムインタラクションを持つ必要があるユーザー全体の最小割合。	1%
Users データセットのメタデータフィールドの最大数。	25
USER_ID データ値の最大文字数。	256
カテゴリ別データ値の最大文字数。	1000 文字
フルインポートモードでのデータセットインポートジョブあたりのバルクユーザーデータの最大量。	100 GB
INCREMENTAL インポートモードでのデータセットインポートジョブあたりのバルクユーザーデータの最大量。	1 GB
Items	
User-Personalization-v2 または Personalized-Ranking-v2 の場合、トレーニング中にモデルによって考慮される項目の最大数。これらの項目は、アイテムとアイテムインタラクションデータセットの両方からのものです。	500 万件

リソース	クォータ
User-Personalization-v2 と Personalized-Ranking-v2 以外のすべてのドメインユースケースとカスタムレシピでは、トレーニング中およびレコメンデーションの生成中にモデルによって考慮されるアイテムの最大数。	750,000
Items データセットのメタデータフィールドの最大数。	100
ITEM_ID データ値の最大文字数。	256
カテゴリ別データ値の最大文字数。	1000 文字
Items データセットのテキストフィールドの最大数。	1
中国語と日本語のテキストデータ値の最大文字数。	7,000 文字
他のすべての言語のテキストデータ値の最大文字数。	20,000 文字
BULK インポートモードでのデータセットインポートジョブあたりのバルクアイテムデータの最大量。	100 GB
INCREMENTAL インポートモードでのデータセットインポートジョブあたりのバルクアイテムデータの最大量。	1 GB
Actions	
トレーニング中およびレコメンデーションの生成中にモデルによって考慮されるアクションの最大数。	1,000
Actions データセットのメタデータフィールドの最大数。	10

リソース	クォータ
ACTION_ID データ値の最大文字数。	256
カテゴリ別データ値の最大文字数。	1000 文字
BULK インポートモードでのデータセットインポートジョブあたりの一括アクションデータの最大量。	100 GB
INCREMENTAL インポートモードでのデータセットインポートジョブあたりのバルクアクションデータの最大量。	1 GB
Action interactions	
トレーニング中にモデルによって考慮されるアクションインタラクションの最大数。	5 億
Action インタラクションデータセット内の予約済みフィールドを除くメタデータ列の最大数。	5
フルインポートモードでのデータセットインポートジョブあたりのバルクインタラクションデータの最大量。	100 GB (モデル で考慮されるアクションアイテムインタラクションの増加に伴って 1TB に増加)
INCREMENTAL インポートモードでのデータセットインポートジョブあたりのバルクインタラクションデータの最大量。	1 GB
Individual record import APIs	
データセットグループあたりの`PutEvents` リクエストの最大レート。	1000/秒
`PutEvents` 呼び出しにおけるイベントの最大数。	10
イベントの最大サイズ。	10 KB

リソース	クォータ
データセットグループあたりのPutAction Interactions　リクエストの最大レート。	1000/秒
PutActionInteractions　呼び出し内のアクションインタラクションイベントの最大数。	10
アクションインタラクションイベントの最大サイズ。	10 KB
データセットグループあたりのPutItemsリクエストの最大レート。	10/秒
PutItems 通話内の項目の最大数。	10
データセットグループあたりのPutUsersリクエストの最大レート。	10/秒
PutUsers 呼び出しの最大ユーザー数。	10
データセットグループあたりのPutActions リクエストの最大レート。	10/秒
PutActions　通話の最大ユーザー数。	10
Legacy recipes	
HRNN-metadata および HRNN-Coldstart レシピの Users and Items データセットの合計データの最大量。	5 GB
HRNN-Coldstart レシピがモデルをトレーニング (ソリューションバージョンを作成) するためにサポートするコールドスタートアイテムの最大数。	80000
HRNN-Coldstart レシピがモデルのトレーニング (ソリューションバージョンの作成) に必要なコールドスタートアイテムの最小数。	100

リソース	クォータ
Filters	
データセットグループあたりのフィルターの総数。	10
フィルターの個別のデータセットフィールドの最大数。	5
データセットグループ内のすべてのフィルターにわたる個別のデータセットフィールドの合計数。	10
フィルターで考慮されるイベントタイプごとのユーザーあたりのアイテムインタラクションの最大数。	100 件のインタラクション (調整可能)
フィルターで考慮されるイベントタイプごとのユーザーあたりのアクションインタラクションの最大数。	300 件のアクションインタラクション (調整可能)
GetRecommendations / GetPersonalizedRanking / GetActionRecommendations requests	
`GetRecommendations` 、、`GetActionRecommendations` および `GetPersonalizedRanking` リクエストの最大トランザクションレート。	2500/秒
キャンペーンごとに 1 秒あたり最大 `GetRecommendations` 個のリクエスト。	500/秒
キャンペーンごとに 1 秒あたり最大 `GetActionRecommendations` 個のリクエスト。	500/秒
キャンペーンごとに 1 秒あたり最大 `GetPersonalizedRanking` 個のリクエスト。	500/秒。

リソース	クォータ
GetRecommendations または GetPersonalizedRanking リクエストあたりのメタデータ列の最大数。	10
メタデータのないGetRecommendation リクエストのレコメンデーション結果の最大数。	500
メタデータを含むGetRecommendation リクエストのレコメンデーション結果の最大数。	50
メタデータのないGetPersonalizedRanking リクエストでランク付けする項目の最大数。	500
メタデータを含むGetPersonalizedRanking リクエストでランク付けする項目の最大数。	50
Metric attribution quotas	
メトリクス属性のメトリクスの最大数	10
一意のイベント属性ソースの最大数	100
Batch inference jobs	
バッチ推論ジョブの入力ファイルの最大数。	1,000
バッチ推論ジョブの入力の最大サイズ。	1 GB
テーマのないバッチ推論ジョブの入力ファイルあたりのレコードの最大数。	5,000 万
テーマ付きバッチ推論ジョブの入力ファイルあたりのレコードの最大数。	100
Batch segment jobs	

リソース	クォータ
バッチセグメントジョブの入力ファイルの最大数。	1,000
バッチセグメントジョブ入力の最大サイズ。	1 GB
Item-Affinity レシピの入力ファイルあたりのクエリの最大数。	500
Item-Attribute-Affinity レシピの入力ファイルあたりのクエリの最大数。	10
セグメントあたりの最大ユーザー数	500 万件
Data deletion jobs	
ステータスが PENDING のデータセットグループのデータ削除ジョブの最大数。	5 (調整可能)
データ削除入力ファイルの最大合計サイズ	50 MB

AWS アカウントには、リージョンごとに次のクォータがあります。

リソース	クォータ
アクティブなスキーマの総数。	500
アクティブなデータセットグループの総数。	5 (調整可能)
保留中または進行中のデータセットのインポートジョブの合計数。	5
保留中または進行中のバッチ推論ジョブの総数。	5 (調整可能)
保留中または進行中のバッチセグメントジョブの総数。	5
保留中または進行中のソリューションバージョンの総数。	20 (調整可能)

各データセットグループには、以下のクォータがあります。

リソース	クォータ
アクティブなソリューションの総数。	10 (調整可能)
アクティブなキャンペーンの総数。	5 (調整可能)
レコメンダーの総数。	5
フィルターの総数。	10 (調整可能)
すべてのフィルターにわたる個別のデータセットフィールドの総数。	10
ステータスが PENDING のデータセットグループのデータ削除ジョブの合計数。	5 (調整可能)

クォータ引き上げのリクエスト

調整可能なクォータの場合、Service Quotas コンソールを使用してクォータの引き上げをリクエストできます。次の Amazon Personalize のクォータは調整可能です。

- トレーニング時にモデルで検討されるアイテムインタラクションの最大数。
- データセットグループごとのアクティブなキャンペーン
- アクティブなデータセットグループ
- データセットグループごとのアクティブなフィルター
- データセットグループごとのアクティブなソリューション
- 増分インポートごとのデータ量。
- フィルターで考慮されるユーザーごと、イベントタイプごとの最大アイテムインタラクション数。
- 保留中または進行中のバッチ推論ジョブの総数
- ステータスが PENDING のデータセットグループのデータ削除ジョブの合計数。
- 保留中または進行中のソリューションバージョンの総数
- `PutEvents` または `PutActionInteraction` リクエストの最大レート

クォータの引き上げをリクエストするには、Service Quotas コンソールを使用して、Service Quotas ユーザーガイドの「クォータの引き上げのリクエスト」のセクションの手順に従います。

API リファレンス

このセクションでは、Amazon Personalize API の操作に関するドキュメントを提供します。リージョンごとの Amazon Personalize のエンドポイントのリストについては、AWS 全般のリファレンスの「AWS リージョンとエンドポイント」を参照してください。

トピック

アクション

次のアクションは、Amazon Personalize でサポートされています。

- DeleteDataset
- DeleteDatasetGroup
- DeleteEventTracker
- DeleteFilter
- DeleteMetricAttribution
- DeleteRecommender
- DeleteSchema
- DeleteSolution
- DescribeAlgorithm
- DescribeBatchInferenceJob
- DescribeBatchSegmentJob
- DescribeCampaign
- DescribeDataDeletionJob
- DescribeDataset
- DescribeDatasetExportJob
- DescribeDatasetGroup
- DescribeDatasetImportJob
- DescribeEventTracker
- DescribeFeatureTransformation
- DescribeFilter
- DescribeMetricAttribution
- DescribeRecipe
- DescribeRecommender
- DescribeSchema
- DescribeSolution
- DescribeSolutionVersion
- GetSolutionMetrics
- ListBatchInferenceJobs
- ListBatchSegmentJobs
- ListCampaigns

- ListDataDeletionJobs
- ListDatasetExportJobs
- ListDatasetGroups
- ListDatasetImportJobs
- ListDatasets
- ListEventTrackers
- ListFilters
- ListMetricAttributionMetrics
- ListMetricAttributions
- ListRecipes
- ListRecommenders
- ListSchemas
- ListSolutions
- ListSolutionVersions
- ListTagsForResource
- StartRecommender
- StopRecommender
- StopSolutionVersionCreation
- TagResource
- UntagResource
- UpdateCampaign
- UpdateDataset
- UpdateMetricAttribution
- UpdateRecommender

次のアクションは、Amazon Personalize Events でサポートされています。

- PutActionInteractions
- PutActions
- PutEvents
- PutItems

- PutUsers

次のアクションは、Amazon Personalize Runtime でサポートされています。

- GetActionRecommendations
- GetPersonalizedRanking
- GetRecommendations

Amazon Personalize

次のアクションは、Amazon Personalize でサポートされています。

- CreateBatchInferenceJob
- CreateBatchSegmentJob
- CreateCampaign
- CreateDataDeletionJob
- CreateDataset
- CreateDatasetExportJob
- CreateDatasetGroup
- CreateDatasetImportJob
- CreateEventTracker
- CreateFilter
- CreateMetricAttribution
- CreateRecommender
- CreateSchema
- CreateSolution
- CreateSolutionVersion
- DeleteCampaign
- DeleteDataset
- DeleteDatasetGroup
- DeleteEventTracker
- DeleteFilter

- DeleteMetricAttribution
- DeleteRecommender
- DeleteSchema
- DeleteSolution
- DescribeAlgorithm
- DescribeBatchInferenceJob
- DescribeBatchSegmentJob
- DescribeCampaign
- DescribeDataDeletionJob
- DescribeDataset
- DescribeDatasetExportJob
- DescribeDatasetGroup
- DescribeDatasetImportJob
- DescribeEventTracker
- DescribeFeatureTransformation
- DescribeFilter
- DescribeMetricAttribution
- DescribeRecipe
- DescribeRecommender
- DescribeSchema
- DescribeSolution
- DescribeSolutionVersion
- GetSolutionMetrics
- ListBatchInferenceJobs
- ListBatchSegmentJobs
- ListCampaigns
- ListDataDeletionJobs
- ListDatasetExportJobs
- ListDatasetGroups
- ListDatasetImportJobs

- ListDatasets
- ListEventTrackers
- ListFilters
- ListMetricAttributionMetrics
- ListMetricAttributions
- ListRecipes
- ListRecommenders
- ListSchemas
- ListSolutions
- ListSolutionVersions
- ListTagsForResource
- StartRecommender
- StopRecommender
- StopSolutionVersionCreation
- TagResource
- UntagResource
- UpdateCampaign
- UpdateDataset
- UpdateMetricAttribution
- UpdateRecommender

CreateBatchInferenceJob

サービス: Amazon Personalize

Amazon S3 に保存されているアイテムまたはユーザーのリストに基づいてバッチレコメンデーションを生成し、レコメンデーションを Amazon S3 バケットにエクスポートします。

バッチレコメンデーションを生成するには、ソリューションバージョンの ARN と、入出力データの Amazon S3 URI を指定します。ユーザーパーソナライズ、人気アイテム、パーソナライズされたランキングソリューションの場合、バッチ推論ジョブによって入力ファイル内のユーザー ID ごとに推奨アイテムのリストが生成されます。関連アイテムソリューションの場合、ジョブによって入力ファイル内のアイテム ID ごとに推奨アイテムのリストが生成されます。

詳細については、「バッチ推論ジョブの作成」を参照してください。

Similar-Items レシピを使用する場合、Amazon Personalize はバッチレコメンデーションにわかりやすいテーマを追加できます。テーマを生成するには、ジョブのモードを THEME_GENERATION に設定し、入力データの項目名を含むフィールド名を指定します。

テーマの生成の詳細については、「Batch recommendations with themes from Content Generator」を参照してください。

Trending-Now レシピまたは Next-Best-Action レシピではバッチレコメンデーションを取得することはできません。

リクエストの構文

```
{
   "batchInferenceJobConfig": {
      "itemExplorationConfig": {
         "string" : "string"
      }
   },
   "batchInferenceJobMode": "string",
   "filterArn": "string",
   "jobInput": {
      "s3DataSource": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "jobName": "string",
   "jobOutput": {
```

```
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "numResults": number,
   "roleArn": "string",
   "solutionVersionArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ],
   "themeGenerationConfig": {
      "fieldsForThemeGeneration": {
         "itemName": "string"
      }
   }
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

batchInferenceJobConfig

バッチ推論ジョブの設定の詳細。

タイプ : BatchInferenceJobConfig オブジェクト

必須: いいえ

batchInferenceJobMode

バッチ推論ジョブのモード。類似アイテムのグループを説明するテーマを生成するには、ジョブモードを THEME_GENERATION に設定します。テーマを生成しない場合は、デフォルトの BATCH_INFERENCE を使用します。

テーマ付きバッチレコメンデーションを取得すると、追加コストが発生します。詳細については、「Amazon Personalize の料金」を参照してください。

型: 文字列

有効な値：BATCH_INFERENCE | THEME_GENERATION

必須：いいえ

filterArn

バッチ推論ジョブに適用するフィルターの ARN。フィルターの使用の詳細については、「バッチのレコメンデーションのフィルタリング」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

jobInput

レコメンデーションのベースとなる入力ファイルの格納場所への Amazon S3 パス。入力資料は JSON 形式である必要があります。

型: BatchInferenceJobInput オブジェクト

必須: はい

jobName

作成するバッチ推論ジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

jobOutput

ジョブの出力が保存される Amazon S3 バケットへのパス。

型: BatchInferenceJobOutput オブジェクト

必須: はい

numResults

取得するレコメンデーションの数。

タイプ: 整数

必須: いいえ

roleArn

入力および出力 Amazon S3 バケットに対してそれぞれ読み取りおよび書き込みを実行するための許可を持つ Amazon Identity and Access Management ロールの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須：はい

solutionVersionArn

バッチ推論レコメンデーションの生成に使用するソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

tags

バッチ推論ジョブに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

themeGenerationConfig

テーマ生成ジョブでは、各アイテムの名前を含むアイテムデータセット内の列の名前を指定します。

タイプ：ThemeGenerationConfig オブジェクト

必須: いいえ

レスポンスの構文

```
{
   "batchInferenceJobArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

batchInferenceJobArn

バッチ推論ジョブの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateBatchSegmentJob

サービス: Amazon Personalize

バッチセグメントジョブを作成します。この操作は最大 5,000 万レコードを処理でき、入力ファイルは JSON 形式である必要があります。詳細については、「バッチレコメンデーションとユーザーセグメントの取得」を参照してください。

リクエストの構文

```
{
   "filterArn": "string",
   "jobInput": {
      "s3DataSource": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "jobName": "string",
   "jobOutput": {
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "numResults": number,
   "roleArn": "string",
   "solutionVersionArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

filterArn

バッチセグメントジョブに適用するフィルターの ARN。フィルターの使用についての詳細は、「バッチのレコメンデーションのフィルタリング」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

jobInput

バッチセグメントジョブの生成に使用される入力データの Amazon S3 パス。

型: BatchSegmentJobInput オブジェクト

必須: はい

jobName

作成するバッチセグメントジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須：はい

jobOutput

ジョブの出力が保存されるバケットの Amazon S3 パス。

型: BatchSegmentJobOutput オブジェクト

必須: はい

numResults

入力データの各行のバッチセグメントジョブによって生成された予測ユーザーの数。セグメントあたりのユーザーの最大数は 500 万人です。

タイプ: 整数

必須: いいえ

roleArn

入力および出力 Amazon S3 バケットに対してそれぞれ読み取りおよび書き込みを実行するための許可を持つ Amazon Identity and Access Management ロールの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須：はい

solutionVersionArn

バッチセグメントジョブでバッチセグメントを生成するために使用するソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

tags

バッチセグメントジョブに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "batchSegmentJobArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

batchSegmentJobArn

バッチセグメントジョブの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateCampaign

サービス: Amazon Personalize

> **Important**
>
> キャンペーンがアクティブである間は、キャンペーン費用が発生します。不要な費用が発生しないように、終了したら必ずキャンペーンを削除してください。キャンペーン費用については、「Amazon Personalize 料金表」を参照してください。

ソリューションバージョンをデプロイするキャンペーンを作成します。クライアントがと GetPersonalizedRankingAPI GetRecommendationsを呼び出すと、リクエストにキャンペーンが指定されます。

最小プロビジョンド TPS およびオートスケーリング

> **Important**
>
> `minProvisionedTPS` の値を高く設定するとコストが増加します。最初は `minProvisionedTPS` に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、`minProvisionedTPS`必要に応じて増やします。

Amazon Personalize のキャンペーンを作成する場合、キャンペーンの 1 秒あたりのプロビジョニングされる最小トランザクション数 (`minProvisionedTPS`) を指定できます。これは Amazon Personalize によってプロビジョニングされたキャンペーンのベーストライントランザクションスループットです。キャンペーンがアクティブである間の最低請求額を設定します。トランザクションは単一の `GetRecommendations` または `GetPersonalizedRanking` リクエストです。デフォルト `minProvisionedTPS` は 1 です。

TPS が `minProvisionedTPS` を超えて増加した場合、Amazon Personalize はプロビジョンド容量を自動スケーリングしますが、`minProvisionedTPS` を下回ることはありません。容量が引き上げられている間に短時間の遅延が生じます。これにより、トランザクションの損失が生じる可能性があります。トラフィックが減少すると、容量は `minProvisionedTPS` に戻ります。

プロビジョニングされた最小 TPS、またはリクエストが `minProvisionedTPS` を超える場合は実際の TPS に対して課金されます。実際の TPS は、行ったレコメンデーションリクエストの総数です。

低い値から始めminProvisionedTPS、Amazon CloudWatch のメトリクスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて値を増やすことをお勧めします。

キャンペーンのコストの詳細については、「Amazon Personalize の料金」を参照してください。

[ステータス]

キャンペーンは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

キャンペーンのステータスを確認するには、お電話くださいDescribeCampaign。

> **Note**
>
> キャンペーンの status が ACTIVE になるまで待ってから、キャンペーンにレコメンデーションを尋ねます。

関連 API

- ListCampaigns
- DescribeCampaign
- UpdateCampaign
- DeleteCampaign

リクエストの構文

```
{
   "campaignConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "syncWithLatestSolutionVersion": boolean
   },
   "minProvisionedTPS": number,
   "name": "string",
```

```
   "solutionVersionArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignConfig

キャンペーンの設定の詳細。

タイプ : CampaignConfig オブジェクト

必須: いいえ

minProvisionedTPS

Amazon Personalize がサポートする、リクエストされた 1 秒あたりの最小プロビジョンドトランザクション (推奨) を指定します。minProvisionedTPS を高く設定すると請求額が増加します。最初は minProvisionedTPS に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて増やします。

タイプ : 整数

有効な範囲: 最小値 は 1 です。

必須: いいえ

name

新しいキャンペーンの名前。キャンペーン名はアカウント内で一意でなければなりません。

型: 文字列

長さの制限 : 最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

solutionVersionArn

キャンペーンでデプロイするトレーニング済みモデルの Amazon リソースネーム (ARN)。ソリューションの最新のソリューションバージョンを指定するには、ソリューションの ARN *SolutionArn/$LATEST* をフォーマットで指定します。syncWithLatestSolutionVersionで設定した場合は、この形式を使用する必要があります。True CampaignConfig

ソリューションの最新のソリューションバージョンではないモデルをデプロイするには、ソリューションバージョンの ARN を指定します。

キャンペーンの自動更新について詳しくは、「キャンペーンの自動更新の有効化」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

必須：はい

tags

関数に適用するタグのリスト

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "campaignArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

campaignArn

キャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateDataDeletionJob

サービス: Amazon Personalize

Amazon Personalize データセットグループから特定のユーザーへのすべての参照をバッチで削除するバッチジョブを作成します。Amazon S3 バケットの userIdsの CSV ファイルで削除するユーザーを指定します。ジョブが完了すると、Amazon Personalize はユーザーのデータに関するトレーニングを行いなくなり、ユーザーセグメントを生成する際にユーザーを考慮しなくなります。データ削除ジョブの作成の詳細については、「ユーザーの削除」を参照してください。

- 入力ファイルは、ユーザー IDs列を持つ CSV ファイルである必要があります。CSV ファイルの準備の詳細については、「データ削除ファイルの準備と Amazon S3 へのアップロード」を参照してください。
- userIds の入力 CSV ファイルにアクセスするアクセス許可を Amazon Personalize に付与するには、データソースから読み取るアクセス許可を持つ IAM サービスロールを指定する必要があります。このロールには`GetObject`、バケットとそのコンテンツに対する および アクセス`ListBucket`許可が必要です。これらのアクセス許可はデータのインポートと同じです。Amazon S3 バケットへのアクセスを許可する方法については、「Amazon Personalize に Amazon S3 リソースへのアクセスを許可する」を参照してください。

ジョブを作成した後、データセットやモデルからユーザーへのすべての参照を削除するまでに最大 1 日かかる場合があります。ジョブが完了するまで、Amazon Personalize はトレーニング時に引き続きデータを使用します。また、ユーザーセグメンテーションレシピを使用する場合、ユーザーはユーザーセグメントに表示されることがあります。

[ステータス]

データ削除ジョブには、次のいずれかのステータスがあります。

- 保留中 > IN_PROGRESS > COMPLETED - or- FAILED

データ削除ジョブのステータスを取得するには、 DescribeDataDeletionJob API オペレーションを呼び出し、ジョブの Amazon リソースネーム (ARN) を指定します。ステータスが FAILED の場合、レスポンスには、ジョブが失敗した理由を説明する`failureReason`キーが含まれます。

関連 API

- ListDataDeletionJobs
- DescribeDataDeletionJob

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "dataSource": {
      "dataLocation": "string"
   },
   "jobName": "string",
   "roleArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

レコードを削除するデータセットがあるデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

dataSource

削除するユーザーの userIdsのリストを含む Amazon S3 バケット。

型: DataSource オブジェクト

必須: はい

jobName

データ削除ジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須：はい

roleArn

Amazon S3 データソースから読み取るアクセス許可を持つ IAM ロールの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須：はい

tags

データ削除ジョブに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "dataDeletionJobArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

dataDeletionJobArn

データ削除ジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDataset

サービス: Amazon Personalize

空のデータセットを作成し、指定したデータセットグループに追加します。CreateDatasetImportJobトレーニングデータをデータセットにインポートするために使用します。

データセットには次の 5 つのタイプがあります。

- アイテムインタラクション
- 項目
- [ユーザー]
- Action インタラクション
- アクション

各データセットタイプには、必須のフィールドタイプに関連付けられたスキーマがあります。モデルのトレーニング (ソリューションの作成とも呼ばれます) に必要なのは、`Item interactions` データセットのみです。

データセットは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

データセットのステータスを取得するには、を呼び出しますDescribeDataset。

関連 API

- CreateDatasetGroup
- ListDatasets
- DescribeDataset
- DeleteDataset

リクエストの構文

```
{
```

```
    "datasetGroupArn": "string",
    "datasetType": "string",
    "name": "string",
    "schemaArn": "string",
    "tags": [
       {
          "tagKey": "string",
          "tagValue": "string"
       }
    ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

データセットを追加するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

datasetType

データセットのタイプ。

次の値のいずれか (大文字と小文字は区別されません):

- インタラクション
- 項目
- [ユーザー]
- アクション
- Action_Interactions

型: 文字列

長さの制限：最大長は 256 です。

必須: はい

name

データセットの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

schemaArn

データセットに関連付けるスキーマの ARN。スキーマはデータセットフィールドを定義します。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

必須：はい

tags

データセットに適用するタグのリスト

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "datasetArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetArn

データセットの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateDatasetExportJob

サービス: Amazon Personalize

データセットから Amazon S3 バケットにデータをエクスポートするジョブを作成します。Amazon Personalize がトレーニングデータをエクスポートできるようにするには、Amazon S3 バケットについての PutObject の許可を Amazon Personalize に付与する、サービスにリンクされた IAM ロールを指定する必要があります。詳細については、Amazon Personalize デベロッパーガイドの「データセットのエクスポート」を参照してください。

[ステータス]

データセットのエクスポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

エクスポートジョブのステータスを取得するには、を呼び出しDescribeDatasetExportJob、データセットエクスポートジョブの Amazon リソースネーム (ARN) を指定します。ステータスが ACTIVE と表示されると、データセットのエクスポートが完了します。ステータスが CREATE FAILED と表示されている場合、レスポンスには、ジョブが失敗した理由を記述する failureReason キーが含まれています。

リクエストの構文

```
{
   "datasetArn": "string",
   "ingestionMode": "string",
   "jobName": "string",
   "jobOutput": {
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "roleArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

エクスポートするデータを含むデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

ingestionMode

エクスポートするデータ (データをインポートした方法に基づく)。BULKデータセットのインポートジョブを使用してインポートしたデータのみをエクスポートするか、(コンソール、、PutItems オペレーションを使用して) PUT 段階的にインポートしたデータのみをエクスポートするか PutEvents、 PutUsers ALLまたは両方のタイプをエクスポートするかを選択できます。デフォルト値は PUT です。

型: 文字列

有効な値：`BULK | PUT | ALL`

必須：いいえ

jobName

データセットのエクスポートジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須：はい

jobOutput

ジョブの出力が保存される Amazon S3 バケットへのパス。

型: DatasetExportJobOutput オブジェクト

必須: はい

roleArn

出力 Amazon S3 バケットにデータを追加するための許可を持つ IAM サービスロールの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須：はい

tags

データセットのエクスポートジョブに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "datasetExportJobArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetExportJobArn

データセットのエクスポートジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateDatasetGroup

サービス: Amazon Personalize

空のデータセットグループを作成します。データセットグループは、Amazon Personalize のリソースのコンテナです。データセットグループには、データセットのタイプごとに 1 つずつ、最大 3 つのデータセットを含めることができます。

- アイテムインタラクション
- 項目
- [ユーザー]
- アクション
- Action インタラクション

データセットグループは、ドメインを指定してレコメンダーなどの事前設定されたリソースを使用するドメインデータセットグループ、または、キャンペーンでデプロイするカスタムリソースを使用するカスタムデータセットグループ (ソリューションバージョンを使用したソリューションなど) にすることができます。ドメインデータセットグループで始める場合でも、カスタムユースケースのレシピでトレーニングされ、キャンペーンでデプロイされたソリューションやソリューションバージョンなどのカスタムリソースを追加できます。

データセットグループは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING

データセットグループのステータスを取得するには、を呼び出しますDescribeDatasetGroup。ステータスが CREATE FAILED と表示されている場合、レスポンスには、作成が失敗した理由を記述する `failureReason` キーが含まれています。

> **Note**
>
> データセットをグループに追加する前に、データセットグループの `status` が `ACTIVE` になるまで待つ必要があります。

AWS Key Management Service (KMS) キーを指定してグループ内のデータセットを暗号化できます。KMS キーを指定する場合は、キーにアクセスするための許可を持つ AWS Identity and Access Management (IAM) ロールも含める必要があります。

リクエストでデータセットグループ ARN を必要とする API

- CreateDataset
- CreateEventTracker
- CreateSolution

関連 API

- ListDatasetGroups
- DescribeDatasetGroup
- DeleteDatasetGroup

リクエストの構文

```
{
   "domain": "string",
   "kmsKeyArn": "string",
   "name": "string",
   "roleArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

domain

データセットグループのドメイン。ドメインを指定して、ドメインデータセットグループを作成します。指定するドメインによって、データセットのデフォルトスキーマと、レコメンダーが利

用できるユースケースが決まります。ドメインを指定しない場合は、キャンペーンでデプロイするソリューションバージョンを使用してカスタムデータセットグループを作成します。

型: 文字列

有効な値：ECOMMERCE | VIDEO_ON_DEMAND

必須：いいえ

kmsKeyArn

データセットの暗号化に使用される (KMS) キーの Amazon リソースネーム AWS Key Management Service (ARN)。

型: 文字列

長さの制限: 最大長は 2048 です。

Pattern: arn:aws.*:kms:.*:[0-9]{12}:key/.*

必須: いいえ

name

新しいデータセットグループの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

roleArn

(KMS) キーにアクセスする権限を持つ AWS Identity and Access Management (IAM) ロールの AWS Key Management Service ARN。IAM ロールの指定は、KMS キーも指定する場合にのみ有効です。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+

必須: いいえ

tags

データセットグループに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "datasetGroupArn": "string",
   "domain": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetGroupArn

新しいデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

domain

新しいドメインデータセットグループのドメイン。

型: 文字列

有効な値：`ECOMMERCE | VIDEO_ON_DEMAND`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateDatasetImportJob

サービス: Amazon Personalize

データソース (Amazon S3 バケット) から Amazon Personalize データセットにトレーニングデータをインポートするジョブを作成します。Amazon Personalize がトレーニングデータをインポートできるようにするには、Amazon Personalize がデータのコピーを作成して内部で処理するため、データソースからの読み取り許可を持つ IAM サービスロールを指定する必要があります。Amazon S3 バケットへのアクセスを許可する方法については、「Amazon Personalize に Amazon S3 リソースへのアクセスを許可する」を参照してください。

レコメンダーをすでに作成しているか、キャンペーンでカスタムソリューションバージョンをデプロイしている場合、新しいバルクレコードがレコメンデーションにどのように影響するかは、使用するドメインのユースケースまたはレシピによって異なります。詳細については、「新しいデータがリアルタイムのレコメンデーションに与える影響」を参照してください。

> **Important**
>
> デフォルトでは、データセットのインポートジョブは、一括でインポートしたデータセット内の既存のデータを置き換えます。既存のデータを置き換えずに新しいレコードを追加するには、 CreateDatasetImportJob 操作のインポートモードに INCREMENTAL を指定します。

[ステータス]

データセットのインポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

インポートジョブのステータスを取得するには、データセットインポートジョブの Amazon リソースネーム (ARN) を指定してを呼び出しますDescribeDatasetImportJob。ステータスが ACTIVE と表示されると、データセットのインポートが完了します。ステータスが CREATE FAILED と表示されている場合、レスポンスには、ジョブが失敗した理由を記述する `failureReason` キーが含まれています。

> **Note**
>
> インポートには時間がかかります。ステータスが ACTIVE になるまで待ってから、データセットを使用してモデルをトレーニングしてください。

関連 API

- ListDatasetImportJobs
- DescribeDatasetImportJob

リクエストの構文

```
{
   "datasetArn": "string",
   "dataSource": {
      "dataLocation": "string"
   },
   "importMode": "string",
   "jobName": "string",
   "publishAttributionMetricsToS3": boolean,
   "roleArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

インポートされたデータを受け取るデータセットの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

dataSource

インポートするトレーニングデータを含む Amazon S3 バケット。

型: DataSource オブジェクト

必須: はい

importMode

新しいレコードを既存のデータセットに追加する方法を指定します。デフォルトのインポートモードは FULL です。以前にデータセットに一括レコードをインポートしたことがない場合は、FULL を指定することしかできません。

- データセット内の既存のバルクデータをすべて上書きするように FULL を指定します。個別にインポートしたデータは置き換えられません。
- データセット内の既存のデータに新しいレコードを追加するように INCREMENTAL を指定します。Amazon Personalize は、同じ ID のレコードをすべて新しいレコードに置き換えます。

型: 文字列

有効な値：FULL | INCREMENTAL

必須：いいえ

jobName

データセットのインポートジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

publishAttributionMetricsToS3

メトリクス属性を作成した場合は、このインポートジョブのメトリクスを Amazon S3 に発行するかどうかを指定します。

型: ブール値

必須: いいえ

roleArn

Amazon S3 データソースから読み取るための許可を持つ IAM ロールの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須：はい

tags

データセットのインポートジョブに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "datasetImportJobArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetImportJobArn

データセットのインポートジョブの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK

- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateEventTracker

サービス: Amazon Personalize

PutEventsAPI を使用して指定されたデータセットグループにイベントデータを追加するときに使用するイベントトラッカーを作成します。

> **Note**
>
> データセットグループに関連付けることができるイベントトラッカーは 1 つのみです。既存のイベントトラッカーと同じデータセットグループを使用して `CreateEventTracker` を呼び出すと、エラーが発生します。

イベントトラッカーを作成すると、レスポンスにはトラッキング ID が含まれます。トラッキング ID は、PutEventsオペレーションを使用するときにパラメータとして渡されます。その後、Amazon Personalize は、イベントトラッカーで指定したデータセットグループのアイテムインタラクションデータセットにイベントデータを追加します。

イベントトラッカーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

イベントトラッカーのステータスを取得するには、を呼び出しますDescribeEventTracker。

> **Note**
>
> 追跡 ID を使用する前に、イベントトラッカーが ACTIVE 状態になっている必要があります。

関連 API

- ListEventTrackers
- DescribeEventTracker
- DeleteEventTracker

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "name": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

イベントデータを受け取るデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

name

イベントトラッカーの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須：はい

tags

イベントトラッカーに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "eventTrackerArn": "string",
   "trackingId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

eventTrackerArn

イベントトラッカーの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

trackingId

イベントトラッカーの ID。この ID を PutEventsAPI へのリクエストに含めてください。

型: 文字列

長さの制限：最大長は 256 です。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK

- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateFilter

サービス: Amazon Personalize

レコメンデーションフィルターを作成します。詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "filterExpression": "string",
   "name": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

フィルターが属するデータセットグループの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

filterExpression

フィルター式は、レコメンデーションに含める、またはレコメンデーションから除外するアイテムを定義します。フィルター式は、特定のフォーマットルールに従う必要があります。フィルター式の構造と構文については、「フィルター式」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 2500 です。

必須: はい

name

作成するフィルターの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

tags

フィルターに適用するタグのリスト

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "filterArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

filterArn

新しいフィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK

- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateMetricAttribution

サービス: Amazon Personalize

メトリクス属性を作成します。メトリクス属性は、Amazon Personalize にインポートしたデータに関するレポートを作成します。データをインポートした方法に応じて、Amazon CloudWatch または Amazon S3 でレポートを表示できます。詳細については、「レコメンデーションの影響の測定」を参照してください。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "metrics": [
      {
         "eventType": "string",
         "expression": "string",
         "metricName": "string"
      }
   ],
   "metricsOutputConfig": {
      "roleArn": "string",
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "name": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

メトリクス属性に対する宛先ドメインデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

metrics

メトリクス属性のメトリクス属性のリスト。各メトリクス属性は、追跡するイベントタイプと関数を指定します。使用できる関数は、SUM( ) または SAMPLECOUNT( ) です。SUM() 関数では、データセットのタイプ (インタラクションまたはアイテム) と合計する列をパラメーターとして指定します。例えば、SUM(Items.PRICE) などです。

型: MetricAttribute オブジェクトの配列

配列メンバー: 最大数は 10 項目です。

必須: はい

metricsOutputConfig

メトリクス属性の出力設定の詳細。

型: MetricAttributionOutput オブジェクト

必須: はい

name

メトリクス属性の名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

レスポンスの構文

```
{
   "metricAttributionArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

metricAttributionArn

新しいメトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateRecommender

サービス: Amazon Personalize

指定したレシピ (ドメインデータセットグループのユースケース) を使用してレコメンダーを作成します。ドメインデータセットグループのレコメンダーを作成し、リクエストを行うときにレコメンダーの Amazon リソースネーム (ARN) を指定します。GetRecommendations

1 秒あたりの最小レコメンデーションリクエスト数

> **Important**
>
> `minRecommendationRequestsPerSecond` を高く設定すると請求額が増加します。最初は `minRecommendationRequestsPerSecond` に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、`minRecommendationRequestsPerSecond`必要に応じて増やします。

レコメンダーを作成する際、レコメンダーの 1 秒あたりの最小レコメンデーションリクエスト数を設定できます。1 秒あたりの最小レコメンデーションリクエスト数 (`minRecommendationRequestsPerSecond`) は、Amazon Personalize によってプロビジョニングされるベースラインレコメンデーションリクエストスループットを指定します。minRecommendationRequestsPerSecond デフォルトはです1。レコメンデーションリクエストは 1 回の `GetRecommendations` 操作です。リクエストスループットは 1 秒あたりのリクエスト数で測定されます。Amazon Personalize は 1 秒あたりのリクエスト数を使用して、1 時間あたりのリクエスト数とレコメンダーの使用量を算出します。

1 秒あたりのリクエスト数が `minRecommendationRequestsPerSecond` を超えて増加した場合、Amazon Personalize はプロビジョンド容量を自動的にスケールアップまたはスケールダウンしますが、`minRecommendationRequestsPerSecond` を下回ることはありません。容量が引き上げられている間に短時間の遅延が生じます。これにより、リクエストの損失が生じる可能性があります。

請求額は、1 時間あたりの最小リクエスト数 (基準 minRecommendationRequestsPerSecond) または実際のリクエスト数のどちらか大きい方です。実際に使用されるリクエストのスループットは、5 分間のウィンドウ内の平均リクエスト数/秒として計算されます。デフォルトから始めminRecommendationRequestsPerSecond、Amazon CloudWatch メトリックスを使用して使用状況を追跡し、`minRecommendationRequestsPerSecond`必要に応じて値を増やすことをお勧めします。

[ステータス]

レコメンダーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

レコメンダーステータスを取得するには、お電話くださいDescribeRecommender。

> **Note**
>
> レコメンダーの `status` が `ACTIVE` になるまで待ってから、レコメンダーにレコメンデーションを尋ねます。

関連 API

- ListRecommenders
- DescribeRecommender
- UpdateRecommender
- DeleteRecommender

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "name": "string",
   "recipeArn": "string",
   "recommenderConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "minRecommendationRequestsPerSecond": number,
      "trainingDataConfig": {
         "excludedDatasetColumns": {
            "string" : [ "string" ]
```

```
        }
      }
   },
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

レコメンダーの宛先ドメインデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

name

レコメンダーの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須：はい

recipeArn

レコメンダーが使用するレシピの Amazon リソースネーム (ARN)。レコメンダーについて、レシピはドメインデータセットグループのユースケースです。レコメンダーを作成する際、ドメインデータセットグループのユースケースのみを使用できます。ユースケースの詳細については、「レコメンダーのユースケースの選択」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

recommenderConfig

レコメンダーの設定の詳細。

タイプ：RecommenderConfig オブジェクト

必須: いいえ

tags

レコメンダーに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "recommenderArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

recommenderArn

レコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateSchema

サービス: Amazon Personalize

指定されたスキーマ文字列から Amazon Personalize スキーマを作成します。作成するスキーマは Avro JSON 形式である必要があります。

Amazon Personalize は、3 つのスキーマバリアントを認識します。各スキーマはデータセットタイプに関連付けられており、必須フィールドとキーワードのセットを備えています。ドメインデータセットグループ内のデータセットのスキーマを作成する場合は、ドメインデータセットグループのドメインを指定します。呼び出すときにスキーマを指定しますCreateDataset。

スキーマの詳細については、「データセットとスキーマ」を参照してください。

関連 API

- ListSchemas
- DescribeSchema
- DeleteSchema

リクエストの構文

```
{
   "domain": "string",
   "name": "string",
   "schema": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

domain

スキーマのドメイン。ドメインデータセットグループ内のデータセットのスキーマを作成する場合は、ドメインデータセットグループを作成したときに選択したドメインを指定します。

型: 文字列

有効な値：ECOMMERCE | VIDEO_ON_DEMAND

必須：いいえ

name

スキーマの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

schema

Avro JSON 形式のスキーマ。

型: 文字列

長さの制限: 最大長は 20,000 です。

必須: はい

レスポンスの構文

```
{
   "schemaArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

schemaArn

作成したスキーマの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateSolution

サービス: Amazon Personalize

> **Important**
>
> ソリューションを作成した後は、その構成を変更することはできません。デフォルトでは、新しいソリューションはすべて自動トレーニングを使用します。自動トレーニングでは、ソリューションがアクティブである間はトレーニングコストが発生します。ソリューションの自動トレーニングを停止することはできません。不要なコストが発生しないように、終了したら必ずソリューションを削除してください。トレーニング費用の詳細については、「Amazon Personalize 料金表」を参照してください。

モデルをトレーニングするための設定を作成します (ソリューションバージョンを作成します)。この構成には、モデルトレーニングに使用するレシピと、トレーニングに使用する列や特徴変換パラメーターなどのオプションのトレーニング構成が含まれます。ソリューションの設定について詳しくは、「ソリューションの作成と設定」を参照してください。

デフォルトでは、新しいソリューションは自動トレーニングを使用して 7 日ごとにソリューションバージョンを作成します。トレーニングの頻度は変更できます。ソリューションバージョンの自動作成は、ソリューションがアクティブになってから 1 時間後に開始されます。1 時間以内にソリューションバージョンを手動で作成すると、ソリューションは最初の自動トレーニングをスキップします。詳細については、「自動トレーニングの設定」を参照してください。

`performAutoTraining`自動トレーニングをオフにするには、false に設定します。自動トレーニングをオフにする場合は、CreateSolutionVersionオペレーションを呼び出してソリューションバージョンを手動で作成する必要があります。

トレーニングが開始されると、ListSolutionVersionsAPI オペレーションでソリューションバージョンの Amazon リソースネーム (ARN) を取得できます。ステータスを取得するには、を使用してください。DescribeSolutionVersion

トレーニングが完了したら、GetSolutionMetricsを呼び出してモデルの精度を評価できます。満足のいくソリューションバージョンが得られたら、CreateCampaignを使用してデプロイします。キャンペーンは GetRecommendationsAPI を通じてクライアントにレコメンデーションを提供します。

> **Note**
>
> 現時点では、Amazon Personalize は、ソリューションのハイパーパラメータ最適化のため ん、hpoObjective の設定をサポートしていません。

[ステータス]

ソリューションは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

ソリューションのステータスを確認するには、DescribeSolutionに電話してください。マニュアルトレーニングを利用する場合は、電話をかける前にステータスが ACTIVE になっている必要がありますCreateSolutionVersion。

関連 API

- ListSolutions
- CreateSolutionVersion
- DescribeSolution
- DeleteSolution

- ListSolutionVersions
- DescribeSolutionVersion

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "eventType": "string",
   "name": "string",
   "performAutoML": boolean,
   "performAutoTraining": boolean,
   "performHPO": boolean,
   "recipeArn": "string",
   "solutionConfig": {
```

```
   "algorithmHyperParameters": {
      "string" : "string"
   },
   "autoMLConfig": {
      "metricName": "string",
      "recipeList": [ "string" ]
   },
   "autoTrainingConfig": {
      "schedulingExpression": "string"
   },
   "eventValueThreshold": "string",
   "featureTransformationParameters": {
      "string" : "string"
   },
   "hpoConfig": {
      "algorithmHyperParameterRanges": {
         "categoricalHyperParameterRanges": [
            {
               "name": "string",
               "values": [ "string" ]
            }
         ],
         "continuousHyperParameterRanges": [
            {
               "maxValue": number,
               "minValue": number,
               "name": "string"
            }
         ],
         "integerHyperParameterRanges": [
            {
               "maxValue": number,
               "minValue": number,
               "name": "string"
            }
         ]
      },
      "hpoObjective": {
         "metricName": "string",
         "metricRegex": "string",
         "type": "string"
      },
      "hpoResourceConfig": {
         "maxNumberOfTrainingJobs": "string",
```

```
            "maxParallelTrainingJobs": "string"
         }
      },
      "optimizationObjective": {
         "itemAttribute": "string",
         "objectiveSensitivity": "string"
      },
      "trainingDataConfig": {
         "excludedDatasetColumns": {
            "string" : [ "string" ]
         }
      }
   },
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

トレーニングデータを提供するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

eventType

(EVENT_TYPE スキーマフィールドを使用して) 複数のイベントタイプがある場合、このパラメータは、モデルのトレーニングに使用されるイベントタイプ (例えば、「クリック」または「いいね」) を指定します。

`eventType` を指定しない場合、Amazon Personalize は、タイプにかかわらず、同じ重みでトレーニングするためにすべてのインタラクションを使用します。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

name

ソリューションの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須：はい

performAutoML

> **Important**
>
> 自動機械学習の有効化は推奨されません。代わりに、利用可能な Amazon Personalize のレシピにユースケースをマッチさせます。詳細については、の「Choosing a recipe」を参照してください。

自動機械学習 (AutoML) を実行するかどうか。デフォルトは `false` です。この場合、`recipeArn` を指定する必要があります。

`true` に設定すると、Amazon Personalize はトレーニングデータを分析し、最適な USER_PERSONALIZATION レシピとハイパーパラメータを選択します。この場合、`recipeArn` を省略する必要があります。Amazon Personalize は、ハイパーパラメータにさまざまな値を使用してテストを実行することにより、最適なレシピを決定します。AutoML では、特定のレシピを選択する場合と比較して、トレーニングプロセスに長時間を要します。

型: ブール値

必須: いいえ

performAutoTraining

ソリューションが自動トレーニングを使用して新しいソリューションバージョン (トレーニング済みモデル) を作成するかどうか。`True`デフォルトはで、ソリューションは 7 日ごとに新しいソ

リューションバージョンを自動的に作成します。AutoTrainingConfigソリューション構成の一部として a schedulingExpression を指定することで、トレーニングの頻度を変更できます。自動学習について詳しくは、「自動学習の設定」を参照してください。

ソリューションバージョンの自動作成は、ソリューションがアクティブになってから 1 時間後に開始されます。1 時間以内にソリューションバージョンを手動で作成すると、ソリューションは最初の自動トレーニングをスキップします。

トレーニングが開始されると、ListSolutionVersionsAPI オペレーションでソリューションバージョンの Amazon リソースネーム (ARN) を取得できます。ステータスを取得するには、を使用してください。DescribeSolutionVersion

型: ブール値

必須: いいえ

performHPO

指定または選択したレシピでハイパーパラメータ最適化 (HPO) を実行するかどうか。デフォルトは false です。

AutoML を実行する場合、このパラメータは常に true となります。false に設定しないでください。

型: ブール値

必須: いいえ

recipeArn

モデルトレーニングに使用するレシピの Amazon リソースネーム (ARN)。これは、performAutoML が false の場合に必要です。Amazon Personalize のさまざまなレシピとその ARN の詳細については、「Choosing a recipe」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

solutionConfig

ソリューションで使用する設定。performAutoML が true に設定されている場合、Amazon Personalize はソリューション設定の autoMLConfig セクションのみを評価します。

> **Note**
>
> 現時点では、Amazon Personalize は hpoObjective の設定をサポートしていません。

タイプ : SolutionConfig オブジェクト

必須: いいえ

tags

ソリューションに適用するタグのリスト。

タイプ : Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

レスポンスの構文

```
{
   "solutionArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

solutionArn

ソリューションの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

CreateSolutionVersion

サービス: Amazon Personalize

カスタムデータセットグループでアクティブなソリューションをトレーニングまたは再トレーニングします。CreateSolutionソリューションは操作を使用して作成され、呼び出し前は ACTIVE 状態である必要がありますCreateSolutionVersion。この操作を呼び出すたびに、ソリューションの新しいバージョンが作成されます。

[ステータス]

ソリューションバージョンは、次のいずれかの状態になります。

- CREATE PENDING
- CREATE IN_PROGRESS
- ACTIVE
- CREATE FAILED
- CREATE STOPPING
- CREATE STOPPED

バージョンのステータスを取得するには、を呼び出しますDescribeSolutionVersion。ステータスが ACTIVE と表示されるまで待ってから、CreateCampaign を呼び出します。

ステータスが CREATE FAILED と表示されている場合、レスポンスには、ジョブが失敗した理由を記述する failureReason キーが含まれています。

関連 API

- ListSolutionVersions
- DescribeSolutionVersion
- ListSolutions
- CreateSolution
- DescribeSolution
- DeleteSolution

リクエストの構文

```
{
```

```
   "name": "string",
   "solutionArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ],
   "trainingMode": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

name

ソリューションバージョンの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

solutionArn

トレーニング設定情報を含むソリューションの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

必須：はい

tags

ソリューションバージョンに適用するタグのリスト。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: いいえ

trainingMode

ソリューションバージョンを作成するときに実行するトレーニングの範囲。デフォルトは FULL です。これにより、データセットグループ内のデータセットからのトレーニングデータ全体に基づいて、完全に新しいモデルが作成されます。

User-Personalizationを使用する場合は、UPDATE のトレーニングモードを指定できます。これによりモデルが更新され、新しい項目をレコメンデーションの対象として検討するようになります。これは完全な再トレーニングではありません。それでも、毎週完全な再トレーニングを完了する必要があります。UPDATE を指定した場合、Amazon Personalize はソリューションバージョンの自動更新を停止します。更新を再開するには、トレーニングモードを FULL に設定して新しいソリューションを作成し、キャンペーンにデプロイします。自動更新の詳細については、「自動更新」を参照してください。

UPDATE オプションは、FULL オプションを使用して入力ソリューションから作成されたアクティブなソリューションバージョンが既にあり、入力ソリューションが User-Personalization レシピまたは HRNN-Coldstart レシピでトレーニングされている場合にのみ使用できます。

型: 文字列

有効な値：FULL | UPDATE | AUTOTRAIN

必須：いいえ

レスポンスの構文

```
{
   "solutionVersionArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

solutionVersionArn

新しいソリューションバージョンの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteCampaign

サービス: Amazon Personalize

ソリューションのデプロイを削除して、キャンペーンを削除します。キャンペーンのベースとなっているソリューションは削除されず、必要に応じて再デプロイできます。GetRecommendations削除されたキャンペーンはリクエストで指定できなくなりました。キャンペーンの作成方法については、を参照してくださいCreateCampaign。

リクエストの構文

```
{
   "campaignArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignArn

削除するキャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteDataset

サービス: Amazon Personalize

データセットを削除します。関連付けられた DatasetImportJob または SolutionVersion が CREATE PENDING または IN PROGRESS 状態にある場合、データセットを削除することはできません。データセットの詳細については、を参照してくださいCreateDataset。

リクエストの構文

```
{
   "datasetArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

削除するデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteDatasetGroup

サービス: Amazon Personalize

データセットグループを削除します。データセットグループを削除する前に、以下を削除する必要があります。

- すべての関連付けられたイベントトラッカー。
- すべての関連付けられたソリューション。
- データセットグループのすべてのデータセット。

リクエストの構文

```
{
   "datasetGroupArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

削除するデータセットグループの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteEventTracker

サービス: Amazon Personalize

イベントトラッカーを削除します。データセットグループからデータセットを削除しません。イベントトラッカーの詳細については、を参照してくださいCreateEventTracker。

リクエストの構文

```
{
   "eventTrackerArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

eventTrackerArn

削除するイベントトラッカーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteFilter

サービス: Amazon Personalize

フィルターを削除します。

リクエストの構文

```
{
   "filterArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

filterArn

削除するフィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteMetricAttribution

サービス: Amazon Personalize

メトリクス属性を削除します。

リクエストの構文

```
{
   "metricAttributionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

metricAttributionArn

メトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteRecommender

サービス: Amazon Personalize

レコメンダーを非アクティブにして削除します。GetRecommendations削除されたレコメンダーはリクエストで指定できなくなりました。

リクエストの構文

```
{
   "recommenderArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

recommenderArn

削除するレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteSchema

サービス: Amazon Personalize

スキーマを削除します。スキーマを削除する前に、そのスキーマを参照しているすべてのデータセットを削除する必要があります。スキーマの詳細については、を参照してくださいCreateSchema。

リクエストの構文

```
{
   "schemaArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

schemaArn

削除するスキーマの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DeleteSolution

サービス: Amazon Personalize

ソリューションのすべてのバージョンと Solution オブジェクト自体を削除します。ソリューションを削除する前に、そのソリューションに基づくすべてのキャンペーンを削除する必要があります。どのキャンペーンがソリューションを使用しているかを判断するには、ソリューションの Amazon リソースネーム (ARN) ListCampaignsに電話して提供してください。関連付けられた SolutionVersion が CREATE PENDING または IN PROGRESS 状態にある場合、ソリューションを削除することはできません。ソリューションの詳細については、を参照してくださいCreateSolution。

リクエストの構文

```
{
   "solutionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

solutionArn

削除するソリューションの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeAlgorithm

サービス: Amazon Personalize

特定のアルゴリズムを記述します。

リクエストの構文

```
{
   "algorithmArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

algorithmArn

記述するアルゴリズムの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "algorithm": {
      "algorithmArn": "string",
      "algorithmImage": {
         "dockerURI": "string",
         "name": "string"
      },
      "creationDateTime": number,
      "defaultHyperParameterRanges": {
         "categoricalHyperParameterRanges": [
            {
               "isTunable": boolean,
               "name": "string",
```

```
                "values": [ "string" ]
            }
        ],
        "continuousHyperParameterRanges": [
            {
                "isTunable": boolean,
                "maxValue": number,
                "minValue": number,
                "name": "string"
            }
        ],
        "integerHyperParameterRanges": [
            {
                "isTunable": boolean,
                "maxValue": number,
                "minValue": number,
                "name": "string"
            }
        ]
    },
    "defaultHyperParameters": {
        "string" : "string"
    },
    "defaultResourceConfig": {
        "string" : "string"
    },
    "lastUpdatedDateTime": number,
    "name": "string",
    "roleArn": "string",
    "trainingInputMode": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

algorithm

アルゴリズムのプロパティのリスト。

型: Algorithm オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeBatchInferenceJob

サービス: Amazon Personalize

名前、Amazon リソースネーム (ARN)、ステータス、入力と出力の設定、およびレコメンデーションの生成に使用されたソリューションバージョンの ARN を含むバッチ推論ジョブのプロパティを取得します。

リクエストの構文

```
{
   "batchInferenceJobArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

batchInferenceJobArn

記述するバッチ推論ジョブの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "batchInferenceJob": {
      "batchInferenceJobArn": "string",
      "batchInferenceJobConfig": {
         "itemExplorationConfig": {
            "string" : "string"
         }
      },
      "batchInferenceJobMode": "string",
      "creationDateTime": number,
      "failureReason": "string",
      "filterArn": "string",
```

```
      "jobInput": {
         "s3DataSource": {
            "kmsKeyArn": "string",
            "path": "string"
         }
      },
      "jobName": "string",
      "jobOutput": {
         "s3DataDestination": {
            "kmsKeyArn": "string",
            "path": "string"
         }
      },
      "lastUpdatedDateTime": number,
      "numResults": number,
      "roleArn": "string",
      "solutionVersionArn": "string",
      "status": "string",
      "themeGenerationConfig": {
         "fieldsForThemeGeneration": {
            "itemName": "string"
         }
      }
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

batchInferenceJob

指定されたバッチ推論ジョブに関する情報。

型: BatchInferenceJob オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeBatchSegmentJob

サービス: Amazon Personalize

名前、Amazon リソースネーム (ARN)、ステータス、入力と出力の設定、およびセグメントの生成に使用されたソリューションバージョンの ARN を含むバッチセグメントジョブのプロパティを取得します。

リクエストの構文

```
{
   "batchSegmentJobArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

batchSegmentJobArn

記述するバッチセグメントジョブの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "batchSegmentJob": {
      "batchSegmentJobArn": "string",
      "creationDateTime": number,
      "failureReason": "string",
      "filterArn": "string",
      "jobInput": {
         "s3DataSource": {
            "kmsKeyArn": "string",
            "path": "string"
         }
      },
```

```
      "jobName": "string",
      "jobOutput": {
         "s3DataDestination": {
            "kmsKeyArn": "string",
            "path": "string"
         }
      },
      "lastUpdatedDateTime": number,
      "numResults": number,
      "roleArn": "string",
      "solutionVersionArn": "string",
      "status": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

batchSegmentJob

指定されたバッチセグメントジョブに関する情報。

型: BatchSegmentJob オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeCampaign

サービス: Amazon Personalize

ステータスを含む、特定のキャンペーンを記述します。

キャンペーンは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

`status` が `CREATE FAILED` の場合、レスポンスには理由を記述する `failureReason` キーが含まれます。

キャンペーンの詳細については、を参照してくださいCreateCampaign。

リクエストの構文

```
{
   "campaignArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignArn

キャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "campaign": {
```

```
      "campaignArn": "string",
      "campaignConfig": {
         "enableMetadataWithRecommendations": boolean,
         "itemExplorationConfig": {
            "string" : "string"
         },
         "syncWithLatestSolutionVersion": boolean
      },
      "creationDateTime": number,
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "latestCampaignUpdate": {
         "campaignConfig": {
            "enableMetadataWithRecommendations": boolean,
            "itemExplorationConfig": {
               "string" : "string"
            },
            "syncWithLatestSolutionVersion": boolean
         },
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "minProvisionedTPS": number,
         "solutionVersionArn": "string",
         "status": "string"
      },
      "minProvisionedTPS": number,
      "name": "string",
      "solutionVersionArn": "string",
      "status": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

campaign

キャンペーンのプロパティ。

型: Campaign オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeDataDeletionJob

サービス: Amazon Personalize

によって作成されたデータ削除ジョブをCreateDataDeletionJob、ジョブのステータスを含めて記述します。

リクエストの構文

```
{
   "dataDeletionJobArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

dataDeletionJobArn

データ削除ジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "dataDeletionJob": {
      "creationDateTime": number,
      "dataDeletionJobArn": "string",
      "datasetGroupArn": "string",
      "dataSource": {
         "dataLocation": "string"
      },
      "failureReason": "string",
      "jobName": "string",
      "lastUpdatedDateTime": number,
      "numDeleted": number,
```

```
        "roleArn": "string",
        "status": "string"
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

dataDeletionJob

ステータスを含む、データ削除ジョブに関する情報。

ステータスは、次のいずれかの値です。

- 保留中
- IN_PROGRESS
- COMPLETED
- FAILED

型: DataDeletionJob オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDataset

サービス: Amazon Personalize

特定のデータセットの記述を表示します。データセットの詳細については、を参照してくださいCreateDataset。

リクエストの構文

```
{
   "datasetArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

記述するデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "dataset": {
      "creationDateTime": number,
      "datasetArn": "string",
      "datasetGroupArn": "string",
      "datasetType": "string",
      "lastUpdatedDateTime": number,
      "latestDatasetUpdate": {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "schemaArn": "string",
```

```
            "status": "string"
        },
        "name": "string",
        "schemaArn": "string",
        "status": "string",
        "trackingId": "string"
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

dataset

データセットのプロパティのリスト。

型: Dataset オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET

- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeDatasetExportJob

サービス: Amazon Personalize

によって作成されたデータセットのエクスポートジョブについてCreateDatasetExportJob、エクスポートジョブのステータスを含めて説明します。

リクエストの構文

```
{
   "datasetExportJobArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetExportJobArn

記述するデータセットのエクスポートジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "datasetExportJob": {
      "creationDateTime": number,
      "datasetArn": "string",
      "datasetExportJobArn": "string",
      "failureReason": "string",
      "ingestionMode": "string",
      "jobName": "string",
      "jobOutput": {
         "s3DataDestination": {
            "kmsKeyArn": "string",
```

```
            "path": "string"
         }
      },
      "lastUpdatedDateTime": number,
      "roleArn": "string",
      "status": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetExportJob

ステータスを含む、データセットのエクスポートジョブに関する情報。

ステータスは、次のいずれかの値です。

- CREATE PENDING
- CREATE IN_PROGRESS
- ACTIVE
- CREATE FAILED

型: DatasetExportJob オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeDatasetGroup

サービス: Amazon Personalize

特定のデータセットグループを記述します。データセットグループの詳細については、を参照してください CreateDatasetGroup。

リクエストの構文

```
{
   "datasetGroupArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

記述するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "datasetGroup": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "domain": "string",
      "failureReason": "string",
      "kmsKeyArn": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "roleArn": "string",
      "status": "string"
   }
```

```
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetGroup

データセットグループのプロパティのリスト。

型: DatasetGroup オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK

- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeDatasetImportJob

サービス: Amazon Personalize

によって作成されたデータセットインポートジョブについてCreateDatasetImportJob、インポートジョブのステータスを含めて説明します。

リクエストの構文

```
{
   "datasetImportJobArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetImportJobArn

記述するデータセットのインポートジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "datasetImportJob": {
      "creationDateTime": number,
      "datasetArn": "string",
      "datasetImportJobArn": "string",
      "dataSource": {
         "dataLocation": "string"
      },
      "failureReason": "string",
      "importMode": "string",
      "jobName": "string",
      "lastUpdatedDateTime": number,
```

```
        "publishAttributionMetricsToS3": boolean,
        "roleArn": "string",
        "status": "string"
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetImportJob

ステータスを含む、データセットのインポートジョブに関する情報。

ステータスは、次のいずれかの値です。

- CREATE PENDING
- CREATE IN_PROGRESS
- ACTIVE
- CREATE FAILED

型: DatasetImportJob オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeEventTracker

サービス: Amazon Personalize

イベントトラッカーの説明を記述します。レスポンスには、イベントトラッカーの `trackingId` と `status` が含まれます。イベントトラッカーの詳細については、を参照してくださいCreateEventTracker。

リクエストの構文

```
{
   "eventTrackerArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

eventTrackerArn

記述するイベントトラッカーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "eventTracker": {
      "accountId": "string",
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "eventTrackerArn": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "status": "string",
      "trackingId": "string"
   }
```

```
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

eventTracker

イベントトラッカーを記述するオブジェクトです。

型: EventTracker オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK

- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeFeatureTransformation

サービス: Amazon Personalize

特定の特徴変換を記述します。

リクエストの構文

```
{
   "featureTransformationArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

featureTransformationArn

記述する特徴変換の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "featureTransformation": {
      "creationDateTime": number,
      "defaultParameters": {
         "string" : "string"
      },
      "featureTransformationArn": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "status": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

featureTransformation

FeatureTransformation プロパティのリスト。

型: FeatureTransformation オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK

- AWS ルビー V3 用 SDK

DescribeFilter

サービス: Amazon Personalize

フィルターのプロパティを記述します。

リクエストの構文

```
{
   "filterArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

filterArn

記述するフィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "filter": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "failureReason": "string",
      "filterArn": "string",
      "filterExpression": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "status": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

filter

フィルターの詳細。

型: Filter オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeMetricAttribution

サービス: Amazon Personalize

メトリクス属性について説明します。

リクエストの構文

```
{
   "metricAttributionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

metricAttributionArn

メトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "metricAttribution": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "metricAttributionArn": "string",
      "metricsOutputConfig": {
         "roleArn": "string",
         "s3DataDestination": {
            "kmsKeyArn": "string",
            "path": "string"
         }
```

```
        },
        "name": "string",
        "status": "string"
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

metricAttribution

メトリックス属性の詳細。

型: MetricAttribution オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK

- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeRecipe

サービス: Amazon Personalize

レシピを説明を記述します。

レシピには 3 つのアイテムが含まれています。

- モデルをトレーニングするアルゴリズム。
- トレーニングを統制するハイパーパラメータ。
- トレーニング前に入力データを変更するための特徴変換に関する情報。

Amazon Personalize は、事前定義された一連のレシピを提供します。CreateSolutionAPI を使用してソリューションを作成するときにレシピを指定します。 `CreateSolution`指定されたレシピのアルゴリズムとトレーニングデータセットを使用してモデルをトレーニングします。ソリューションをキャンペーンとしてデプロイすると、GetRecommendationsAPI を使用してレコメンデーションを提供できます。

リクエストの構文

```
{
   "recipeArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

recipeArn

記述するレシピの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "recipe": {
      "algorithmArn": "string",
      "creationDateTime": number,
      "description": "string",
      "featureTransformationArn": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "recipeArn": "string",
      "recipeType": "string",
      "status": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

recipe

レシピを記述するオブジェクト。

型: Recipe オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeRecommender

サービス: Amazon Personalize

ステータスを含む、特定のレコメンダーを記述します。

レコメンダーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

`status` が `CREATE FAILED` の場合、レスポンスには理由を記述する `failureReason` キーが含まれます。

レコメンダーを作成または削除しているときは、`modelMetrics` キーは null になります。

レコメンダーの詳細については、を参照してくださいCreateRecommender。

リクエストの構文

```
{
   "recommenderArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

recommenderArn

記述するレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "recommender": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "latestRecommenderUpdate": {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "recommenderConfig": {
            "enableMetadataWithRecommendations": boolean,
            "itemExplorationConfig": {
               "string" : "string"
            },
            "minRecommendationRequestsPerSecond": number,
            "trainingDataConfig": {
               "excludedDatasetColumns": {
                  "string" : [ "string" ]
               }
            }
         },
         "status": "string"
      },
      "modelMetrics": {
         "string" : number
      },
      "name": "string",
      "recipeArn": "string",
      "recommenderArn": "string",
      "recommenderConfig": {
         "enableMetadataWithRecommendations": boolean,
         "itemExplorationConfig": {
            "string" : "string"
         },
         "minRecommendationRequestsPerSecond": number,
         "trainingDataConfig": {
            "excludedDatasetColumns": {
               "string" : [ "string" ]
            }
         }
      },
```

```
        "status": "string"
    }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

recommender

レコメンダーのプロパティ。

型: Recommender オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK

- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeSchema

サービス: Amazon Personalize

スキーマの説明を記述します。スキーマの詳細については、を参照してくださいCreateSchema。

リクエストの構文

```
{
   "schemaArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

schemaArn

取得するスキーマの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "schema": {
      "creationDateTime": number,
      "domain": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "schema": "string",
      "schemaArn": "string"
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

schema

リクエストされたスキーマ。

型: DatasetSchema オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeSolution

サービス: Amazon Personalize

ソリューションの説明を記述します。ソリューションの詳細については、を参照してくださいCreateSolution。

リクエストの構文

```
{
   "solutionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

solutionArn

記述するソリューションの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "solution": {
      "autoMLResult": {
         "bestRecipeArn": "string"
      },
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "eventType": "string",
      "lastUpdatedDateTime": number,
      "latestSolutionVersion": {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
```

```
      "solutionVersionArn": "string",
      "status": "string",
      "trainingMode": "string",
      "trainingType": "string"
   },
   "name": "string",
   "performAutoML": boolean,
   "performAutoTraining": boolean,
   "performHPO": boolean,
   "recipeArn": "string",
   "solutionArn": "string",
   "solutionConfig": {
      "algorithmHyperParameters": {
         "string" : "string"
      },
      "autoMLConfig": {
         "metricName": "string",
         "recipeList": [ "string" ]
      },
      "autoTrainingConfig": {
         "schedulingExpression": "string"
      },
      "eventValueThreshold": "string",
      "featureTransformationParameters": {
         "string" : "string"
      },
      "hpoConfig": {
         "algorithmHyperParameterRanges": {
            "categoricalHyperParameterRanges": [
               {
                  "name": "string",
                  "values": [ "string" ]
               }
            ],
            "continuousHyperParameterRanges": [
               {
                  "maxValue": number,
                  "minValue": number,
                  "name": "string"
               }
            ],
            "integerHyperParameterRanges": [
               {
                  "maxValue": number,
```

```
                    "minValue": number,
                    "name": "string"
                 }
              ]
           },
           "hpoObjective": {
              "metricName": "string",
              "metricRegex": "string",
              "type": "string"
           },
           "hpoResourceConfig": {
              "maxNumberOfTrainingJobs": "string",
              "maxParallelTrainingJobs": "string"
           }
        },
        "optimizationObjective": {
           "itemAttribute": "string",
           "objectiveSensitivity": "string"
        },
        "trainingDataConfig": {
           "excludedDatasetColumns": {
              "string" : [ "string" ]
           }
        }
     },
     "status": "string"
  }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

solution

ソリューションを記述するオブジェクト。

型: Solution オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

DescribeSolutionVersion

サービス: Amazon Personalize

ソリューションの特定のバージョンを記述します。ソリューションの詳細については、を参照してください。CreateSolution

リクエストの構文

```
{
   "solutionVersionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

solutionVersionArn

ソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "solutionVersion": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "eventType": "string",
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "performAutoML": boolean,
      "performHPO": boolean,
      "recipeArn": "string",
      "solutionArn": "string",
      "solutionConfig": {
```

```
        "algorithmHyperParameters": {
           "string" : "string"
        },
        "autoMLConfig": {
           "metricName": "string",
           "recipeList": [ "string" ]
        },
        "autoTrainingConfig": {
           "schedulingExpression": "string"
        },
        "eventValueThreshold": "string",
        "featureTransformationParameters": {
           "string" : "string"
        },
        "hpoConfig": {
           "algorithmHyperParameterRanges": {
              "categoricalHyperParameterRanges": [
                 {
                    "name": "string",
                    "values": [ "string" ]
                 }
              ],
              "continuousHyperParameterRanges": [
                 {
                    "maxValue": number,
                    "minValue": number,
                    "name": "string"
                 }
              ],
              "integerHyperParameterRanges": [
                 {
                    "maxValue": number,
                    "minValue": number,
                    "name": "string"
                 }
              ]
           },
           "hpoObjective": {
              "metricName": "string",
              "metricRegex": "string",
              "type": "string"
           },
           "hpoResourceConfig": {
              "maxNumberOfTrainingJobs": "string",
```

```
                "maxParallelTrainingJobs": "string"
            }
         },
         "optimizationObjective": {
            "itemAttribute": "string",
            "objectiveSensitivity": "string"
         },
         "trainingDataConfig": {
            "excludedDatasetColumns": {
               "string" : [ "string" ]
            }
         }
      },
      "solutionVersionArn": "string",
      "status": "string",
      "trainingHours": number,
      "trainingMode": "string",
      "trainingType": "string",
      "tunedHPOParams": {
         "algorithmHyperParameters": {
            "string" : "string"
         }
      }
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

solutionVersion

ソリューションバージョン。

型: SolutionVersion オブジェクト

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

GetSolutionMetrics

サービス: Amazon Personalize

指定されたソリューションバージョンのメトリクスを取得します。

リクエストの構文

```
{
   "solutionVersionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

solutionVersionArn

メトリクスを取得するソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "metrics": {
      "string" : number
   },
   "solutionVersionArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

metrics

ソリューションバージョンのメトリクス。詳細については、[メトリクスを使用してソリューションバージョンを評価する]を参照してください。

タイプ: ダブルマップへの文字列。

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

solutionVersionArn

リクエストで指定されたものと同じソリューションバージョン ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListBatchInferenceJobs

サービス: Amazon Personalize

ソリューションバージョンから実行されたバッチ推論ジョブのリストを取得します。

リクエストの構文

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionVersionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

各ページで返されるバッチ推論ジョブの結果の最大数。デフォルト値は 100 です。

型: 整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

次の結果ページを要求するためのトークン。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

solutionVersionArn

バッチ推論ジョブが作成されたソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

レスポンスの構文

```
{
   "batchInferenceJobs": [
      {
         "batchInferenceJobArn": "string",
         "batchInferenceJobMode": "string",
         "creationDateTime": number,
         "failureReason": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "solutionVersionArn": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

batchInferenceJobs

返される各ジョブに関する情報を含むリスト。

型: BatchInferenceJobSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

次の結果ページの取得に使用するトークン。返される結果がそれ以上存在しない場合、値は null になります。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListBatchSegmentJobs

サービス: Amazon Personalize

指定したソリューションバージョンから実行されたバッチセグメントジョブのリストを取得します。

リクエストの構文

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionVersionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

各ページで返されるバッチセグメントジョブの結果の最大数。デフォルト値は 100 です。

型: 整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

次の結果ページを要求するためのトークン。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

solutionVersionArn

バッチセグメントジョブがバッチセグメントを生成するために使用したソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

レスポンスの構文

```
{
   "batchSegmentJobs": [
      {
         "batchSegmentJobArn": "string",
         "creationDateTime": number,
         "failureReason": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "solutionVersionArn": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

batchSegmentJobs

返される各ジョブに関する情報を含むリスト。

型: BatchSegmentJobSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

次の結果ページの取得に使用するトークン。返される結果がそれ以上存在しない場合、値は `null` になります。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListCampaigns

サービス: Amazon Personalize

特定のソリューションを使用するキャンペーンのリストを返します。ソリューションが指定されていない場合、アカウントに関連付けられているすべてのキャンペーンが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) など、各キャンペーンのプロパティを提供します。キャンペーンの詳細については、を参照してくださいCreateCampaign。

リクエストの構文

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

返されるキャンペーンの最大数。

タイプ : 整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

前回の呼び出しで返されたトークンで、次のキャンペーンセット (存在する場合) ListCampaignsを取得するために返されます。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

solutionArn

キャンペーンを一覧表示するソリューションの Amazon リソースネーム (ARN)。ソリューションが指定されていない場合、アカウントに関連付けられているすべてのキャンペーンが一覧表示されます。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

レスポンスの構文

```
{
   "campaigns": [
      {
         "campaignArn": "string",
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

campaigns

キャンペーンのリスト。

型: CampaignSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

キャンペーンの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：\p{ASCII}{0,1500}

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListDataDeletionJobs

サービス: Amazon Personalize

データセットグループのデータ削除ジョブのリストを、作成時刻順に最新のものとともに返します。データセットグループが指定されていない場合、アカウントに関連付けられているすべてのデータ削除ジョブが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) を含む各ジョブのプロパティを提供します。データ削除ジョブの詳細については、「ユーザーの削除」を参照してください。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

データ削除ジョブを一覧表示するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるデータ削除ジョブの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

次のジョブセットを取得ListDataDeletionJobsするために、前の への呼び出しから返されたトークン (存在する場合) 。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "dataDeletionJobs": [
      {
         "creationDateTime": number,
         "dataDeletionJobArn": "string",
         "datasetGroupArn": "string",
         "failureReason": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

dataDeletionJobs

データ削除ジョブのリスト。

型: DataDeletionJobSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

次のデータ削除ジョブのセットを取得するためのトークン (存在する場合) 。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：\p{ASCII}{0,1500}

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListDatasetExportJobs

サービス: Amazon Personalize

特定のデータセットを使用するデータセットのエクスポートジョブのリストを返します。データセットが指定されていない場合、アカウントに関連付けられているすべてのデータセットのエクスポートジョブが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) など、データセットの各エクスポートジョブのプロパティを提供します。データセットのエクスポートジョブの詳細については、を参照してくださいCreateDatasetExportJob。データセットの詳細については、を参照してくださいCreateDataset。

リクエストの構文

```
{
   "datasetArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

データセットのエクスポートジョブを一覧表示するデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるデータセットのエクスポートジョブの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

データセットのエクスポートジョブの次のセットを取得するために、前の ListDatasetExportJobs に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "datasetExportJobs": [
      {
         "creationDateTime": number,
         "datasetExportJobArn": "string",
         "failureReason": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetExportJobs

データセットのエクスポートジョブのリスト。

型: DatasetExportJobSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

データセットのエクスポートジョブの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListDatasetGroups

サービス: Amazon Personalize

データセットグループのリストを返します。レスポンスは、Amazon リソースネーム (ARN) など、各データセットグループのプロパティを提供します。データセットグループの詳細については、を参照してくださいCreateDatasetGroup。

リクエストの構文

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

返されるデータセットグループの最大数。

タイプ : 整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

データセットグループの次のセットを取得するために、前の `ListDatasetGroups` に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

レスポンスの構文

```
{
```

```
   "datasetGroups": [
      {
         "creationDateTime": number,
         "datasetGroupArn": "string",
         "domain": "string",
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetGroups

データセットグループのリスト。

型: DatasetGroupSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

データセットグループの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListDatasetImportJobs

サービス: Amazon Personalize

特定のデータセットを使用するデータセットのインポートジョブのリストを返します。データセットが指定されていない場合、アカウントに関連付けられているすべてのデータセットのインポートジョブが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) など、データセットの各インポートジョブのプロパティを提供します。データセットのインポートジョブの詳細については、を参照してくださいCreateDatasetImportJob。データセットの詳細については、を参照してくださいCreateDataset。

リクエストの構文

```
{
   "datasetArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

データセットのインポートジョブを一覧表示するデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるデータセットのインポートジョブの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

データセットのインポートジョブの次のセットを取得するために、前の ListDatasetImportJobs に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "datasetImportJobs": [
      {
         "creationDateTime": number,
         "datasetImportJobArn": "string",
         "failureReason": "string",
         "importMode": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetImportJobs

データセットのインポートジョブのリスト。

型: DatasetImportJobSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

データセットのインポートジョブの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListDatasets

サービス: Amazon Personalize

特定のデータセットグループに含まれるデータセットのリストを返します。レスポンスは、Amazon リソースネーム (ARN) など、各データセットのプロパティを提供します。データセットの詳細については、を参照してくださいCreateDataset。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

一覧表示するデータセットを含むデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるデータセットの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

データセットのインポートジョブの次のセットを取得するために、前の `ListDatasets` に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

レスポンスの構文

```
{
   "datasets": [
      {
         "creationDateTime": number,
         "datasetArn": "string",
         "datasetType": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasets

Dataset オブジェクトの配列。各オブジェクトはメタデータ情報を提供します。

型: DatasetSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

データセットの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListEventTrackers

サービス: Amazon Personalize

アカウントに関連付けられているイベントトラッカーのリストを返します。レスポンスは、Amazon リソースネーム (ARN) や追跡 ID など、各イベントトラッカーのプロパティを提供します。イベントトラッカーの詳細については、を参照してくださいCreateEventTracker。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

レスポンスのフィルタリングに使用されるデータセットグループの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるイベントトラッカーの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

イベントトラッカーの次のセットを取得するために、前の `ListEventTrackers` に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "eventTrackers": [
      {
         "creationDateTime": number,
         "eventTrackerArn": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

eventTrackers

イベントトラッカーのリスト。

型: EventTrackerSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

イベントトラッカーの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：\p{ASCII}{0,1500}

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListFilters

サービス: Amazon Personalize

特定のデータセットグループに属するすべてのフィルターを一覧表示します。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

フィルターを含むデータセットグループの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるフィルターの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

フィルターの次のセットを取得するために、前の `ListFilters` に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "Filters": [
      {
         "creationDateTime": number,
         "datasetGroupArn": "string",
         "failureReason": "string",
         "filterArn": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

Filters

返されたフィルターのリスト。

型: FilterSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

フィルターの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListMetricAttributionMetrics

サービス: Amazon Personalize

メトリクス属性のメトリクスを一覧表示します。

リクエストの構文

```
{
   "maxResults": number,
   "metricAttributionArn": "string",
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

結果の 1 ページで返されるメトリクスの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

metricAttributionArn

属性を取得するメトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

nextToken

次のページの結果を取得する以前のリクエストによって返される、ページ分割トークンを指定します。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

レスポンスの構文

```
{
   "metrics": [
      {
         "eventType": "string",
         "expression": "string",
         "metricName": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

metrics

指定されたメトリクス属性のメトリクス。

型: MetricAttribute オブジェクトの配列

配列メンバー: 最大数は 10 項目です。

nextToken

次のページの結果を取得する以前の `ListMetricAttributionMetricsResponse` リクエストによって返される、ページ分割トークンを指定します。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：\p{ASCII}{0,1500}

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListMetricAttributions

サービス: Amazon Personalize

メトリクス属性を一覧表示します。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

メトリクス属性のデータセットグループ Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

結果の 1 ページで返されるメトリクス属性の最大数です。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

次のページの結果を取得する以前のリクエストによって返される、ページ分割トークンを指定します。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "metricAttributions": [
      {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "metricAttributionArn": "string",
         "name": "string",
         "status": "string"
      }
   ],
   "nextToken": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

metricAttributions

メトリクス属性のリスト。

型: MetricAttributionSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

nextToken

次のページの結果を取得する以前のリクエストによって返される、ページ分割トークンを指定します。

型: 文字列

長さの制限: 最大長は 1500 です。

パターン：`\p{ASCII}{0,1500}`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListRecipes

サービス: Amazon Personalize

利用可能なレシピのリストを返します。レスポンスは、レシピの Amazon リソースネーム (ARN) など、各レシピのプロパティを提供します。

リクエストの構文

```
{
   "domain": "string",
   "maxResults": number,
   "nextToken": "string",
   "recipeProvider": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

domain

ドメインデータセットグループのドメインごとに返されたレシピをフィルタリングします。このドメインのレシピ (ドメインデータセットグループのユースケース) のみがレスポンスに含まれます。ドメインを指定しない場合、すべてのレシピが返されます。

型: 文字列

有効な値：ECOMMERCE | VIDEO_ON_DEMAND

必須：いいえ

maxResults

返されるレシピの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

レシピの次のセットを取得するために、前の ListRecipes に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

recipeProvider

デフォルトは SERVICE です。

型: 文字列

有効な値：SERVICE

必須：いいえ

レスポンスの構文

```
{
   "nextToken": "string",
   "recipes": [
      {
         "creationDateTime": number,
         "domain": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "recipeArn": "string",
         "status": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

nextToken

次のレシピセットを取得するためのトークン。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

recipes

使用可能なレシピのリスト。

型: RecipeSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK

- AWS ルビー V3 用 SDK

ListRecommenders

サービス: Amazon Personalize

特定のドメインデータセットグループ内のレコメンダーのリストを返します。ドメインデータセットグループが指定されていない場合、アカウントに関連付けられているすべてのレコメンダーが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) など、各レコメンダーのプロパティを提供します。レコメンダーの詳細については、を参照してくださいCreateRecommender。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

レコメンダーを一覧表示するドメインデータセットグループの Amazon リソースネーム (ARN)。ドメインデータセットグループが指定されていない場合、アカウントに関連付けられているすべてのレコメンダーが一覧表示されます。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるレコメンダーの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

レコメンダーの次のセットを取得するために、前の ListRecommenders に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "nextToken": "string",
   "recommenders": [
      {
         "creationDateTime": number,
         "datasetGroupArn": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "recipeArn": "string",
         "recommenderArn": "string",
         "recommenderConfig": {
            "enableMetadataWithRecommendations": boolean,
            "itemExplorationConfig": {
               "string" : "string"
            },
            "minRecommendationRequestsPerSecond": number,
            "trainingDataConfig": {
               "excludedDatasetColumns": {
                  "string" : [ "string" ]
               }
            }
         },
         "status": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

nextToken

レコメンダーの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

recommenders

レコメンダーのリスト。

型: RecommenderSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード : 400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード : 400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListSchemas

サービス: Amazon Personalize

アカウントに関連付けられているスキーマのリストを返します。レスポンスは、Amazon リソースネーム (ARN) など、各スキーマのプロパティを提供します。スキーマの詳細については、を参照してくださいCreateSchema。

リクエストの構文

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

返されるスキーマの最大数。

タイプ : 整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

スキーマの次のセットを取得するために、前の ListSchemas に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
```

```
   "nextToken": "string",
   "schemas": [
      {
         "creationDateTime": number,
         "domain": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "schemaArn": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

nextToken

スキーマの次のセットを取得するために使用するトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

schemas

スキーマのリスト。

型: DatasetSchemaSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

エラー

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListSolutions

サービス: Amazon Personalize

特定のデータセットグループ内のソリューションのリストを返します。データセットグループが指定されていない場合、アカウントに関連付けられているすべてのソリューションが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) など、各ソリューションのプロパティを提供します。ソリューションの詳細については、を参照してくださいCreateSolution。

リクエストの構文

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetGroupArn

データセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

maxResults

返されるソリューションの最大数。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

ソリューションの次のセットを取得するために、前の ListSolutions に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

必須: いいえ

レスポンスの構文

```
{
   "nextToken": "string",
   "solutions": [
      {
         "creationDateTime": number,
         "lastUpdatedDateTime": number,
         "name": "string",
         "recipeArn": "string",
         "solutionArn": "string",
         "status": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

nextToken

ソリューションの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

solutions

現在のソリューションのリスト。

型: SolutionSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListSolutionVersions

サービス: Amazon Personalize

特定のソリューションのソリューションバージョンのリストを返します。ソリューションが指定されていない場合、アカウントに関連付けられているすべてのソリューションバージョンが一覧表示されます。レスポンスは、Amazon リソースネーム (ARN) など、各ソリューションバージョンのプロパティを提供します。

リクエストの構文

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

maxResults

返されるソリューションバージョンの最大数。

タイプ : 整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須: いいえ

nextToken

ソリューションバージョンの次のセットを取得するために、前の `ListSolutionVersions` に対する呼び出しから返されたトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: `\p{ASCII}{0,1500}`

必須: いいえ

solutionArn

ソリューションの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

レスポンスの構文

```
{
   "nextToken": "string",
   "solutionVersions": [
      {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "solutionVersionArn": "string",
         "status": "string",
         "trainingMode": "string",
         "trainingType": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

nextToken

ソリューションバージョンの次のセットを取得するためのトークン (存在する場合)。

型: 文字列

長さの制限: 最大長は 1500 です。

Pattern: \p{ASCII}{0,1500}

solutionVersions

バージョンのプロパティを記述するソリューションバージョンのリスト。

型: SolutionVersionSummary オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

InvalidNextTokenException

トークンが無効です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK

- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

ListTagsForResource

サービス: Amazon Personalize

リソースに添付されているタグのリストを取得します。

リクエストの構文

```
{
   "resourceArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

resourceArn

リソースの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

tags

リソースのタグ。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK

- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

StartRecommender

サービス: Amazon Personalize

非アクティブなレコメンダーを起動します。レコメンダーを起動しても新しいモデルは作成されませんが、レコメンダーへの請求と自動再トレーニングが再開されます。

リクエストの構文

```
{
   "recommenderArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

recommenderArn

起動するレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "recommenderArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

recommenderArn

起動したレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

StopRecommender

サービス: Amazon Personalize

アクティブなレコメンダーを停止します。レコメンダーを停止すると、レコメンダーへの請求と自動再トレーニングが停止します。

リクエストの構文

```
{
   "recommenderArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

recommenderArn

停止するレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
   "recommenderArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

recommenderArn

停止したレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

StopSolutionVersionCreation

サービス: Amazon Personalize

CREATE_PENDING または CREATE IN_PROGRESS の状態にあるソリューションバージョンの作成を停止します。

ソリューションバージョンの現在の状態に応じて、ソリューションバージョンの状態は次のように変化します。

- CREATE_PENDING > CREATE_STOPPED

  または

- CREATE_IN_PROGRESS > CREATE_STOPPING > CREATE_STOPPED

ソリューションバージョンの作成を停止するまでに完了したすべてのトレーニングについて請求されます。停止したソリューションバージョンの作成を再開することはできません。

リクエストの構文

```
{
   "solutionVersionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

solutionVersionArn

作成を停止するソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

TagResource

サービス: Amazon Personalize

リソースにタグのリストを追加します。

リクエストの構文

```
{
   "resourceArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

resourceArn

リソースの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

tags

リソースに適用するタグ。詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。

タイプ：Tag オブジェクトの配列

配列メンバー: 最小数は 0 項目です。最大数は 200 項目です。

必須: はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

LimitExceededException

1 秒あたりのリクエスト数の上限を超えています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagsException

このリソースに適用できるタグの最大数を超過しています。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++

- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

UntagResource

サービス: Amazon Personalize

リソースに添付されている指定されたタグを削除します。詳細については、「Amazon Personalize リソースからのタグの削除」を参照してください。

リクエストの構文

```
{
   "resourceArn": "string",
   "tagKeys": [ "string" ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

resourceArn

リソースの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

tagKeys

削除するタグのキー。

タイプ：文字列の配列

配列メンバー：最小数は 0 項目です。最大数は 200 項目です。

長さの制限：最小長は 1 です。最大長は 128 です。

パターン: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

必須：はい

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

TooManyTagKeysException

リクエストには、リソースに関連付けることができる数よりも多くのタグキーが含まれています (リソースごとに 50 個のタグキー)。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK

- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

UpdateCampaign

サービス: Amazon Personalize

キャンペーンを更新して、既存のキャンペーンで再トレーニング済みのソリューションバージョンをデプロイしたり、キャンペーンを変更したりminProvisionedTPS、キャンペーンの設定を変更したりします。たとえば、enableMetadataWithRecommendations既存のキャンペーンを true に設定できます。

キャンペーンを更新して自動的に最新のソリューションバージョンを使用し始めるには、以下を指定します。

- SolutionVersionArnパラメータには、ソリューションの Amazon リソースネーム (ARN) SolutionArn/$LATEST をフォーマットで指定します。
- でcampaignConfig、syncWithLatestSolutionVersionをに設定します。true

キャンペーンを更新するには、キャンペーンのステータスが ACTIVE または CREATE FAILED である必要があります。DescribeCampaignオペレーションを使用してキャンペーンのステータスを確認します。

> **Note**
>
> 更新の進行中でも、キャンペーンからレコメンデーションを取得できます。最新のキャンペーン更新ステータスが Active になるまで、キャンペーンは以前のソリューションバージョンとキャンペーン設定を使用してレコメンデーションを生成します。

コードサンプルなど、キャンペーンの更新について詳しくは、「キャンペーンの更新」を参照してください。キャンペーンの作成について詳細は、「キャンペーンの作成」を参照してください。

リクエストの構文

```
{
   "campaignArn": "string",
   "campaignConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "syncWithLatestSolutionVersion": boolean
```

```
    },
    "minProvisionedTPS": number,
    "solutionVersionArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignArn

キャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

campaignConfig

キャンペーンの設定の詳細。

タイプ：CampaignConfig オブジェクト

必須: いいえ

minProvisionedTPS

Amazon Personalize がサポートする、リクエストされた 1 秒あたりの最小プロビジョンドトランザクション (推奨) を指定します。`minProvisionedTPS` を高く設定すると請求額が増加します。最初は `minProvisionedTPS` に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、`minProvisionedTPS`必要に応じて増やします。

タイプ：整数

有効な範囲: 最小値 は 1 です。

必須: いいえ

solutionVersionArn

デプロイする新しいモデルの Amazon リソースネーム (ARN)。ソリューションの最新のソリューションバージョンを指定するには、ソリューションの ARN *SolutionArn/$LATEST* をフォーマットで指定します。syncWithLatestSolutionVersionで設定した場合は、この形式を使用する必要があります。True CampaignConfig

ソリューションの最新のソリューションバージョンではないモデルをデプロイするには、ソリューションバージョンの ARN を指定します。

キャンペーンの自動更新について詳しくは、「キャンペーンの自動更新の有効化」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

レスポンスの構文

```
{
   "campaignArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

campaignArn

リクエストで指定されたものと同じキャンペーン ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン：arn:([a-z\d-]+):personalize:.*:.*:.+

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

UpdateDataset

サービス: Amazon Personalize

データセットを更新して、そのスキーマを新しいスキーマまたは既存のスキーマに置き換えます。詳細については、「データセットのスキーマの置き換え」を参照してください。

リクエストの構文

```
{
   "datasetArn": "string",
   "schemaArn": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

更新するデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

schemaArn

使用するデータセットのAmazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
{
```

```
   "datasetArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

datasetArn

更新したデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス

- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

UpdateMetricAttribution

サービス: Amazon Personalize

メトリクス属性を更新します。

リクエストの構文

```
{
   "addMetrics": [
      {
         "eventType": "string",
         "expression": "string",
         "metricName": "string"
      }
   ],
   "metricAttributionArn": "string",
   "metricsOutputConfig": {
      "roleArn": "string",
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "removeMetrics": [ "string" ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

addMetrics

メトリクス属性に新しいメトリック属性を追加します。

型: MetricAttribute オブジェクトの配列

配列メンバー: 最大数は 10 項目です。

必須: いいえ

metricAttributionArn

更新するメトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

metricsOutputConfig

メトリクス属性の出力設定。

タイプ：MetricAttributionOutput オブジェクト

必須: いいえ

removeMetrics

メトリクス属性からメトリクス属性を削除します。

タイプ：文字列の配列

配列メンバー: 最大数は 10 項目です。

長さの制限：最大長は 256 です。

必須: いいえ

レスポンスの構文

```
{
   "metricAttributionArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

metricAttributionArn

更新するメトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceAlreadyExistsException

指定したリソースはすでに存在しています。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK

- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

UpdateRecommender

サービス: Amazon Personalize

レコメンダーを更新して、レコメンダーの設定を変更します。レコメンダーを更新してトレーニングに使用した列を変更すると、Amazon Personalize はレコメンダーを裏付けるモデルの完全な再トレーニングを自動的に開始します。更新が完了しても、レコメンダーからレコメンデーションを取得できます。レコメンダーは、更新が完了するまで以前の設定を使用します。この更新のステータスを追跡するには、`latestRecommenderUpdate`DescribeRecommender操作で返されたものを使用します。

リクエストの構文

```
{
   "recommenderArn": "string",
   "recommenderConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "minRecommendationRequestsPerSecond": number,
      "trainingDataConfig": {
         "excludedDatasetColumns": {
            "string" : [ "string" ]
         }
      }
   }
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

recommenderArn

変更するレコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

recommenderConfig

レコメンダーの設定の詳細。

型: RecommenderConfig オブジェクト

必須: はい

レスポンスの構文

```
{
   "recommenderArn": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

recommenderArn

リクエストで指定されたものと同じレコメンダー Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

Amazon Personalize Events

次のアクションは、Amazon Personalize Events でサポートされています。

- PutActionInteractions
- PutActions
- PutEvents
- PutItems
- PutUsers

PutActionInteractions

サービス: Amazon Personalize Events

アクションインタラクションのイベントデータを記録します。アクションインタラクションイベントは、ユーザーとアクションの間のインタラクションです。例えば、ユーザーがメンバーシッププログラムへの登録やアプリのダウンロードなどのアクションを実行する場合です。

アクションインタラクションの記録の詳細については、「アクションインタラクションイベントの記録」を参照してください。アクションデータセット内のアクションの詳細については、「Actions dataset」を参照してください。

リクエストの構文

```
POST /action-interactions HTTP/1.1
Content-type: application/json

{
   "actionInteractions": [
      {
         "actionId": "string",
         "eventId": "string",
         "eventType": "string",
         "impression": [ "string" ],
         "properties": "string",
         "recommendationId": "string",
         "sessionId": "string",
         "timestamp": number,
         "userId": "string"
      }
   ],
   "trackingId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

actionInteractions

セッションからのアクションインタラクションのイベントのリスト。

型: ActionInteraction オブジェクトの配列

配列メンバー：最小数は 1 項目です。最大数は 10 項目です。

必須: はい

trackingId

アクションインタラクションイベントトラッカーの ID。Action インタラクションデータセットを作成すると、Amazon Personalize は自動的にアクションインタラクションイベントトラッカーを作成します。詳細については、「Action interaction event tracker ID」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード: 409

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

PutActions

サービス: Amazon Personalize Events

1 つ以上のアクションを Actions データセットに追加します。詳細については、「Importing actions individually」を参照してください。

リクエストの構文

```
POST /actions HTTP/1.1
Content-type: application/json

{
   "actions": [
      {
         "actionId": "string",
         "properties": "string"
      }
   ],
   "datasetArn": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

actions

アクションデータのリスト。

型: Action オブジェクトの配列

配列メンバー : 最小数は 1 項目です。最大数は 10 項目です。

必須: はい

datasetArn

1 つまたは複数のアクションを追加する Actions データセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード: 409

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET

- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

PutEvents

サービス: Amazon Personalize Events

アイテムインタラクションのイベントデータを記録します。詳細については、「Recording item interaction events」を参照してください。

> **Note**
>
> AWS Lambda 関数を使用して PutEvents オペレーションを呼び出す場合、関数の実行ロールには、 `Resource`要素`*`のワイルドカードを使用して`personalize:PutEvents`アクションを実行するアクセス許可が必要です。

リクエストの構文

```
POST /events HTTP/1.1
Content-type: application/json

{
   "eventList": [
      {
         "eventId": "string",
         "eventType": "string",
         "eventValue": number,
         "impression": [ "string" ],
         "itemId": "string",
         "metricAttribution": {
            "eventAttributionSource": "string"
         },
         "properties": "string",
         "recommendationId": "string",
         "sentAt": number
      }
   ],
   "sessionId": "string",
   "trackingId": "string",
   "userId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

eventList

セッションからのイベントデータのリスト。

型: Event オブジェクトの配列

配列メンバー：最小数は 1 項目です。最大数は 10 項目です。

必須：はい

sessionId

ユーザーの訪問に関連付けられたセッション ID。アプリケーションは、ユーザーが最初にウェブサイトにアクセスしたとき、またはアプリケーションを使用したときに、sessionId を生成します。Amazon Personalize は、sessionId を使用して、ユーザーがログインする前にイベントをユーザーに関連付けます。詳細については、「Recording item interaction events」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須：はい

trackingId

イベントの追跡 ID。ID は、CreateEventトラッカー API の呼び出しによって生成されます。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須：はい

userId

イベントに関連付けられているユーザー。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutItems

サービス: Amazon Personalize Events

1 つ以上のアイテムを Items データセットに追加します。詳細については、「Importing items individually」を参照してください。

リクエストの構文

```
POST /items HTTP/1.1
Content-type: application/json

{
   "datasetArn": "string",
   "items": [
      {
         "itemId": "string",
         "properties": "string"
      }
   ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

1 つまたは複数のアイテムを追加する Items データセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

items

アイテムデータのリスト。

型: Item オブジェクトの配列

配列メンバー : 最小数は 1 項目です。最大数は 10 項目です。

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード : 400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード: 409

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK

- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

PutUsers

サービス: Amazon Personalize Events

1 名以上のユーザーを Users データセットに追加します。詳細については、「Importing users individually」を参照してください。

リクエストの構文

```
POST /users HTTP/1.1
Content-type: application/json

{
   "datasetArn": "string",
   "users": [
      {
         "properties": "string",
         "userId": "string"
      }
   ]
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

datasetArn

1 名または複数のユーザーを追加する Users データセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

users

ユーザーデータのリスト。

型: User オブジェクトの配列

配列メンバー : 最小数は 1 項目です。最大数は 10 項目です。

必須: はい

レスポンスの構文

```
HTTP/1.1 200
```

レスポンス要素

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 200 レスポンスを返します。

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード : 400

ResourceInUseException

指定されたリソースは使用中です。

HTTP ステータスコード: 409

ResourceNotFoundException

指定されたリソースが見つかりませんでした。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK

- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK
- AWS ルビー V3 用 SDK

Amazon Personalize Runtime

次のアクションは、Amazon Personalize Runtime でサポートされています。

- GetActionRecommendations
- GetPersonalizedRanking
- GetRecommendations

GetActionRecommendations

サービス: Amazon Personalize Runtime

予測スコアで降順にソートされたレコメンデーションアクションのリストを返します。PERSONALIZED_ACTIONS レシピでトレーニングされたソリューションバージョンをデプロイするカスタムキャンペーンがある場合は、GetActionRecommendations API を使用してください。

PERSONALIZED_ACTIONS レシピの詳細については、「PERSONALIZED_ACTIONS recipes」を参照してください。アクションレコメンデーションの取得については、「Getting action recommendations」を参照してください。

リクエストの構文

```
POST /action-recommendations HTTP/1.1
Content-type: application/json

{
   "campaignArn": "string",
   "filterArn": "string",
   "filterValues": {
      "string" : "string"
   },
   "numResults": number,
   "userId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignArn

アクションレコメンデーションの取得に使用するキャンペーンの Amazon リソースネーム (ARN)。このキャンペーンでは、PERSONALIZED_ACTIONS レシピでトレーニングされたソリューションバージョンをデプロイする必要があります。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

filterArn

返されたレコメンデーションに適用するフィルターの ARN。詳細については、「レコメンデーションのフィルタリング」を参照してください。

このパラメータを使用するときは、フィルターのリソースが ACTIVE であることを確認してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

filterValues

レコメンデーションをフィルタリングするときに使用する値。フィルター式の各プレースホルダーパラメータについて、(大文字と小文字が一致した状態で) パラメータ名をキーとして指定し、フィルター値を対応する値として指定します。1 つのパラメータの複数の値をコンマで区切ります。

INCLUDE 要素を使用してアクションを含めるフィルター式について、式で定義されているすべてのパラメータの値を指定する必要があります。EXCLUDE 要素を使用してアクションを除外する式を含むフィルターについては、`filter-values` を省略できます。この場合、Amazon Personalize は、式のその部分を使用してレコメンデーションをフィルタリングしません。

詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 25 です。

キーの長さの制限: 最大長は 50 です。

キーパターン: `[A-Za-z0-9_]+`

値の長さの制限: 最大長は 1000 です。

必須: いいえ

numResults

返される結果の数。デフォルトは 5 です。最大は 100 です。

タイプ: 整数

値の範囲: 最小値 は 0 です。

必須: いいえ

userId

アクションレコメンデーションを提供するユーザーのユーザー ID。

型: 文字列

長さの制限 : 最大長は 256 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "actionList": [
      {
         "actionId": "string",
         "score": number
      }
   ],
   "recommendationId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

actionList

予測スコアで降順にソートされたアクションレコメンデーションのリスト。リストには最大 100 個のアクションを含めることができます。アクションスコアの詳細については、「How action recommendation scoring works」を参照してください。

型: PredictedAction オブジェクトの配列

recommendationId

レコメンデーションの ID。

型: 文字列

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード : 400

ResourceNotFoundException

指定されたリソースは存在しません。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS Go バージョン 2 用 SDK
- AWS Java V2 用 SDK
- AWS V3 用 JavaScript SDK
- AWS PHP V3 用SDK
- AWS Python 用 SDK

- AWS ルビー V3 用 SDK

GetPersonalizedRanking

サービス: Amazon Personalize Runtime

特定のユーザーに推奨されるアイテムのリストを再ランク付けします。リストの最初のアイテムは、ユーザーが関心を持つ可能性が最も高いアイテムとみなされます。

> **Note**
>
> キャンペーンにデプロイするソリューションが PERSONALIZED_RANKING タイプのレシピを使用して作成済みであることが必要です。

リクエストの構文

```
POST /personalize-ranking HTTP/1.1
Content-type: application/json

{
   "campaignArn": "string",
   "context": {
      "string" : "string"
   },
   "filterArn": "string",
   "filterValues": {
      "string" : "string"
   },
   "inputList": [ "string" ],
   "metadataColumns": {
      "string" : [ "string" ]
   },
   "userId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignArn

パーソナライズされたランキングの生成に使用するキャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):personalize:.*:.*:.+`

必須：はい

context

レコメンデーションを取得するときに使用するコンテキストメタデータ。コンテキストメタデータには、ユーザーの現在の場所やデバイスタイプなど、ユーザーのレコメンデーションを取得するときに関連する可能性のあるインタラクション情報が含まれます。

タイプ: 文字列から文字列へのマップ

マップエントリ: アイテムの最大数は 150 です。

キーの長さの制限: 最大長は 150 です。

キーパターン: `[A-Za-z\d_]+`

値の長さの制限: 最大長は 1000 です。

必須: いいえ

filterArn

特定のユーザーのレコメンデーションからアイテムを含めたり、アイテムを除外したりするために作成したフィルターの Amazon リソースネーム (ARN)。詳細については、「レコメンデーションのフィルタリング」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

filterValues

レコメンデーションをフィルタリングするときに使用する値。フィルター式の各プレースホルダーパラメータについて、(大文字と小文字が一致した状態で) パラメータ名をキーとして指定し、フィルター値を対応する値として指定します。1 つのパラメータの複数の値をコンマで区切ります。

INCLUDE 要素を使用してアイテムを含めるフィルター式について、式で定義されているすべてのパラメータの値を指定する必要があります。EXCLUDE 要素を使用してアイテムを除外する式を含むフィルターについては、filter-values を省略できます。この場合、Amazon Personalize は、レコメンデーションをフィルタリングするために式のその部分を使用しません。

詳細については、「レコメンデーションのフィルタリング」を参照してください。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 25 です。

キーの長さの制限: 最大長は 50 です。

キーパターン: [A-Za-z0-9_]+

値の長さの制限: 最大長は 1000 です。

必須: いいえ

inputList

ランク付けするアイテムのリスト (itemId による)。アイテムがトレーニングデータセットに含まれていなかった場合、そのアイテムは再ランク付けされたリストの最後に追加されます。レコメンデーションにメタデータを含める場合、最大数は 50 です。それ以外の場合、最大数は 500 です。

タイプ：文字列の配列

長さの制限：最大長は 256 です。

必須: はい

metadataColumns

キャンペーンを作成または更新したときにレコメンデーションのメタデータを有効にした場合は、アイテムデータセットのメタデータ列を指定して、パーソナライズしたランキングに含める

ようにします。マップキーは ITEMS で、値はアイテムデータセットの列名のリストです。指定できる列の最大数は 10 です。

キャンペーンのメタデータを有効にする方法については、「Enabling metadata in recommendations for a campaign」を参照してください。

タイプ: 文字列マップの配列への文字列

マップエントリ: アイテムの最大数は 1 です。

キーの長さの制限: 最大長は 256 です。

配列メンバー: 最大数は 99 アイテムです。

長さの制限: 最大長は 1,024 です。

必須: いいえ

userId

キャンペーンによるパーソナライズされたランキングの提供先とするユーザー。

型: 文字列

長さの制限：最大長は 256 です。

必須: はい

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json

{
   "personalizedRanking": [
      {
         "itemId": "string",
         "metadata": {
            "string" : "string"
         },
         "promotionName": "string",
         "reason": [ "string" ],
         "score": number
```

```
        }
    ],
    "recommendationId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

personalizedRanking

アイテムのリスト (ユーザーが関心を持つ可能性が最も高い順)。最大数は 500 です。

型: PredictedItem オブジェクトの配列

recommendationId

レコメンデーションの ID。

型: 文字列

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード : 400

ResourceNotFoundException

指定されたリソースは存在しません。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetRecommendations

サービス: Amazon Personalize Runtime

推奨アイテムのリストを返します。キャンペーンについて、キャンペーンの Amazon リソースネーム (ARN) は必須であり、必要なユーザーとアイテムの入力は、キャンペーンをサポートするソリューションの作成に使用されたレシピタイプによって次のように異なります。

- USER_PERSONALIZATION - `userId` 必須、`itemId` 未使用
- RELATED_ITEMS - `itemId` 必須、`userId` 未使用

> **Note**
>
> タイプ PERSONALIZED_RANKING のレシピを使用して作成されたソリューションにサポートされたキャンペーンは、GetPersonalizedRanking API を使用します。

レコメンダーの場合、レコメンダーの ARN が必要であり、必要なアイテムとユーザー入力は、レコメンダーをサポートするユースケース (ドメインベースのレシピ) によって異なります。ユースケースにおける要件の詳細については、「レコメンダーのユースケースの選択」を参照してください。

リクエストの構文

```
POST /recommendations HTTP/1.1
Content-type: application/json

{
   "campaignArn": "string",
   "context": {
      "string" : "string"
   },
   "filterArn": "string",
   "filterValues": {
      "string" : "string"
   },
   "itemId": "string",
   "metadataColumns": {
      "string" : [ "string" ]
   },
   "numResults": number,
   "promotions": [
```

```
      {
         "filterArn": "string",
         "filterValues": {
            "string" : "string"
         },
         "name": "string",
         "percentPromotedItems": number
      }
   ],
   "recommenderArn": "string",
   "userId": "string"
}
```

URI リクエストパラメータ

リクエストでは URI パラメータを使用しません。

リクエストボディ

リクエストは以下の JSON 形式のデータを受け入れます。

campaignArn

レコメンデーションの取得に使用するキャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

context

レコメンデーションを取得するときに使用するコンテキストメタデータ。コンテキストメタデータには、ユーザーの現在の場所やデバイスタイプなど、ユーザーのレコメンデーションを取得するときに関連する可能性のあるインタラクション情報が含まれます。

タイプ: 文字列から文字列へのマップ

マップエントリ: アイテムの最大数は 150 です。

キーの長さの制限: 最大長は 150 です。

キーパターン: [A-Za-z\d_]+

値の長さの制限: 最大長は 1000 です。

必須: いいえ

filterArn

返されたレコメンデーションに適用するフィルターの ARN。詳細については、「レコメンデーションのフィルタリング」を参照してください。

このパラメータを使用するときは、フィルターのリソースが ACTIVE であることを確認してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

filterValues

レコメンデーションをフィルタリングするときに使用する値。フィルター式の各プレースホルダーパラメータについて、(大文字と小文字が一致した状態で) パラメータ名をキーとして指定し、フィルター値を対応する値として指定します。1 つのパラメータの複数の値をコンマで区切ります。

INCLUDE 要素を使用してアイテムを含めるフィルター式について、式で定義されているすべてのパラメータの値を指定する必要があります。EXCLUDE 要素を使用してアイテムを除外する式を含むフィルターについては、filter-values を省略できます。この場合、Amazon Personalize は、レコメンデーションをフィルタリングするために式のその部分を使用しません。

詳細については、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 25 です。

キーの長さの制限: 最大長は 50 です。

キーパターン: [A-Za-z0-9_]+

値の長さの制限: 最大長は 1000 です。

必須: いいえ

itemId

レコメンデーションを提供するアイテム ID。

RELATED_ITEMS レシピタイプに必要です。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

metadataColumns

キャンペーンまたはレコメンダーを作成または更新したときにレコメンデーションのメタデータを有効にした場合は、アイテムデータセットのメタデータ列をアイテムレコメンデーションに含めるように指定します。マップキーは ITEMS で、値はアイテムデータセットの列名のリストです。指定できる列の最大数は 10 です。

キャンペーンのメタデータを有効にする方法については、「Enabling metadata in recommendations for a campaign」を参照してください。レコメンダーのメタデータを有効にする方法については、「Enabling metadata in recommendations for a recommender」を参照してください。

タイプ: 文字列マップの配列への文字列

マップエントリ: アイテムの最大数は 1 です。

キーの長さの制限: 最大長は 256 です。

配列メンバー: 最大数は 99 アイテムです。

長さの制限: 最大長は 1,024 です。

必須: いいえ

numResults

返される結果の数。デフォルトは 25 です。レコメンデーションにメタデータを含める場合、最大数は 50 です。それ以外の場合、最大数は 500 です。

タイプ: 整数

値の範囲: 最小値 は 0 です。

必須: いいえ

promotions

レコメンデーションリクエストに適用するプロモーション。プロモーションは、設定可能なおすすめアイテムのサブセットに適用される追加のビジネスルールを定義します。

型: Promotion オブジェクトの配列

配列メンバー: 最大数は 1 項目です。

必須: いいえ

recommenderArn

レコメンデーションを取得するために使用するレコメンダーの Amazon リソースネーム (ARN)。ドメインユースケース向けのレコメンダーを含むドメインデータセットグループを作成した場合は、レコメンダー ARN を提供します。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

userId

レコメンデーションを提供するユーザー ID。

`USER_PERSONALIZATION` レシピタイプに必要です。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

レスポンスの構文

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "itemList": [
      {
         "itemId": "string",
         "metadata": {
            "string" : "string"
         },
         "promotionName": "string",
         "reason": [ "string" ],
         "score": number
      }
   ],
   "recommendationId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

itemList

予測スコアでソートされたレコメンデーションのリスト (降順)。リストには最大 500 個のアイテムを含めることができます。

型: PredictedItem オブジェクトの配列

recommendationId

レコメンデーションの ID。

型: 文字列

エラー

InvalidInputException

フィールドまたはパラメータに有効な値を指定します。

HTTP ステータスコード：400

ResourceNotFoundException

指定されたリソースは存在しません。

HTTP ステータスコード: 404

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

データ型

次のデータ型は、Amazon Personalize でサポートされています。

- Algorithm
- AlgorithmImage
- AutoMLConfig
- AutoMLResult
- AutoTrainingConfig
- BatchInferenceJob
- BatchInferenceJobConfig
- BatchInferenceJobInput
- BatchInferenceJobOutput

- BatchInferenceJobSummary
- BatchSegmentJob
- BatchSegmentJobInput
- BatchSegmentJobOutput
- BatchSegmentJobSummary
- Campaign
- CampaignConfig
- CampaignSummary
- CampaignUpdateSummary
- CategoricalHyperParameterRange
- ContinuousHyperParameterRange
- DataDeletionJob
- DataDeletionJobSummary
- Dataset
- DatasetExportJob
- DatasetExportJobOutput
- DatasetExportJobSummary
- DatasetGroup
- DatasetGroupSummary
- DatasetImportJob
- DatasetImportJobSummary
- DatasetSchema
- DatasetSchemaSummary
- DatasetSummary
- DatasetUpdateSummary
- DataSource
- DefaultCategoricalHyperParameterRange
- DefaultContinuousHyperParameterRange
- DefaultHyperParameterRanges
- DefaultIntegerHyperParameterRange

- EventTracker
- EventTrackerSummary
- FeatureTransformation
- FieldsForThemeGeneration
- Filter
- FilterSummary
- HPOConfig
- HPOObjective
- HPOResourceConfig
- HyperParameterRanges
- IntegerHyperParameterRange
- MetricAttribute
- MetricAttribution
- MetricAttributionOutput
- MetricAttributionSummary
- OptimizationObjective
- Recipe
- RecipeSummary
- Recommender
- RecommenderConfig
- RecommenderSummary
- RecommenderUpdateSummary
- S3DataConfig
- Solution
- SolutionConfig
- SolutionSummary
- SolutionVersion
- SolutionVersionSummary
- Tag
- ThemeGenerationConfig

- TrainingDataConfig
- TunedHPOParams

次のデータ型は、Amazon Personalize Events でサポートされています。

- Action
- ActionInteraction
- Event
- Item
- MetricAttribution
- User

次のデータ型は、Amazon Personalize Runtime でサポートされています。

- PredictedAction
- PredictedItem
- Promotion

Amazon Personalize

次のデータ型は、Amazon Personalize でサポートされています。

- Algorithm
- AlgorithmImage
- AutoMLConfig
- AutoMLResult
- AutoTrainingConfig
- BatchInferenceJob
- BatchInferenceJobConfig
- BatchInferenceJobInput
- BatchInferenceJobOutput
- BatchInferenceJobSummary
- BatchSegmentJob

- BatchSegmentJobInput
- BatchSegmentJobOutput
- BatchSegmentJobSummary
- Campaign
- CampaignConfig
- CampaignSummary
- CampaignUpdateSummary
- CategoricalHyperParameterRange
- ContinuousHyperParameterRange
- DataDeletionJob
- DataDeletionJobSummary
- Dataset
- DatasetExportJob
- DatasetExportJobOutput
- DatasetExportJobSummary
- DatasetGroup
- DatasetGroupSummary
- DatasetImportJob
- DatasetImportJobSummary
- DatasetSchema
- DatasetSchemaSummary
- DatasetSummary
- DatasetUpdateSummary
- DataSource
- DefaultCategoricalHyperParameterRange
- DefaultContinuousHyperParameterRange
- DefaultHyperParameterRanges
- DefaultIntegerHyperParameterRange
- EventTracker
- EventTrackerSummary

- FeatureTransformation
- FieldsForThemeGeneration
- Filter
- FilterSummary
- HPOConfig
- HPOObjective
- HPOResourceConfig
- HyperParameterRanges
- IntegerHyperParameterRange
- MetricAttribute
- MetricAttribution
- MetricAttributionOutput
- MetricAttributionSummary
- OptimizationObjective
- Recipe
- RecipeSummary
- Recommender
- RecommenderConfig
- RecommenderSummary
- RecommenderUpdateSummary
- S3DataConfig
- Solution
- SolutionConfig
- SolutionSummary
- SolutionVersion
- SolutionVersionSummary
- Tag
- ThemeGenerationConfig
- TrainingDataConfig
- TunedHPOParams

Algorithm

サービス: Amazon Personalize

カスタムアルゴリズムを記述します。

コンテンツ

algorithmArn

アルゴリズムの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

algorithmImage

アルゴリズムイメージの Docker コンテナの URI。

タイプ：AlgorithmImage オブジェクト

必須: いいえ

creationDateTime

アルゴリズムの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

defaultHyperParameterRanges

デフォルトのハイパーパラメータ、それらの範囲、およびそれらがチューニング可能かどうかを指定します。チューニング可能なハイパーパラメータは、ハイパーパラメータ最適化 (HPO) 中にその値を決定できます。

タイプ：DefaultHyperParameterRanges オブジェクト

必須: いいえ

defaultHyperParameters

デフォルトのハイパーパラメータを指定します。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

defaultResourceConfig

トレーニングジョブと並列トレーニングジョブのデフォルトの最大数を指定します。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

lastUpdatedDateTime

アルゴリズムの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

アルゴリズムの名前

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

roleArn

ロールの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

trainingInputMode

トレーニング入力モード。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

AlgorithmImage

サービス: Amazon Personalize

アルゴリズムイメージを記述します。

コンテンツ

dockerURI

アルゴリズムイメージの Docker コンテナの URI。

型: 文字列

長さの制限：最大長は 256 です。

必須: はい

name

アルゴリズムイメージの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

AutoMLConfig

サービス: Amazon Personalize

ソリューションが AutoML を実行する (`performAutoML`が true になる CreateSolution) と、Amazon Personalize は、指定されたリストのどのレシピが特定のメトリクスを最適化するかを判断します。その後、Amazon Personalize はそのレシピをソリューションに使用します。

コンテンツ

metricName

最適化するメトリクス。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

recipeList

候補レシピのリスト。

タイプ：文字列の配列

配列メンバー: 最大数は 100 項目です。

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

AutoMLResult

サービス: Amazon Personalize

ソリューションが AutoML を実行する場合 (`performAutoML`が true の場合 CreateSolution)、指定されたメトリックを最適に最適化したレシピを指定します。

コンテンツ

bestRecipeArn

最適なレシピの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

AutoTrainingConfig

サービス: Amazon Personalize

true `performAutoTraining` のときに使用する自動トレーニング設定。

コンテンツ

schedulingExpression

新しいソリューションバージョンを自動的にトレーニングする頻度を指定します。レート (値 単位) 形式でレート式を指定します。value には、1 から 30 までの数値を指定します。unit には、dayまたはを指定しますdays。たとえば、5 日ごとに新しいソリューションバージョンを自動的に作成するには、を指定しますrate(5 days)。デフォルトは 7 日ごとです。

auto トレーニングの詳細については、「ソリューションの作成と構成」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 16 です。

Pattern: `rate\(\d+ days?\)`

必須: いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchInferenceJob

サービス: Amazon Personalize

バッチ推論ジョブに関する情報が含まれます。

コンテンツ

batchInferenceJobArn

バッチ推論ジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

batchInferenceJobConfig

バッチ推論ジョブの設定の詳細の文字列から文字列へのマップ。

タイプ：BatchInferenceJobConfig オブジェクト

必須: いいえ

batchInferenceJobMode

ジョブのモード。

型: 文字列

有効な値：`BATCH_INFERENCE | THEME_GENERATION`

必須：いいえ

creationDateTime

バッチ推論ジョブの作成時刻。

型: タイムスタンプ

必須: いいえ

failureReason

バッチ推論ジョブが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

filterArn

バッチ推論ジョブで使用されるフィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

jobInput

バッチ推論ジョブの生成に使用される入力データの格納場所への Amazon S3 パス。

タイプ：BatchInferenceJobInput オブジェクト

必須: いいえ

jobName

バッチ推論ジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

jobOutput

バッチ推論ジョブによって生成される出力データを含む Amazon S3 バケット。

タイプ：BatchInferenceJobOutput オブジェクト

必須: いいえ

lastUpdatedDateTime

バッチ推論ジョブの最終更新時刻。

型: タイムスタンプ

必須: いいえ

numResults

バッチ推論ジョブによって生成されたレコメンデーションの数。この数には、失敗した入力レコードについて生成されたエラーメッセージが含まれます。

タイプ: 整数

必須: いいえ

roleArn

バッチ推論ジョブをリクエストした Amazon Identity and Access Management (IAM) ロールの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須: いいえ

solutionVersionArn

バッチ推論ジョブが作成されたソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

バッチ推論ジョブのステータス。ステータスは、次のいずれかの値です。

- 保留中

- IN PROGRESS
- ACTIVE
- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

themeGenerationConfig

ジョブのテーマ生成設定。

タイプ：ThemeGenerationConfig オブジェクト

必須: いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchInferenceJobConfig

サービス: Amazon Personalize

バッチ推論ジョブの設定の詳細。

コンテンツ

itemExplorationConfig

`explorationWeight` と `explorationItemAgeCutOff` を含む探索設定ハイパーパラメータ (Amazon Personalize がアイテムを推奨するときに使用するアイテム探索の量を設定するために使用するもの) を指定する文字列から文字列へのマップ。User-Personalization を参照してください。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchInferenceJobInput

サービス: Amazon Personalize

バッチ推論ジョブの入力設定。

コンテンツ

s3DataSource

入力データを含む Amazon S3 の口ケーションの URI。Amazon S3 バケットは、呼び出す API エンドポイントと同じリージョンにある必要があります。

型: S3DataConfig オブジェクト

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchInferenceJobOutput

サービス: Amazon Personalize

バッチ推論ジョブの出力設定パラメータ。

コンテンツ

s3DataDestination

バッチ推論ジョブの出力が保存される Amazon S3 バケットに関する情報。

型: S3DataConfig オブジェクト

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchInferenceJobSummary

サービス: Amazon Personalize

の短縮版です。BatchInferenceJobListBatchInferenceJobsこのオペレーションでは、バッチ推論ジョブの概要のリストが返されます。

コンテンツ

batchInferenceJobArn

バッチ推論ジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

batchInferenceJobMode

ジョブのモード。

型: 文字列

有効な値：`BATCH_INFERENCE | THEME_GENERATION`

必須：いいえ

creationDateTime

バッチ推論ジョブの作成時刻。

型: タイムスタンプ

必須: いいえ

failureReason

バッチ推論ジョブが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

jobName

バッチ推論ジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

lastUpdatedDateTime

バッチ推論ジョブの最終更新時刻。

型: タイムスタンプ

必須: いいえ

solutionVersionArn

バッチ推論ジョブで使用されるソリューションバージョンの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: arn:([a-z\d-]+):personalize:.*:.*:.+

必須: いいえ

status

バッチ推論ジョブのステータス。ステータスは、次のいずれかの値です。

- 保留中
- IN PROGRESS
- ACTIVE
- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchSegmentJob

サービス: Amazon Personalize

バッチセグメントジョブに関する情報が含まれます。

コンテンツ

batchSegmentJobArn

バッチセグメントジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

creationDateTime

バッチセグメントジョブの作成時刻。

型: タイムスタンプ

必須: いいえ

failureReason

バッチセグメントジョブが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

filterArn

バッチセグメントジョブで使用されるフィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

jobInput

バッチセグメントジョブの生成に使用される入力データの格納場所への Amazon S3 パス。

タイプ : BatchSegmentJobInput オブジェクト

必須: いいえ

jobName

バッチセグメントジョブの名前。

型: 文字列

長さの制限 : 最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

jobOutput

バッチセグメントジョブによって生成される出力データを含む Amazon S3 バケット。

タイプ : BatchSegmentJobOutput オブジェクト

必須: いいえ

lastUpdatedDateTime

バッチセグメントジョブの最終更新時刻。

型: タイムスタンプ

必須: いいえ

numResults

入力データの各行のバッチセグメントジョブによって生成された予測ユーザーの数。セグメントあたりのユーザーの最大数は 500 万人です。

タイプ: 整数

必須: いいえ

roleArn

バッチセグメントジョブをリクエストした Amazon Identity and Access Management (IAM) ロールの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須: いいえ

solutionVersionArn

バッチセグメントを生成するためにバッチセグメントジョブで使用されるソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

バッチセグメントジョブのステータス。ステータスは、次のいずれかの値です。

- 保留中
- IN PROGRESS
- ACTIVE
- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchSegmentJobInput

サービス: Amazon Personalize

バッチセグメントジョブの入力設定。

コンテンツ

s3DataSource

Amazon S3 の入力または出力バケットの設定の詳細。

型: S3DataConfig オブジェクト

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchSegmentJobOutput

サービス: Amazon Personalize

バッチセグメントジョブの出力設定パラメータ。

コンテンツ

s3DataDestination

Amazon S3 の入力または出力バケットの設定の詳細。

型: S3DataConfig オブジェクト

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

BatchSegmentJobSummary

サービス: Amazon Personalize

BatchSegmentJobデータ型を切り詰めたもの。ListBatchSegmentJobsオペレーションはバッチセグメントのジョブ概要のリストを返します。

コンテンツ

batchSegmentJobArn

バッチセグメントジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

creationDateTime

バッチセグメントジョブの作成時刻。

型: タイムスタンプ

必須: いいえ

failureReason

バッチセグメントジョブが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

jobName

バッチセグメントジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

lastUpdatedDateTime

バッチセグメントジョブの最終更新時刻。

型: タイムスタンプ

必須: いいえ

solutionVersionArn

バッチセグメントを生成するためにバッチセグメントジョブで使用されるソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

バッチセグメントジョブのステータス。ステータスは、次のいずれかの値です。

- 保留中
- IN PROGRESS
- ACTIVE
- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++

- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Campaign

サービス: Amazon Personalize

ソリューションバージョンのデプロイを記述するオブジェクト。キャンペーンの詳細については、を参照してくださいCreateCampaign。

コンテンツ

campaignArn

キャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

campaignConfig

キャンペーンの設定の詳細。

タイプ：CampaignConfig オブジェクト

必須: いいえ

creationDateTime

キャンペーンの作成日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

failureReason

キャンペーンが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

キャンペーンの最終更新日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

latestCampaignUpdate

キャンペーンの更新のプロパティの概要を提供します。詳細なリストについては、DescribeCampaignAPI を呼び出してください。

タイプ：CampaignUpdateSummary オブジェクト

必須: いいえ

minProvisionedTPS

1 秒あたりにリクエストされる最小プロビジョンドトランザクション (レコメンデーション) を指定します。minProvisionedTPS を高く設定すると請求額が増加します。最初は minProvisionedTPS に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、minProvisionedTPS必要に応じて増やします。

タイプ：整数

有効な範囲: 最小値 は 1 です。

必須: いいえ

name

キャンペーンの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

solutionVersionArn

キャンペーンが使用するソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

キャンペーンのステータス。

キャンペーンは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

CampaignConfig

サービス: Amazon Personalize

キャンペーンの設定の詳細。

コンテンツ

enableMetadataWithRecommendations

レコメンデーションを含むメタデータがキャンペーンで有効になっているかどうか。有効にすると、レコメンデーションのリクエストでアイテムデータセットの列を指定できます。Amazon Personalize は、レコメンデーションレスポンス内の各アイテムについてこのデータを返します。キャンペーンのメタデータを有効にする方法については、「Enabling metadata in recommendations for a campaign」を参照してください。

レコメンデーションのメタデータを有効にすると、追加費用が発生します。詳細については、「Amazon Personalize の料金」を参照してください。

型: ブール値

必須: いいえ

itemExplorationConfig

`explorationWeight` と `explorationItemAgeCutOff` を含む探索設定ハイパーパラメータ (Amazon Personalize がアイテムを推奨するときに使用するアイテム探索の量を設定するために使用するもの) を指定します。ソリューションが User-Personalization レシピを使用している場合にのみ `itemExplorationConfig` データを提供します。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

syncWithLatestSolutionVersion

キャンペーンがソリューションの最新バージョン (トレーニング済みモデル) を使用するように自動的に更新されるかどうか。指定する場合は`True`、*`SolutionVersionArn`*パラメータにソ

リューションの ARN を指定する必要があります。SolutionArn/$LATEST形式にする必要があります。Falseデフォルトはで、最新のソリューションバージョンをデプロイするにはキャンペーンを手動で更新する必要があります。

キャンペーンの自動更新について詳しくは、「キャンペーンの自動更新の有効化」を参照してください。

型: ブール値

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

CampaignSummary

サービス: Amazon Personalize

キャンペーンのプロパティの概要を提供します。完全なリストについては、DescribeCampaignAPI を呼び出してください。

コンテンツ

campaignArn

キャンペーンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

creationDateTime

キャンペーンの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

failureReason

キャンペーンが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

キャンペーンの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

キャンペーンの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

status

キャンペーンのステータス。

キャンペーンは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

CampaignUpdateSummary

サービス: Amazon Personalize

キャンペーンの更新のプロパティの概要を提供します。完全なリストについては、DescribeCampaignAPI を呼び出してください。

コンテンツ

campaignConfig

キャンペーンの設定の詳細。

タイプ：CampaignConfig オブジェクト

必須: いいえ

creationDateTime

キャンペーンの更新の作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

failureReason

キャンペーンの更新が失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

キャンペーンの更新の最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

minProvisionedTPS

Amazon Personalize がサポートする、リクエストされた 1 秒あたりの最小プロビジョンドトランザクション (推奨) を指定します。

タイプ：整数

有効な範囲: 最小値 は 1 です。

必須: いいえ

solutionVersionArn

デプロイされたソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

キャンペーン更新のステータス。

キャンペーンの更新は、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

CategoricalHyperParameterRange

サービス: Amazon Personalize

カテゴリカルハイパーパラメータの名前と範囲を指定します。

コンテンツ

name

ハイパーパラメータの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

values

ハイパーパラメータのカテゴリのリスト。

タイプ：文字列の配列

配列メンバー: 最大数は 100 項目です。

長さの制限: 最大長は 1,000 です。

必須: いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

ContinuousHyperParameterRange

サービス: Amazon Personalize

連続ハイパーパラメータの名前と範囲を指定します。

コンテンツ

maxValue

ハイパーパラメータの最大許容値。

型: 倍精度

値の範囲: 最小値 は -1000000 です。

必須: いいえ

minValue

ハイパーパラメータの最小許容値。

型: 倍精度

値の範囲: 最小値 は -1000000 です。

必須: いいえ

name

ハイパーパラメータの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

DataDeletionJob

サービス: Amazon Personalize

Amazon Personalize データセットグループから特定のユーザーへのすべての参照をバッチで削除するジョブについて説明します。データ削除ジョブの作成については、「ユーザーの削除」を参照してください。

内容

creationDateTime

データ削除ジョブの作成日時 (Unix 時間) 。

型: タイムスタンプ

必須: いいえ

dataDeletionJobArn

データ削除ジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetGroupArn

ジョブがレコードを削除するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

dataSource

データセットにアップロードするデータを含むデータソース、または Amazon Personalize から削除するレコードのリストについて説明します。

タイプ：DataSource オブジェクト

必須: いいえ

failureReason

データ削除ジョブが失敗した場合、その理由を提供します。

タイプ: 文字列

必須: いいえ

jobName

データ削除ジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

lastUpdatedDateTime

データ削除ジョブが最後に更新された日時 (Unix 時間) 。

型: タイムスタンプ

必須: いいえ

numDeleted

完了ジョブによって削除されたレコードの数。

タイプ: 整数

必須: いいえ

roleArn

Amazon S3 データソースから読み取るアクセス許可を持つ IAM ロールの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須: いいえ

status

データ削除ジョブのステータス。

データ削除ジョブには、次のいずれかのステータスがあります。

- 保留中 > IN_PROGRESS > COMPLETED - or- FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DataDeletionJobSummary

サービス: Amazon Personalize

データ削除ジョブのプロパティの概要を提供します。詳細なリストについては、DescribeDataDeletionJob API オペレーションを呼び出します。

内容

creationDateTime

データ削除ジョブの作成日時 (Unix 時間) 。

型: タイムスタンプ

必須: いいえ

dataDeletionJobArn

データ削除ジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetGroupArn

ジョブがレコードを削除したデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

データ削除ジョブが失敗した場合、その理由を提供します。

タイプ: 文字列

必須: いいえ

jobName

データ削除ジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

lastUpdatedDateTime

データ削除ジョブが最後に更新された日時 (Unix 時間) 。

型: タイムスタンプ

必須: いいえ

status

データ削除ジョブのステータス。

データ削除ジョブには、次のいずれかのステータスがあります。

- 保留中 > IN_PROGRESS > COMPLETED - or- FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Dataset

サービス: Amazon Personalize

データセットのメタデータを提供します。

コンテンツ

creationDateTime

データセットの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetArn

メタデータの取得対象とするデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetGroupArn

データセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetType

次のいずれかの値になります。

- インタラクション
- 項目

- [ユーザー]
- アクション
- Action_Interactions

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

lastUpdatedDateTime

データセットがいつ更新されたかを示すタイムスタンプ。

型: タイムスタンプ

必須: いいえ

latestDatasetUpdate

データセットの最新の更新について説明します。

タイプ：DatasetUpdateSummary オブジェクト

必須: いいえ

name

データセットの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

schemaArn

関連付けられたスキーマの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

データセットのステータス。

データセットは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

trackingId

アクションインタラクションデータセットのイベントトラッカーの ID。トラッカーの ID は `PutActionInteractions` API オペレーションで指定します。Amazon Personalize は、これを使用して、データセットグループのアクションインタラクションデータセットに新しいデータを送信します。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetExportJob

サービス: Amazon Personalize

データセットを Amazon S3 バケットにエクスポートするジョブを記述します。詳細については、を参照してくださいCreateDatasetExportJob。

データセットのエクスポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

コンテンツ

creationDateTime

データセットのエクスポートジョブの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetArn

エクスポートするデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetExportJobArn

データセットのエクスポートジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

データセットのエクスポートジョブが失敗した場合、その理由を提供します。

タイプ: 文字列

必須: いいえ

ingestionMode

エクスポートするデータ (データをインポートした方法に基づく)。BULKデータセットのインポートジョブを使用してインポートしたデータ、(コンソール、 PutUsers PutItems およびオペレーションを使用して) PUT 段階的にインポートしたデータ PutEvents、ALLまたは両方のタイプのデータをエクスポートできます。デフォルト値は PUT です。

型: 文字列

有効な値：BULK | PUT | ALL

必須：いいえ

jobName

エクスポートジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

jobOutput

ジョブの出力が保存される Amazon S3 バケットへのパス。例:

s3://bucket-name/folder-name/

タイプ：DatasetExportJobOutput オブジェクト

必須: いいえ

lastUpdatedDateTime

データセットのエクスポートジョブのステータスの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

roleArn

出力 Amazon S3 バケットにデータを追加するための許可を持つ IAM サービスロールの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

データセットのエクスポートジョブのステータス。

データセットのエクスポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetExportJobOutput

サービス: Amazon Personalize

データセットのエクスポートジョブの出力設定パラメータ。

コンテンツ

s3DataDestination

Amazon S3 の入力または出力バケットの設定の詳細。

型: S3DataConfig オブジェクト

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetExportJobSummary

サービス: Amazon Personalize

データセットのエクスポートジョブのプロパティの概要を提供します。完全なリストについては、DescribeDatasetExportJobAPI を呼び出してください。

コンテンツ

creationDateTime

データセットのエクスポートジョブの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetExportJobArn

データセットのエクスポートジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

データセットのエクスポートジョブが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

jobName

データセットのエクスポートジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

lastUpdatedDateTime

データセットのエクスポートジョブのステータスの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

status

データセットのエクスポートジョブのステータス。

データセットのエクスポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetGroup

サービス: Amazon Personalize

データセットグループは、関連するデータセットのコレクションです (アイテムインタラクション、ユーザー、アイテム、アクション、アクションインタラクション)。データセットグループを作成するには、を呼び出しますCreateDatasetGroup。次に、データセットを作成し、を呼び出してデータセットグループに追加しますCreateDataset。データセットグループは、呼び出しによるソリューションの作成とトレーニングに使用されますCreateSolution。データセットグループには、各タイプのデータセットを 1 つのみ含めることができます。

AWS Key Management Service (KMS) キーを指定してグループ内のデータセットを暗号化できます。

コンテンツ

creationDateTime

データセットグループの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

データセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

domain

ドメインデータセットグループのドメイン。

型: 文字列

有効な値：`ECOMMERCE | VIDEO_ON_DEMAND`

必須：いいえ

failureReason

データセットグループの作成が失敗した場合、その理由を提供します。

タイプ: 文字列

必須: いいえ

kmsKeyArn

データセットの暗号化に使用される (KMS) キーの Amazon リソースネーム AWS Key Management Service (ARN)。

型: 文字列

長さの制限: 最大長は 2048 です。

Pattern: `arn:aws.*:kms:.*:[0-9]{12}:key/.*`

必須: いいえ

lastUpdatedDateTime

データセットグループの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

データセットグループの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

roleArn

(KMS) キーにアクセスする権限を持つ AWS Identity and Access Management (IAM) ロールの AWS Key Management Service ARN。IAM ロールの指定は、KMS キーも指定する場合にのみ有効です。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須: いいえ

status

データセットグループの現在のステータス。

データセットグループは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetGroupSummary

サービス: Amazon Personalize

データセットグループのプロパティの概要を提供します。完全なリストについては、DescribeDatasetGroupAPI を呼び出してください。

コンテンツ

creationDateTime

データセットグループの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

データセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

domain

ドメインデータセットグループのドメイン。

型: 文字列

有効な値：`ECOMMERCE | VIDEO_ON_DEMAND`

必須：いいえ

failureReason

データセットグループの作成が失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

データセットグループが最後に更新された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

データセットグループの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

status

データセットグループのステータス。

データセットグループは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetImportJob

サービス: Amazon Personalize

トレーニングデータをデータソース (Amazon S3バケット) から Amazon Personalize データセットにインポートするジョブを記述します。詳細については、を参照してくださいCreateDatasetImportJob。

データセットのインポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

コンテンツ

creationDateTime

データセットのインポートジョブの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetArn

インポートされたデータを受け取るデータセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetImportJobArn

データセットのインポートジョブの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

dataSource

インポートするトレーニングデータを含む Amazon S3 バケット。

タイプ：DataSource オブジェクト

必須: いいえ

failureReason

データセットのインポートジョブが失敗した場合、その理由を提供します。

タイプ: 文字列

必須: いいえ

importMode

データセットのインポートジョブが新しいレコードをインポートするために使用するインポートモード。

型: 文字列

有効な値：FULL | INCREMENTAL

必須：いいえ

jobName

インポートジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

lastUpdatedDateTime

データセットが最後に更新された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

publishAttributionMetricsToS3

ジョブがメトリクス属性のために Amazon S3 にメトリクスを公開するかどうか。

型: ブール値

必須: いいえ

roleArn

Amazon S3 データソースから読み取るための許可を持つ IAM ロールの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

データセットのインポートジョブのステータス。

データセットのインポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetImportJobSummary

サービス: Amazon Personalize

データセットのインポートジョブのプロパティの概要を提供します。完全なリストについては、DescribeDatasetImportJobAPI を呼び出してください。

コンテンツ

creationDateTime

データセットのインポートジョブの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetImportJobArn

データセットのインポートジョブの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

データセットのインポートジョブが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

importMode

データセットのインポートジョブがデータセット内のデータを更新するために使用するインポートモード。詳細については、「既存のバルクデータの更新」を参照してください。

型: 文字列

有効な値：`FULL | INCREMENTAL`

必須：いいえ

jobName

データセットのインポートジョブの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

lastUpdatedDateTime

データセットのインポートジョブのステータスの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

status

データセットのインポートジョブのステータス。

データセットのインポートジョブは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetSchema

サービス: Amazon Personalize

データセットのスキーマを記述します。スキーマの詳細については、を参照してくださいCreateSchema。

コンテンツ

creationDateTime

スキーマが作成された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

domain

ドメインデータセットグループ内のデータセット用に作成したスキーマのドメイン。

型: 文字列

有効な値：`ECOMMERCE | VIDEO_ON_DEMAND`

必須：いいえ

lastUpdatedDateTime

スキーマが最後に更新された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

スキーマの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

schema

スキーマ。

型: 文字列

長さの制限: 最大長は 20,000 です。

必須: いいえ

schemaArn

スキーマの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetSchemaSummary

サービス: Amazon Personalize

データセットスキーマのプロパティの概要を提供します。完全なリストについては、DescribeSchemaAPI を呼び出してください。

コンテンツ

creationDateTime

スキーマが作成された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

domain

ドメインデータセットグループ内のデータセット用に作成したスキーマのドメイン。

型: 文字列

有効な値：`ECOMMERCE | VIDEO_ON_DEMAND`

必須：いいえ

lastUpdatedDateTime

スキーマが最後に更新された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

スキーマの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

schemaArn

スキーマの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetSummary

サービス: Amazon Personalize

データセットのプロパティの概要を提供します。完全なリストについては、DescribeDatasetAPI を呼び出してください。

コンテンツ

creationDateTime

データセットの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetArn

データセットの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

datasetType

データセットのタイプ。次のいずれかの値になります。

- インタラクション
- 項目
- [ユーザー]
- イベントインタラクション

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

lastUpdatedDateTime

データセットが最後に更新された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

データセットの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

status

データセットのステータス。

データセットは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DatasetUpdateSummary

サービス: Amazon Personalize

データセットの更新について説明します。

コンテンツ

creationDateTime

データセット更新の作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

failureReason

データセットの更新が失敗した場合、その理由を提供します。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

データセットの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

schemaArn

データセットの以前のスキーマを置き換えたスキーマの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

データセット更新のステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DataSource

サービス: Amazon Personalize

データセットにアップロードするデータを含むデータソース、または Amazon Personalize から削除するレコードのリストについて説明します。

内容

dataLocation

データセットのインポートジョブの場合、データセットにアップロードするデータが保存される Amazon S3 バケットへのパス。データ削除ジョブの場合、削除するレコードのリストを保存する Amazon S3 バケットへのパス。

例:

`s3://bucket-name/folder-name/fileName.csv`

CSV ファイルが Amazon S3 バケット内のフォルダにあり、インポートジョブまたはデータ削除ジョブで複数のファイルを考慮する場合は、フォルダへのパスを指定できます。データ削除ジョブでは、Amazon Personalize は フォルダとサブフォルダ内のすべてのファイルを使用します。フォルダ名の後に / を付けて次の構文を使用します。

`s3://bucket-name/folder-name/`

型: 文字列

長さの制限：最大長は 256 です。

パターン: `(s3|http|https)://.+`

必須: いいえ

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DefaultCategoricalHyperParameterRange

サービス: Amazon Personalize

カテゴリ別ハイパーパラメータの名前とデフォルトの範囲、およびハイパーパラメータがチューニング可能かどうかを示します。チューニング可能なハイパーパラメータは、ハイパーパラメータ最適化 (HPO) 中にその値を決定できます。

コンテンツ

isTunable

ハイパーパラメータがチューニング可能かどうか。

型: ブール値

必須: いいえ

name

ハイパーパラメータの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

values

ハイパーパラメータのカテゴリのリスト。

タイプ：文字列の配列

配列メンバー: 最大数は 100 項目です。

長さの制限: 最大長は 1,000 です。

必須: いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++

- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DefaultContinuousHyperParameterRange

サービス: Amazon Personalize

連続ハイパーパラメータの名前とデフォルトの範囲、およびハイパーパラメータがチューニング可能かどうかを示します。チューニング可能なハイパーパラメータは、ハイパーパラメータ最適化 (HPO) 中にその値を決定できます。

コンテンツ

isTunable

ハイパーパラメータがチューニング可能かどうか。

型: ブール値

必須: いいえ

maxValue

ハイパーパラメータの最大許容値。

型: 倍精度

値の範囲: 最小値 は -1000000 です。

必須: いいえ

minValue

ハイパーパラメータの最小許容値。

型: 倍精度

値の範囲: 最小値 は -1000000 です。

必須: いいえ

name

ハイパーパラメータの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

DefaultHyperParameterRanges

サービス: Amazon Personalize

ハイパーパラメータとそのデフォルトの範囲を指定します。ハイパーパラメータのタイプは、カテゴリ別、継続的、または整数値のいずれかです。

コンテンツ

categoricalHyperParameterRanges

カテゴリハイパーパラメータとそのデフォルトの範囲。

型: DefaultCategoricalHyperParameterRange オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

必須: いいえ

continuousHyperParameterRanges

連続ハイパーパラメータとそのデフォルトの範囲。

型: DefaultContinuousHyperParameterRange オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

必須: いいえ

integerHyperParameterRanges

整数値のハイパーパラメータとそのデフォルトの範囲。

型: DefaultIntegerHyperParameterRange オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

DefaultIntegerHyperParameterRange

サービス: Amazon Personalize

整数値のハイパーパラメータの名前とデフォルトの範囲、およびハイパーパラメータがチューニング可能かどうかを示します。チューニング可能なハイパーパラメータは、ハイパーパラメータ最適化 (HPO) 中にその値を決定できます。

コンテンツ

isTunable

ハイパーパラメータがチューニング可能かどうかを示します。

型: ブール値

必須: いいえ

maxValue

ハイパーパラメータの最大許容値。

タイプ: 整数

有効な範囲: 最大値は 1000000 です。

必須: いいえ

minValue

ハイパーパラメータの最小許容値。

タイプ: 整数

値の範囲: 最小値 は -1000000 です。

必須: いいえ

name

ハイパーパラメータの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

EventTracker

サービス: Amazon Personalize

イベントトラッカーに関する情報を提供します。

コンテンツ

accountId

AWS イベントトラッカーを所有するアカウント。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

creationDateTime

イベントトラッカーの作成日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

イベントデータを受け取るデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

eventTrackerArn

イベントトラッカーの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

lastUpdatedDateTime

イベントトラッカーの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

イベントトラッカーの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

status

イベントトラッカーのステータス。

イベントトラッカーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

trackingId

イベントトラッカーの ID。この ID を PutEventsAPI へのリクエストに含めてください。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

EventTrackerSummary

サービス: Amazon Personalize

イベントトラッカーのプロパティの概要を提供します。完全なリストについては、DescribeEventTrackerAPI を呼び出してください。

コンテンツ

creationDateTime

イベントトラッカーの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

eventTrackerArn

イベントトラッカーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

lastUpdatedDateTime

イベントトラッカーの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

イベントトラッカーの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

status

イベントトラッカーのステータス。

イベントトラッカーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

FeatureTransformation

サービス: Amazon Personalize

特徴変換に関する情報を提供します。機能変換は、raw 入力データを変更してモデルトレーニングにより適した形式にするプロセスです。

コンテンツ

creationDateTime

特徴変換の作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

defaultParameters

特徴変換のデフォルトパラメータを提供します。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

featureTransformationArn

FeatureTransformation オブジェクトの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

lastUpdatedDateTime

特徴変換の最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

特徴変換の名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

status

特徴変換のステータス。

特徴変換は、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

FieldsForThemeGeneration

サービス: Amazon Personalize

テーマ生成の設定の詳細の文字列から文字列へのマップ。

コンテンツ

itemName

データセット内の各アイテムの名前を保存するアイテムデータセット列の名前。

型: 文字列

長さの制限: 最大長は 1,024 です。

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Filter

サービス: Amazon Personalize

ARN、ステータス、フィルター式など、レコメンデーションフィルターに関する情報が含まれます。

コンテンツ

creationDateTime

フィルターが作成された時刻。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

フィルターが属するデータセットグループの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

フィルターが失敗した場合、その失敗の理由。

タイプ: 文字列

必須: いいえ

filterArn

フィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

filterExpression

レコメンデーションの結果から除外するアイテムのインタラクションのタイプを指定します。フィルター式は、特定のフォーマットルールに従う必要があります。フィルター式の構造と構文については、「フィルター式」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 2500 です。

必須: いいえ

lastUpdatedDateTime

フィルターが最後に更新された時刻。

型: タイムスタンプ

必須: いいえ

name

フィルターの名前。

タイプ: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

status

フィルターのステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

FilterSummary

サービス: Amazon Personalize

フィルターの属性の簡単な概要。

コンテンツ

creationDateTime

フィルターが作成された時刻。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

フィルターが属するデータセットグループの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

フィルターが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

filterArn

フィルターの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

lastUpdatedDateTime

フィルターが最後に更新された時刻。

型: タイムスタンプ

必須: いいえ

name

フィルターの名前。

タイプ: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

status

フィルターのステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

HPOConfig

サービス: Amazon Personalize

ハイパーパラメータ最適化 (HPO) のプロパティを記述します。

コンテンツ

algorithmHyperParameterRanges

ハイパーパラメータとその許容範囲。

タイプ：HyperParameterRanges オブジェクト

必須: いいえ

hpoObjective

HPO 中に最適化するメトリクス。

> **Note**
> 現時点では、Amazon Personalize は `hpoObjective` の設定をサポートしていません。

タイプ：HPOObjective オブジェクト

必須: いいえ

hpoResourceConfig

HPO のリソース設定を記述します。

タイプ：HPOResourceConfig オブジェクト

必須: いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

HPOObjective

サービス: Amazon Personalize

ハイパーパラメータ最適化 (HPO) 中に最適化するメトリクス。

> **Note**
>
> 現時点では、Amazon Personalize は `hpoObjective` の設定をサポートしていません。

コンテンツ

metricName

メトリクスの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

metricRegex

トレーニングジョブのログでメトリクスを見つけるための正規表現。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

type

メトリクスのタイプ。有効な値は、`Maximize` および `Minimize` です。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

HPOResourceConfig

サービス: Amazon Personalize

ハイパーパラメータ最適化 (HPO) のリソース設定を記述します。

コンテンツ

maxNumberOfTrainingJobs

ソリューションバージョンを作成するときのトレーニングジョブの最大数。`maxNumberOfTrainingJobs` の最大値は `40` です。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

maxParallelTrainingJobs

ソリューションバージョンを作成するときの並列トレーニングジョブの最大数。`maxParallelTrainingJobs` の最大値は `10` です。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

HyperParameterRanges

サービス: Amazon Personalize

ハイパーパラメータとその範囲を指定します。ハイパーパラメータのタイプは、カテゴリ別、継続的、または整数値のいずれかです。

コンテンツ

categoricalHyperParameterRanges

カテゴリハイパーパラメータとその範囲。

型: CategoricalHyperParameterRange オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

必須: いいえ

continuousHyperParameterRanges

連続ハイパーパラメータとその範囲。

型: ContinuousHyperParameterRange オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

必須: いいえ

integerHyperParameterRanges

整数値のハイパーパラメータとその範囲。

型: IntegerHyperParameterRange オブジェクトの配列

配列メンバー: 最大数は 100 項目です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

IntegerHyperParameterRange

サービス: Amazon Personalize

整数値のハイパーパラメータの名前と範囲を指定します。

コンテンツ

maxValue

ハイパーパラメータの最大許容値。

タイプ: 整数

有効な範囲: 最大値は 1000000 です。

必須: いいえ

minValue

ハイパーパラメータの最小許容値。

タイプ: 整数

値の範囲: 最小値 は -1000000 です。

必須: いいえ

name

ハイパーパラメータの名前。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

MetricAttribute

サービス: Amazon Personalize

メトリクス属性のレポート対象となるメトリクスに関する情報が含まれます。詳細については、「レコメンデーションの影響の測定」を参照してください。

コンテンツ

eventType

メトリクスのイベントタイプ。

型: 文字列

長さの制限：最大長は 256 です。

必須: はい

expression

属性の式。使用できるオプションは、SUM( ) または SAMPLECOUNT( ) です。SUM() 関数では、データセットのタイプ (インタラクションまたはアイテム) と合計する列をパラメーターとして指定します。例えば、SUM(Items.PRICE) などです。

型: 文字列

長さの制限：最大長は 256 です。

必須: はい

metricName

メトリクスの名前。この名前は、Amazon CloudWatch または Amazon S3 のメトリックスを識別するのに役立ちます。

型: 文字列

長さの制限：最大長は 256 です。

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

MetricAttribution

サービス: Amazon Personalize

メトリクス属性に関する情報が含まれます。メトリクス属性は、Amazon Personalize にインポートしたデータに関するレポートを作成します。データのインポート方法に応じて、Amazon CloudWatch または Amazon S3 でレポートを表示できます。詳細については、「レコメンデーションの影響の測定」を参照してください。

コンテンツ

creationDateTime

メトリクス属性の作成日時。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

メトリクス属性のデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

メトリクス属性の失敗理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

メトリクス属性の最終更新日時。

型: タイムスタンプ

必須: いいえ

metricAttributionArn

メトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

metricsOutputConfig

メトリクス属性の出力設定。

タイプ：MetricAttributionOutput オブジェクト

必須: いいえ

name

メトリクス属性の名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

status

メトリクス属性のステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

MetricAttributionOutput

サービス: Amazon Personalize

メトリクス属性の出力設定の詳細。

コンテンツ

roleArn

出力 Amazon S3 バケットにデータを追加し、Amazon にメトリクスを追加する権限を持つ IAM サービスロールの Amazon リソースネーム (ARN)。 CloudWatch詳細については、「レコメンデーションの影響の測定」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須：はい

s3DataDestination

Amazon S3 の入力または出力バケットの設定の詳細。

タイプ：S3DataConfig オブジェクト

必須: いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

MetricAttributionSummary

サービス: Amazon Personalize

メトリクス属性のプロパティの概要を提供します。完全なリストについては、に電話してくださいDescribeMetricAttribution。

コンテンツ

creationDateTime

メトリクス属性の作成日時。

型: タイムスタンプ

必須: いいえ

failureReason

メトリクス属性の失敗理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

メトリクス属性の最終更新日時。

型: タイムスタンプ

必須: いいえ

metricAttributionArn

メトリクス属性の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

name

メトリクス属性の名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

status

メトリクス属性のステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

OptimizationObjective

サービス: Amazon Personalize

ストリーミング時間 (分) の最大化や収益の増加など、ソリューションの追加の目的を記述します。詳細については、「ソリューションの最適化」を参照してください。

コンテンツ

itemAttribute

最適化の目標に関連する Items データセットの数値メタデータ列。例えば、VIDEO_LENGTH (ストリーミング (分) を最大化するため) または PRICE (収益を最大化するため) です。

型: 文字列

長さの制限：最小長は 1 です。最大長は 150 です。

必須: いいえ

objectiveSensitivity

Amazon Personalize が最適化の目的の重要性と関連性のバランスをどのように取るかを指定します。

型: 文字列

有効な値：`LOW | MEDIUM | HIGH | OFF`

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Recipe

サービス: Amazon Personalize

レシピに関する情報を提供します。各レシピには、CreateSolutionオペレーションを使用する際に Amazon Personalize がモデルトレーニングで使用するアルゴリズムが用意されています。

コンテンツ

algorithmArn

Amazon Personalize がモデルをトレーニングするために使用するアルゴリズムの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

creationDateTime

レシピの作成日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

description

recipe の説明。

タイプ: 文字列

必須: いいえ

featureTransformationArn

FeatureTransformation オブジェクトの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

lastUpdatedDateTime

レシピの最終更新日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

name

レシピの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

recipeArn

レシピの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

recipeType

次のいずれかの値になります。

- PERSONALIZED_RANKING
- RELATED_ITEMS
- USER_PERSONALIZATION

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

status

レシピのステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

RecipeSummary

サービス: Amazon Personalize

レシピのプロパティの概要を提供します。完全なリストについては、DescribeRecipeAPI を呼び出してください。

コンテンツ

creationDateTime

レシピの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

domain

レシピのドメイン (レシピがドメインデータセットグループのユースケースの場合)。

型: 文字列

有効な値：`ECOMMERCE | VIDEO_ON_DEMAND`

必須：いいえ

lastUpdatedDateTime

レシピが最後に更新された日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

レシピの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

recipeArn

レシピの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

レシピのステータス。

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Recommender

サービス: Amazon Personalize

ドメインデータセットグループのレコメンデーションジェネレーターを記述します。特定のドメインユースケース (ドメインレシピ) のドメインデータセットグループにレコメンダーを作成し、リクエストでレコメンダーを指定します。GetRecommendations

コンテンツ

creationDateTime

レコメンダーの作成日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

レコメンダーを含むドメインデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

failureReason

レコメンダーが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

レコメンダーの最終更新日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

latestRecommenderUpdate

レコメンダーに対する最新の更新の概要を提供します。

タイプ：RecommenderUpdateSummary オブジェクト

必須: いいえ

modelMetrics

レコメンダーのパフォーマンスを判断するのに役立つ評価メトリクスを提供します。詳細については、「レコメンダーの評価」を参照してください。

タイプ: ダブルマップへの文字列。

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

必須: いいえ

name

レコメンダーの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

recipeArn

レコメンダーが作成されたレシピ (ドメインデータセットグループのユースケース) の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

recommenderArn

レコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

recommenderConfig

レコメンダーの設定の詳細。

タイプ：RecommenderConfig オブジェクト

必須: いいえ

status

レコメンダーのステータス。

レコメンダーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

RecommenderConfig

サービス: Amazon Personalize

レコメンダーの設定の詳細。

コンテンツ

enableMetadataWithRecommendations

レコメンデーション付きのメタデータがレコメンダーに対して有効になっているかどうか。有効にすると、レコメンデーションのリクエストでアイテムデータセットの列を指定できます。Amazon Personalize は、レコメンデーションレスポンス内の各アイテムについてこのデータを返します。レコメンダーのメタデータを有効にする方法については、「Enabling metadata in recommendations for a recommender」を参照してください。

レコメンデーションのメタデータを有効にすると、追加費用が発生します。詳細については、「Amazon Personalize の料金」を参照してください。

型: ブール値

必須: いいえ

itemExplorationConfig

`explorationWeight` と `explorationItemAgeCutOff` を含む探索設定ハイパーパラメータ (Amazon Personalize がアイテムを推奨するときに使用するアイテム探索の量を設定するために使用するもの) を指定します。レコメンダーがユーザー向けにパーソナライズされたレコメンデーションを生成する場合にのみ `itemExplorationConfig` データを提供します (人気のあるアイテムや類似アイテムではありません)。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

minRecommendationRequestsPerSecond

Amazon Personalize がサポートする、リクエストされた 1 秒あたりの最小プロビジョンドトランザクション (推奨) を指定します。`minRecommendationRequestsPerSecond` の値を高く設

定すると請求額が増加します。最初は minRecommendationRequestsPerSecond に 1 を使用することをお勧めします (デフォルト)。Amazon CloudWatch メトリックスを使用して使用状況を追跡し、minRecommendationRequestsPerSecond必要に応じて増やします。

タイプ：整数

有効な範囲: 最小値 は 1 です。

必須: いいえ

trainingDataConfig

ドメインレコメンダーを作成するときに使用するトレーニングデータ設定を指定します。

タイプ：TrainingDataConfig オブジェクト

必須: いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

RecommenderSummary

サービス: Amazon Personalize

レコメンダーのプロパティの概要を提供します。

コンテンツ

creationDateTime

レコメンダーの作成日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

レコメンダーを含むドメインデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

lastUpdatedDateTime

レコメンダーの最終更新日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

name

レコメンダーの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

recipeArn

レコメンダーが作成されたレシピ (ドメインデータセットグループのユースケース) の Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

recommenderArn

レコメンダーの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

recommenderConfig

レコメンダーの設定の詳細。

タイプ：RecommenderConfig オブジェクト

必須: いいえ

status

レコメンダーのステータス。レコメンダーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

RecommenderUpdateSummary

サービス: Amazon Personalize

レコメンダーの更新のプロパティの概要を提供します。完全なリストについては、DescribeRecommenderAPI を呼び出してください。

コンテンツ

creationDateTime

レコメンダーの更新の作成日時 (Unix 形式)。

型: タイムスタンプ

必須: いいえ

failureReason

レコメンダーの更新が失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

レコメンダーの更新の最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

recommenderConfig

レコメンダーの更新の設定の詳細。

タイプ：RecommenderConfig オブジェクト

必須: いいえ

status

レコメンダーの更新のステータス。

レコメンダーは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

S3DataConfig

サービス: Amazon Personalize

Amazon S3 の入力または出力バケットの設定の詳細。

コンテンツ

path

Amazon S3 バケットのファイルパス。

型: 文字列

長さの制限：最大長は 256 です。

パターン：`(s3|http|https)://.+`

必須：はい

kmsKeyArn

Amazon Personalize が入力ファイルと出力ファイルの暗号化または復号化に使用する AWS Key Management Service (KMS) キーのアマゾンリソースネーム (ARN)。

型: 文字列

長さの制限: 最大長は 2048 です。

Pattern: `arn:aws.*:kms:.*:[0-9]{12}:key/.*`

必須: いいえ

その他の参照資料

この API AWS を言語固有の SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Solution

サービス: Amazon Personalize

> **Important**
>
> ソリューションを作成した後は、その構成を変更することはできません。デフォルトでは、新しいソリューションはすべて自動トレーニングを使用します。自動トレーニングでは、ソリューションがアクティブである間はトレーニングコストが発生します。ソリューションの自動トレーニングを停止することはできません。不要なコストが発生しないように、終了したら必ずソリューションを削除してください。トレーニング費用の詳細については、「Amazon Personalize 料金表」を参照してください。

ソリューションに関する情報を提供するオブジェクト。ソリューションには、Amazon Personalize がレコメンデーションを生成するために使用するカスタムレシピ、カスタマイズされたパラメータ、トレーニング済みモデル (ソリューションバージョン) が含まれます。

ソリューションを作成した後は、その設定を変更することはできません。変更が必要な場合は、Amazon Personalize コンソールでソリューションを複製するか、新しいソリューションを作成できます。

コンテンツ

autoMLResult

performAutoML が true の場合、見つかった最適なレシピを指定します。

タイプ : AutoMLResult オブジェクト

必須: いいえ

creationDateTime

ソリューションの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

トレーニングデータを提供するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

eventType

モデルのトレーニングに使用されるイベントタイプ (例えば、「クリック」または「いいね」)。`eventType` が指定されていない場合、Amazon Personalize は、タイプにかかわらず、同じ重みでトレーニングするためにすべてのインタラクションを使用します。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

lastUpdatedDateTime

ソリューションの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

latestSolutionVersion

ステータスと ARN を含む、ソリューションの最新バージョンを記述します。

タイプ：SolutionVersionSummary オブジェクト

必須: いいえ

name

ソリューションの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

performAutoML

> **Important**
>
> 自動機械学習の有効化は推奨されません。代わりに、利用可能な Amazon Personalize のレシピにユースケースをマッチさせます。詳細については、「ユースケースの特定」を参照してください。

true の場合、Amazon Personalize は、ソリューション構成で指定されたリストから最適な USER_PERSONALIZATION recipe の検索を実行します (`recipeArn` を指定しないでください)。false (デフォルト) の場合、Amazon Personalize はトレーニングに `recipeArn` を使用します。

型: ブール値

必須: いいえ

performAutoTraining

ソリューションが自動的にソリューションバージョンを作成するかどうかを指定します。`True`デフォルトはで、ソリューションは 7 日ごとに新しいソリューションバージョンを自動的に作成します。

auto トレーニングの詳細については、「ソリューションの作成と構成」を参照してください。

型: ブール値

必須: いいえ

performHPO

選択したレシピでハイパーパラメータ最適化 (HPO) を実行するかどうか。デフォルトは `false` です。

型: ブール値

必須: いいえ

recipeArn

ソリューションの作成に使用されたレシピの ARN。これは、`performAutoML` が false の場合に必要です。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

solutionArn

ソリューションの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

solutionConfig

ソリューションの設定プロパティを記述します。

タイプ：SolutionConfig オブジェクト

必須: いいえ

status

ソリューションのステータス。

ソリューションは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

SolutionConfig

サービス: Amazon Personalize

ソリューションの構成プロパティを記述します。

コンテンツ

algorithmHyperParameters

アルゴリズムのハイパーパラメータとその値を一覧表示します。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

autoMLConfig

AutoML の実行時に検索する recipe のリストを含む AutoMLConfig オブジェクト。

タイプ：AutoMLConfig オブジェクト

必須: いいえ

autoTrainingConfig

使用する自動トレーニング構成を指定します。

タイプ：AutoTrainingConfig オブジェクト

必須: いいえ

eventValueThreshold

モデルのトレーニングには、このしきい値以上の値を持つイベントのみが使用されます。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

featureTransformationParameters

特徴変換パラメータを一覧表示します。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須: いいえ

hpoConfig

ハイパーパラメータ最適化 (HPO) のプロパティを記述します。

タイプ：HPOConfig オブジェクト

必須: いいえ

optimizationObjective

ストリーミング時間 (分) の最大化や収益の増加など、ソリューションの追加の目的を記述します。詳細については、「ソリューションの最適化」を参照してください。

タイプ：OptimizationObjective オブジェクト

必須: いいえ

trainingDataConfig

カスタムソリューションバージョン (トレーニング済みモデル) を作成するときに使用するトレーニングデータ構成を指定します。

タイプ：TrainingDataConfig オブジェクト

必須: いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++

- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

SolutionSummary

サービス: Amazon Personalize

ソリューションのプロパティの概要を提供します。完全なリストについては、DescribeSolutionAPI を呼び出してください。

コンテンツ

creationDateTime

ソリューションの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

lastUpdatedDateTime

ソリューションの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

ソリューションの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

recipeArn

ソリューションによって使用されるレシピの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

solutionArn

ソリューションの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

ソリューションのステータス。

ソリューションは、次に示す状態のいずれかになります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

型: 文字列

長さの制限：最大長は 256 です。

必須：いいえ

その他の参照資料

この API を言語固有の AWS SDK で使用する方法について詳しくは、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

SolutionVersion

サービス: Amazon Personalize

カスタムデータセットグループ内のソリューションの特定のバージョンに関する情報を提供するオブジェクト。

コンテンツ

creationDateTime

このバージョンのソリューションの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

datasetGroupArn

トレーニングデータを提供するデータセットグループの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

eventType

モデルのトレーニングに使用されるイベントタイプ (例えば、「クリック」または「いいね」)。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

failureReason

ソリューションバージョンのトレーニングが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

ソリューションの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

name

ソリューションバージョンの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

必須: いいえ

performAutoML

true の場合、Amazon Personalize は、ソリューション設定に従って最適なレシピを検索します。false (デフォルト) の場合、Amazon Personalize は `recipeArn` を使用します。

型: ブール値

必須: いいえ

performHPO

選択したレシピでハイパーパラメータ最適化 (HPO) を実行するかどうか。デフォルトは `false` です。

型: ブール値

必須: いいえ

recipeArn

ソリューションで使用されるレシピの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

solutionArn

ソリューションの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

solutionConfig

ソリューションの設定プロパティを記述します。

タイプ：SolutionConfig オブジェクト

必須: いいえ

solutionVersionArn

ソリューションバージョンの ARN。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

ソリューションバージョンのステータス。

ソリューションバージョンは、次のいずれかの状態になります。

- CREATE PENDING
- CREATE IN_PROGRESS
- ACTIVE

- CREATE FAILED
- CREATE STOPPING
- CREATE STOPPED

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

trainingHours

モデルのトレーニングに使用した時間。モデルのトレーニングにかかる時間について請求されます。このフィールドは、Amazon Personalize がモデルのトレーニングに成功した後にのみ表示されます。

型: 倍精度

有効な範囲：最小値 は 0 です。

必須: いいえ

trainingMode

ソリューションバージョンを作成するときに実行するトレーニングの範囲。FULLトレーニングでは、データセットグループのすべてのデータが考慮されます。UPDATEは最新のトレーニング以降に変更されたデータのみを処理します。User-Personalization レシピで作成されたソリューションバージョンだけが使用できます。UPDATE

型: 文字列

有効な値：FULL | UPDATE | AUTOTRAIN

必須：いいえ

trainingType

ソリューションバージョンが自動で作成されたか、手動で作成されたか。

型: 文字列

有効な値：AUTOMATIC | MANUAL

必須：いいえ

tunedHPOParams

ハイパーパラメータ最適化が実行された場合、最もパフォーマンスの高いモデルのハイパーパラメータ値が含まれます。

タイプ：TunedHPOParams オブジェクト

必須: いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

SolutionVersionSummary

サービス: Amazon Personalize

ソリューションバージョンのプロパティの概要を提供します。完全なリストについては、DescribeSolutionVersionAPI を呼び出してください。

コンテンツ

creationDateTime

このバージョンのソリューションの作成日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

failureReason

ソリューションバージョンが失敗した場合、失敗の理由。

タイプ: 文字列

必須: いいえ

lastUpdatedDateTime

ソリューションバージョンの最終更新日時 (Unix 時間)。

型: タイムスタンプ

必須: いいえ

solutionVersionArn

ソリューションバージョンの Amazon リソースネーム (ARN)。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

status

ソリューションバージョンのステータス。

ソリューションバージョンは、次のいずれかの状態になります。

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -または- CREATE FAILED

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

trainingMode

ソリューションバージョンを作成するときに実行するトレーニングの範囲。FULLトレーニングでは、データセットグループ内のすべてのデータが考慮されます。UPDATEは最新のトレーニング以降に変更されたデータのみを処理します。User-Personalization レシピで作成されたソリューションバージョンだけが使用できます。UPDATE

型: 文字列

有効な値：FULL | UPDATE | AUTOTRAIN

必須：いいえ

trainingType

ソリューションバージョンが自動で作成されたか、手動で作成されたか。

型: 文字列

有効な値：AUTOMATIC | MANUAL

必須：いいえ

その他の参照資料

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Tag

サービス: Amazon Personalize

リソースに適用し、リソースの分類と整理に役立つオプションのメタデータ。タグはそれぞれ、1 つのキーとオプションの 1 つの値で設定されており、どちらもお客様側が定義します。詳細については、「Amazon Personalize リソースのタグ付け」を参照してください。

コンテンツ

tagKey

タグを構成するキーと値のペアの一部。キーは、より具体的なタグ値のカテゴリのように動作する、一般的なラベルです。

型: 文字列

長さの制限: 最小長は 1 です。最大長は 128 です。

パターン: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

必須：はい

tagValue

タグを構成するキーと値のペアのオプションの一部。値はタグカテゴリ (キー) の記述子として機能します。

型: 文字列

長さの制限: 最小長は 0 です。最大長は 256 です。

パターン：`^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK の 1 つでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK

- AWS ルビー V3 用 SDK

ThemeGenerationConfig

サービス: Amazon Personalize

バッチ推論ジョブでテーマを生成するための設定の詳細。

コンテンツ

fieldsForThemeGeneration

バッチ推論ジョブの説明テーマを生成するために使用されるフィールド。

型: FieldsForThemeGeneration オブジェクト

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

TrainingDataConfig

サービス: Amazon Personalize

ドメインレコメンダーまたはカスタムソリューションバージョン (トレーニング済みモデル) を作成するときに使用するトレーニングデータ構成。

コンテンツ

excludedDatasetColumns

トレーニングから除外する列を指定します。各キーはデータセットタイプで、各値は列のリストです。列を除外することで、Amazon Personalize がレコメンデーションの生成に使用するデータを制御できます。

例えば、レコメンデーションをフィルタリングするためだけに使用したい列があるとします。この列をトレーニングから除外すると、Amazon Personalize はフィルタリング時にのみこの列を考慮します。

タイプ: 文字列マップの配列への文字列

マップエントリ: アイテムの最大数は 3 です。

キーの長さの制限: 最大長は 256 です。

配列メンバー: 最大数は 50 品目です。

長さの制限: 最大長は 1,024 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

TunedHPOParams

サービス: Amazon Personalize

ハイパーパラメータ最適化 (HPO) が実行された場合、最もパフォーマンスの高いモデルのハイパーパラメータ値が含まれます。

コンテンツ

algorithmHyperParameters

最高のパフォーマンスを発揮するモデルのハイパーパラメータ値のリスト。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 100 です。

キーの長さの制限: 最大長は 256 です。

値の長さの制限: 最大長は 1000 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Amazon Personalize Events

次のデータ型は、Amazon Personalize Events でサポートされています。

- Action
- ActionInteraction
- Event
- Item
- MetricAttribution

- User

Action

サービス: Amazon Personalize Events

`PutActions` API を使用してアクションデータセットに追加されたアクションメタデータを表します。詳細については、「Importing actions individually」を参照してください。

コンテンツ

actionId

アクションに関連付けられた ID。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

properties

アクション固有のメタデータの文字列マップ。マップの各要素は、キーバリューペアで設定されています。例えば `{"value": "100"}` です。

キーは、アクションデータセットのスキーマ内のフィールドと一致するキャメルケース名を使用します。前の例では、`value` はアクションスキーマで定義された「VALUE」フィールドと一致します。カテゴリ文字列データについて、単一のアクションに複数のカテゴリを含めるには、各カテゴリをパイプ区切り文字 (`|`) で区切ります。例えば `\"Deluxe|Premium\"` です。

タイプ: 文字列

長さの制限：最小長は 1 です。最大長は 32,000 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

ActionInteraction

サービス: Amazon Personalize Events

PutActionInteractions API を使用して送信されるアクションインタラクションイベントを表します。

コンテンツ

actionId

ユーザーが操作したアクションの ID。これはアクションインタラクションスキーマの ACTION_ID フィールドに対応します。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

eventType

アクションインタラクションイベントのタイプ。Viewed、Taken、および Not Taken イベントタイプを指定できます。アクションインタラクションのイベントタイプデータについて詳しくは、「Event type data」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

sessionId

ユーザーの訪問に関連付けられたセッション ID。アプリケーションは、ユーザーが最初にウェブサイトにアクセスしたとき、またはアプリケーションを使用したときに、一意の sessionId を生成します。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

timestamp

アクションインタラクションイベントが発生したときのタイムスタンプ。タイムスタンプは Unix エポック時刻形式 (秒単位) である必要があります。

型: タイムスタンプ

必須: はい

eventId

イベントに関連付けられた ID。イベント ID が指定されていない場合、Amazon Personalize はイベントの一意の ID を生成します。イベント ID はモデルへの入力として使用されません。Amazon Personalize は、イベント ID を使用して一意のイベントを区別します。同じイベント ID を持つ最初のイベント以降の後続のイベントは、モデルトレーニングでは使用されません。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: いいえ

impression

ユーザーに表示したアクションのシーケンスを表すアクション ID のリスト。例えば `["actionId1", "actionId2", "actionId3"]` です。Amazon Personalize は、アクションインタラクションイベントのインプレッションデータを使用しません。代わりに、アクションごとに複数のイベントを記録し、`Viewed` イベントタイプを使用してください。

タイプ：文字列の配列

配列メンバー：最小数は 1 項目です。最大数は 25 項目です。

長さの制限：最小長は 1 です。最大長は 256 です。

必須: いいえ

properties

記録することを選択する可能性のあるイベント固有のデータの文字列マップ。例えば、ユーザーがアクションを実行した場合 (アクション ID 以外)、ユーザーが行ったアクションの数を送信することもできます。

マップの各アイテムは、キーバリューペアで設定されています。例:

```
{"numberOfActions": "12"}
```

キーは、アクションインタラクションスキーマのフィールドと一致するキャメルケース名を使用します。上の例では、numberOfActions はアクションインタラクションスキーマで定義された「NUMBER_OF_ACTIONS」フィールドと一致します。

以下はプロパティのキーワードとして含めることはできません (大文字と小文字は区別されません)。

- userId
- sessionId
- eventType
- timestamp
- recommendationId
- インプレッション

型: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

必須: いいえ

recommendationId

ユーザーが操作したアイテムを含むレコメンデーションのリストの ID。

型: 文字列

長さの制限：最小長は 1 です。最大長は 40 です。

必須: いいえ

userId

アクションを操作したユーザーの ID。これはアクションインタラクションスキーマの USER_ID フィールドに対応します。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Event

サービス: Amazon Personalize Events

PutEvents API を使用して送信されるアイテムインタラクションのイベント情報を表します。

コンテンツ

eventType

イベントのタイプ (クリックやダウンロードなど)。このプロパティは、アイテムインタラクションデータセットスキーマの EVENT_TYPE フィールドに対応し、追跡しているイベントのタイプによって異なります。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

sentAt

イベントが発生したときのクライアント側のタイムスタンプ (Unix 時間)。

型: タイムスタンプ

必須: はい

eventId

イベントに関連付けられた ID。イベント ID が指定されていない場合、Amazon Personalize はイベントの一意の ID を生成します。イベント ID はモデルへの入力として使用されません。Amazon Personalize は、イベント ID を使用して一意のイベントを区別します。同じイベント ID を持つ最初のイベント以降の後続のイベントは、モデルトレーニングでは使用されません。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: いいえ

eventValue

アイテムインタラクションスキーマの EVENT_VALUE フィールドに対応するイベント値。

タイプ: 浮動小数点

必須: いいえ

impression

ユーザーに表示したアイテムのシーケンスを表すアイテム ID のリスト。例えば ["itemId1", "itemId2", "itemId3"] です。イベントのインプレッションデータを手動で記録する項目のリストを提供します。インプレッションデータの記録についての詳細は、「インプレッションデータの記録」を参照してください。

タイプ：文字列の配列

配列メンバー：最小数は 1 項目です。最大数は 25 項目です。

長さの制限：最小長は 1 です。最大長は 256 です。

必須: いいえ

itemId

アイテムインタラクションデータセットのスキーマの ITEM_ID フィールドに対応するアイテム ID キー。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: いいえ

metricAttribution

イベントに関連するメトリクス属性に関する情報が含まれています。メトリクス属性についての詳細は、「レコメンデーションの影響の測定」を参照してください。

タイプ：MetricAttribution オブジェクト

必須: いいえ

properties

記録することを選択する可能性のあるイベント固有のデータの文字列マップ。例えば、ユーザーがサイトで映画を評価した場合 (映画 ID (itemId) とレーティング (eventValue) 以外)、ユーザーが行った映画のレーティングの数を送信することもできます。

マップの各アイテムは、キーバリューペアで設定されています。例:

```
{"numberOfRatings": "12"}
```

キーは、アイテムインタラクションデータセットのスキーマのフィールドと一致するキャメルケース名を使用します。上の例では、numberOfRatings はアイテムインタラクションデータセットのスキーマで定義された「NUMBER_OF_RATINGS」フィールドと一致します。

以下はプロパティのキーワードとして含めることはできません (大文字と小文字は区別されません)。

- userId
- sessionId
- eventType
- timestamp
- recommendationId
- インプレッション

型: 文字列

長さの制限: 最小長は 1 です。最大長は 1,024 です。

必須: いいえ

recommendationId

ユーザーが操作したアイテムを含むレコメンデーションのリストの ID。Amazon Personalize がユーザーに表示したレコメンデーションをインプレッションデータとして暗黙のうちに記録するには、recommendationId を指定します。または、メトリクス属性を使用してレコメンデーションの影響を測定する場合は recommendationId を指定します。

インプレッションデータの記録についての詳細は、「インプレッションデータの記録」を参照してください。メトリクス属性の作成について詳しくは、「レコメンデーションの影響の測定」を参照してください。

型: 文字列

長さの制限：最小長は 1 です。最大長は 40 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Item

サービス: Amazon Personalize Events

PutItems API を使用して Items データセットに追加されたアイテムメタデータを表します。詳細については、「Importing items individually」を参照してください。

コンテンツ

itemId

アイテムに関連付けられた ID。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

properties

アイテム固有のメタデータの文字列マップ。マップの各要素は、キーバリューペアで設定されています。例えば {"numberOfRatings": "12"} です。

キーは、Items データセットのスキーマ内のフィールドと一致するキャメルケース名を使用します。前の例では、numberOfRatings は Items スキーマで定義された「NUMBER_OF_RATINGS」フィールドと一致します。カテゴリ文字列データについて、単一のアイテムに複数のカテゴリを含めるには、各カテゴリをパイプ区切り文字 (|) で区切ります。例えば \"Horror|Action\" です。

タイプ: 文字列

長さの制限：最小長は 1 です。最大長は 32,000 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

MetricAttribution

サービス: Amazon Personalize Events

イベントに関連するメトリクス属性に関する情報が含まれます。メトリクス属性についての詳細は、「レコメンデーションの影響の測定」を参照してください。

コンテンツ

eventAttributionSource

サードパーティーなどのイベントのソース。

型: 文字列

長さの制限：最大長は 1,024 です。

パターン：`^[\x20-\x7E]*[\x21-\x7E]+[\x20-\x7E]*$`

必須：はい

以下の資料も参照してください。

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

User

サービス: Amazon Personalize Events

PutUsers API を使用して Users データセットに追加されたユーザーメタデータを表します。詳細については、「Importing users individually」を参照してください。

コンテンツ

userId

ユーザーに関連付けられた ID。

型: 文字列

長さの制限：最小長は 1 です。最大長は 256 です。

必須: はい

properties

アイテム固有のメタデータの文字列マップ。マップの各要素は、キーバリューペアで設定されています。例えば {"numberOfVideosWatched": "45"} です。

キーは、Users データセットのスキーマ内のフィールドと一致するキャメルケース名を使用します。前の例では、numberOfVideosWatched は Users スキーマで定義された「NUMBER_OF_VIDEOS_WATCHED」フィールドに一致します。カテゴリ文字列データについて、単一のユーザーに複数のカテゴリを含めるには、各カテゴリをパイプ区切り文字 (|) で区切ります。例えば \"Member|Frequent shopper\" です。

タイプ: 文字列

長さの制限：最小長は 1 です。最大長は 24,000 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

Amazon Personalize Runtime

次のデータ型は、Amazon Personalize Runtime でサポートされています。

- PredictedAction
- PredictedItem
- Promotion

PredictedAction

サービス: Amazon Personalize Runtime

アクションを識別するオブジェクト。

GetActionRecommendations API は `PredictedAction` のリストを返します。

コンテンツ

actionId

レコメンデーションアクションの ID。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

score

レコメンデーションアクションのスコア。アクションスコアの詳細については、「How action recommendation scoring works」を参照してください。

型: 倍精度浮動小数点数

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

PredictedItem

サービス: Amazon Personalize Runtime

アイテムを識別するオブジェクト。

GetRecommendations および GetPersonalizedRanking API は、`PredictedItem` のリストを返します。

内容

itemId

推奨アイテム ID。

型: 文字列

長さの制限：最大長は 256 です。

必須: いいえ

metadata

アイテムデータセットのアイテムに関するメタデータ。

型: 文字列間のマッピング

キーの長さの制限: 最大長は 150 です。

値の長さの制限: 最大長は 20,000 です。

必須: いいえ

promotionName

予測アイテムを含むプロモーションの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

reason

User-Personalization-v2 を使用する場合、そのアイテムがレコメンデーションに含まれていた理由のリスト。考えられる理由は次のとおりです。

- 昇格済みアイテム - レコメンデーションリクエストに適用した昇格の一部としてアイテムが含まれていたことを示します。
- 探索 - 項目が探索に含まれたことを示します。探索では、レコメンデーションには、インタラクションデータやユーザーへの関連性が少ないアイテムが含まれます。探索の詳細については、「探索」を参照してください。
- 人気アイテム - アイテムがプレースホルダーの人気アイテムとして含まれたことを示します。フィルターを使用する場合、フィルターが削除するレコメンデーションの数に応じて、Amazon Personalize はレコメンデーションリクエスト`numResults`の を満たすプレースホルダー項目を追加する場合があります。これらの項目は、インタラクションデータに基づいて、フィルター条件を満たす人気のある項目です。ユーザーには関連性スコアがありません。

タイプ：文字列の配列

長さの制限：最大長は 256 です。

必須: いいえ

score

次にユーザーが該当のアイテムを選択することについての、モデルの確実性の数値表現。スコアリングロジックの詳細については、「レコメンデーションスコア」を参照してください。

型: 倍精度浮動小数点数

必須：いいえ

その他の参照資料

言語固有の AWS SDKs のいずれかでこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Promotion

サービス: Amazon Personalize Runtime

プロモーションに関する情報が含まれています。プロモーションは、設定可能な推奨アイテムのサブセットに適用される追加のビジネスルールを定義します。

コンテンツ

filterArn

プロモーションで使用される Amazon リソースネーム (ARN)。このフィルターはプロモーション対象アイテムの条件を定義します。フィルターを使用する方法については、「プロモーションフィルター」を参照してください。

型: 文字列

長さの制限：最大長は 256 です。

パターン: `arn:([a-z\d-]+):personalize:.*:.*:.+`

必須: いいえ

filterValues

アイテムをプロモーションするときに使用する値。フィルター式の各プレースホルダーパラメータについて、(大文字と小文字が一致した状態で) パラメータ名をキーとして指定し、フィルター値を対応する値として指定します。1 つのパラメータの複数の値をコンマで区切ります。

`INCLUDE` 要素を使用してアイテムを含めるフィルター式について、式で定義されているすべてのパラメータの値を指定する必要があります。`EXCLUDE` 要素を使用してアイテムを除外する式を含むフィルターについては、`filter-values` を省略できます。この場合、Amazon Personalize は、式のその部分を使用してレコメンデーションをフィルタリングしません。

フィルター作成についての詳細は、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。

型: 文字列間のマッピング

マップエントリ: アイテムの最大数は 25 です。

キーの長さの制限: 最大長は 50 です。

キーパターン: `[A-Za-z0-9_]+`

値の長さの制限: 最大長は 1000 です。

必須: いいえ

name

プロモーションの名前。

型: 文字列

長さの制限：最小長は 1 です。最大長は 63 です。

パターン: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

必須: いいえ

percentPromotedItems

プロモーションを適用する推奨アイテム品の割合。

タイプ：整数

有効範囲: 最小値 は 1 です。最大値は 100 です。

必須：いいえ

その他の参照資料

言語固有の AWS SDK でこの API を使用する方法の詳細については、以下を参照してください。

- AWS SDK for C++
- AWS Java V2 用 SDK
- AWS ルビー V3 用 SDK

共通エラー

このセクションでは、AWS のすべてのサービスの API アクションに共通のエラーを一覧表示しています。このサービスの API アクションに固有のエラーについては、その API アクションのトピックを参照してください。

AccessDeniedException

このアクションを実行する十分なアクセス権限がありません。

HTTP ステータスコード: 400

IncompleteSignature

リクエストの署名が AWS 基準に適合しません。

HTTP ステータスコード: 400

InternalFailure

リクエストの処理が、不明なエラー、例外、または障害により実行できませんでした。

HTTP ステータスコード: 500

InvalidAction

リクエストされたアクション、またはオペレーションは無効です。アクションが正しく入力されていることを確認します。

HTTP ステータスコード: 400

InvalidClientTokenId

指定された x.509 証明書、または AWS アクセスキー ID が見つかりません。

HTTP ステータスコード: 403

NotAuthorized

このアクションを実行するにはアクセス許可が必要です。

HTTP ステータスコード: 400

OptInRequired

サービスを利用するためには、AWS アクセスキー ID を取得する必要があります。

HTTP ステータスコード: 403

RequestExpired

リクエストの日付スタンプの 15 分以上後またはリクエストの有効期限 (署名付き URL の場合など) の 15 分以上後に、リクエストが到着しました。または、リクエストの日付スタンプが現在より 15 分以上先です。

HTTP ステータスコード: 400

ServiceUnavailable

リクエストは、サーバーの一時的障害のために実行に失敗しました。

HTTP ステータスコード: 503

ThrottlingException

リクエストは、制限が必要なために実行が拒否されました。

HTTP ステータスコード: 400

ValidationError

入力が、AWS サービスで指定された制約を満たしていません。

HTTP ステータスコード: 400

共通パラメータ

次のリストには、すべてのアクションが署名バージョン 4 リクエストにクエリ文字列で署名するために使用するパラメータを示します。アクション固有のパラメータは、アクションのトピックに示されています。Signature Version 4 の詳細については、「IAM ユーザーガイド」の「AWS API リクエストの署名」を参照してください。

Action

実行するアクション。

型: 文字列

必須: はい

Version

リクエストが想定している API バージョンである、YYYY-MM-DD 形式で表示されます。

型: 文字列

必須: はい

X-Amz-Algorithm

リクエストの署名を作成するのに使用したハッシュアルゴリズム。

条件: HTTP 認証ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

有効な値: AWS4-HMAC-SHA256

必須: 条件による

X-Amz-Credential

認証情報スコープの値で、アクセスキー、日付、対象とするリージョン、リクエストしているサービス、および終了文字列 ("aws4_request") を含む文字列です。値は次の形式で表現されます。[access_key/YYYYYYYYMMDD/リージョン/サービス/aws4_request]

詳細については、「IAM ユーザーガイド」の「署名付きAWS API リクエストの作成」を参照してください。

条件: HTTP 認証ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

X-Amz-Date

署名を作成するときに使用する日付です。形式は ISO 8601 基本形式の YYYYMMDD'T'HHMMSS'Z' でなければなりません。例えば、日付 20120325T120000Z は、有効な X-Amz-Date の値です。

条件: X-Amz-Date はすべてのリクエストに対してオプションです。署名リクエストで使用する日付よりも優先される日付として使用できます。ISO 8601 ベーシック形式で日付ヘッダーが指定されている場合、X-Amz-Date は必要ありません。X-Amz-Date を使用すると、常に Date ヘッダーの値よりも優先されます。詳細については、「IAM ユーザーガイド」の「AWS API リクエスト署名の要素」を参照してください。

タイプ: 文字列

必須: 条件による

X-Amz-Security-Token

AWS Security Token Service (AWS STS) へのを呼び出しで取得された一時的なセキュリティトークン。AWS STS の一時的なセキュリティ認証情報をサポートするサービスのリストについては、「IAM ユーザーガイド」の「IAM と連携するAWS のサービス」を参照してください。

条件:AWS STS の一時的なセキュリティ認証情報を使用する場合、セキュリティトークンを含める必要があります。

タイプ: 文字列

必須: 条件による

X-Amz-Signature

署名する文字列と派生署名キーから計算された 16 進符号化署名を指定します。

条件: HTTP 認証ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

X-Amz-SignedHeaders

正規リクエストの一部として含まれていたすべての HTTP ヘッダーを指定します。署名付きヘッダーの指定に関する詳細については、「IAM ユーザーガイド」の「署名付き AWS API リクエストの作成」を参照してください。

条件: HTTP 認証ヘッダーではなくクエリ文字列に認証情報を含める場合は、このパラメータを指定します。

型: 文字列

必須: 条件による

Amazon Personalize のドキュメント履歴

次の表は、Amazon Personalize デベロッパーガイドの各リリースにおける重要な変更点を示しています。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
新機能	Amazon Personalize では、メタデータやインタラクションデータを含むユーザーとそのデータをデータセットグループから削除できるようになりました。詳細については、「ユーザーの削除」を参照してください。	2024 年 5 月 2 日
新機能	Amazon Personalize では、トレーニングの迅速化、レコメンデーションのレイテンシーの短縮、およびより関連性の高いレコメンデーションにより、最大 500 万件のアイテムを検討できるようになりました。この拡張トレーニングには、User-Personalization-v2と Personalized-Ranking-v2つの新しいカスタムレシピが使用されます。	2024 年 5 月 1 日
新機能	すべての新しい Amazon Personalize カスタムソリューションで、自動トレーニングを使用するようになりました。自動トレーニングの設定の詳細については、「自動ト	2024 年 4 月 19 日

	レーニングの設定」を参照してください。	
新機能	Amazon Personalize は、ユーザーの行動に基づいて次善のアクションをユーザーに推奨する機能をサポートするようになりました。詳細については、「Next-Best-Action レシピ」を参照してください。	2023 年 11 月 26 日
新機能	Amazon Personalize は、生成 AI の助けを借りて、説明的なテーマを一括レコメンデーションに含める機能をサポートするようになりました。テーマを持つ一括レコメンデーションの生成について詳しくは、「テーマを持つ一括レコメンデーション」を参照してください。Amazon Personalize を生成 AI と組み合わせて使用する方法については、「Amazon Personalize and generative AI」を参照してください。	2023 年 11 月 26 日

新機能	Amazon Personalize は、Items データセットのメタデータをレコメンデーションに含める機能をサポートするようになりました。キャンペーンのメタデータを有効にする方法については、「レコメンデーション」の「アイテムメタデータ」を参照してください。レコメンダーのメタデータを有効にする方法については、「Enabling metadata in recommendations for a recommender」を参照してください。	2023 年 11 月 26 日
新機能	Amazon Personalize では、OpenSearch サービスから検索結果をパーソナライズできるようになりました。詳細については、「 からの検索結果のパーソナライズ OpenSearch」を参照してください。	2023 年 10 月 16 日
新機能	Amazon Personalize では、アイテムデータセットでは最大 100 のメタデータ列 (50 から増加)、ユーザーデータセットでは最大 25 のメタデータ列 (5 つから増加) をインポートできるようになりました。Amazon Personalize の制限の詳細については、「Amazon Personalize エンドポイントとクォータ」を参照してください。	2023 年 9 月 5 日

新機能	Amazon Personalize は、 から検索結果をパーソナライズする機能をサポートするようになりました OpenSearch。詳細については、OpenSearch「 (セルフマネージド) から検索結果をパーソナライズする」を参照してください。	2023 年 7 月 25 日
新機能	Amazon Personalize は、データセットのスキーマを新しいスキーマまたは既存のスキーマに置き換える機能をサポートするようになりました。詳細については、「データセットスキーマの置き換え」を参照してください。	2023 年 7 月 13 日
新機能	ユーザーパーソナライゼーションレシピまたはパーソナライズランキングレシピを使用する場合、Amazon Personalize バッチ推論ジョブでは、再トレーニングなしで段階的にインポートしたデータを使用できるようになりました。詳細については、「バッチレコメンデーションの取得」を参照してください。	2023 年 6 月 30 日

新機能	Amazon Personalize では、関連商品レコメンデーションのリクエストで指定した商品に基づいて商品をフィルタリングできるようになりました。フィルタについては、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。	2023 年 6 月 21 日
新機能	Amazon Personalize では、関連商品レコメンデーションのリクエストで指定した商品に基づいて商品をフィルタリングできるようになりました。フィルタについては、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。	2023 年 6 月 21 日
新機能	Amazon Personalize は、インターフェイス Amazon VPC エンドポイントを使用して、仮想プライベートクラウド (VPC) と Amazon Personalize の間のプライベート接続をサポートするようになりました。詳細については、「Amazon Personalize とインターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。	2023 年 6 月 12 日

新機能	Amazon Personalize では、レコメンダーまたはカスタムソリューションの作成時にトレーニング時に使用する列の設定がサポートされるようになりました。レコメンダーを作成する際の列の設定については、「レコメンダーの作成」を参照してください。ソリューション作成時の列の設定について詳しくは、「トレーニング時に使用する列の設定」を参照してください。	2023 年 5 月 30 日
新規ドキュメント機能	Amazon Personalize デベロッパーガイドに、Amazon Personalize の推奨事項による A/B テストの実施に関する情報が追加されました。詳細については、「A/B テストによるレコメンデーションの影響の測定」を参照してください。	2023 年 5 月 5 日
新機能	Amazon Personalize では、人気が類似商品レシピによって生成されるレコメンデーションにどのように影響するかを設定できるようになりました。詳細については、「類似アイテムのレシピ」を参照してください。	2023 年 4 月 21 日

新機能	Amazon Personalize では、Amazon SageMaker Data Wrangler を使用して 40 を超えるソースから Amazon Personalize データセットにデータをインポートできるようになりました。詳細については、「Amazon SageMaker Data Wrangler を使用したデータのインポート」を参照してください。	2023 年 4 月 14 日
新規ドキュメント機能	Amazon Personalize デベロッパーガイドに、新しい準備チェックリストが追加されました。このチェックリストは、独自のデータで Amazon Personalize を使用する準備に役立ちます。詳細については、「準備チェックリスト」を参照してください。	2023 年 2 月 9 日
新機能	Amazon Personalize は、データセットにインポートするデータのインサイトと統計の生成をサポートするようになりました。詳細については、「データセット内のデータの分析」を参照してください。	2023 年 1 月 25 日
新機能	Amazon Personalize は、カスタムデータセットグループ用の新しい Trending-Now レシピをサポートするようになりました。詳細については、「Trending-Now レシピ」を参照してください。	2023 年 1 月 6 日

新機能	Amazon Personalize は IAM ポリシーでのタグの使用をサポートするようになりました。詳細については、「IAM ポリシーでのタグの使用」を参照してください。	2022 年 12 月 28 日
新機能	Amazon Personalize は、トレーニングクォータ時にモデルで検討されるインタラクションの最大数の調整をサポートするようになりました。さらに、トレーニング時にモデルで検討されるユーザーの最大数 (クォータ制限) は適用されなくなりました。詳細については、「Amazon Personalize エンドポイントとクォータ」を参照してください。	2022 年 12 月 15 日
新機能	Amazon Personalize は、レコメンデーションがビジネスに与える影響を測定するメトリックスアトリビューションの作成をサポートするようになりました。詳細については、「レコメンデーションの影響の測定」を参照してください。	2022 年 11 月 17 日

新機能	Amazon Personalize のアクティブなソリューション、アクティブなキャンペーン、レコメンダー、フィルターの総数の割り当てが増えました。これらのクォータはそれぞれ、アカウントごとではなくデータセットグループごとに適用されるようになりました。クォータについては、「Amazon Personalize エンドポイントとクォータ」を参照してください。	2022 年 9 月 7 日
新機能	Amazon Personalize フィルタは、イベントタイプごとにユーザー 1 人あたり最大 100 件のインタラクションを考慮するようになりました。フィルタについては、「レコメンデーションとユーザーセグメントのフィルタリング」を参照してください。	2022 年 8 月 29 日
新機能	Amazon Personalize は、VIDEO_ON_DEMAND ドメインの新しい Trending now ユースケースをサポートするようになりました。詳細については、「VIDEO_ON_DEMAND ユースケース」を参照してください。	2022 年 8 月 17 日

新機能	Amazon Personalize では、別のプロモーションフィルタによるレコメンデーション内の商品のプロモーションをサポートするようになりました。商品のプロモーションについては、「レコメンデーション内の商品のプロモーション」を参照してください。	2022 年 8 月 12 日
新機能	Amazon Personalize では、プレースホルダーパラメータを含むフィルタ式での比較演算子の使用がサポートされるようになりました。フィルタ式の詳細については、「フィルタ式」を参照してください。	2022 年 8 月 12 日
新機能	Amazon Personalize は、データセットの段階的な一括更新をサポートするようになりました。データセットのインポートジョブを使用して、既存のデータを置き換えずにデータセットを更新できるようになりました。詳細については、「既存の一括データの更新」を参照してください。	2022 年 8 月 2 日
新機能	Amazon Personalize は、さまざまな言語の非構造化テキストメタデータを使用できるようになりました。詳細については、「非構造化テキストメタデータ」を参照してください。	2022 年 6 月 6 日

新機能	Amazon Personalize レコメンダーがオフラインメトリックスを生成するようになりました。これらのメトリクスを使用して、レコメンダーのパフォーマンスを評価できます。詳細については、「レコメンダーの評価」を参照してください。	2022 年 5 月 24 日
新機能	Amazon Personalize は、レコメンダーを停止し、後で再開する機能をサポートするようになりました。これにより、レコメンダーへの請求を一時停止し、使用時にのみ支払いを行うことができます。詳細については、「レコメンダーの停止とスタート」を参照してください。	2022 年 4 月 20 日
新機能	Amazon Personalize では、タグを使用して Amazon Personalize リソースを分類および管理できるようになりました。詳細については、「Amazon Personalize のリソースにタグ付けする」を参照してください。	2022 年 4 月 7 日

新機能	Amazon Personalize では、を使用したリソースの指定がサポートされるようになりました AWS CloudFormation。詳細については、「を使用したリソースの指定 AWS CloudFormation」を参照してください。	2022 年 3 月 11 日
新規ドキュメント機能	Amazon Personalize デベロッパーガイドに、Amazon Personalize で発生する可能性のあるエラーメッセージの一般的な質問への回答とトラブルシューティングのアドバイスを提供する新しいトラブルシューティングトピックが追加されました。詳細については、「トラブルシューティング」を参照してください。	2022 年 2 月 15 日
新機能	Amazon Personalize は、ビデオオンデマンドまたは e コマースドメイン向けのユースケース最適化リソースを使用したドメインデータセットグループの作成をサポートするようになりました。詳細については、「ドメインデータセットグループ」を参照してください。	2021 年 11 月 29 日

新機能	Amazon Personalize は、新しい USER_SEGMENTATION レシピを使用したユーザーセグメントの作成をサポートするようになりました。USER_SEGMENTATION レシピは、アイテム入力データに基づいてユーザーのセグメントを生成します。詳細については、「USER_SEGMENTATION のレシピ」を参照してください。	2021 年 11 月 29 日
新機能	Amazon Personalize は、新しい RELATED_ITEMS レコメンデーションレシピである Similar-Items をサポートするようになりました。Similar-Items レシピを使用して、インタラクションデータとアイテムメタデータの両方に基づいて類似アイテムのレコメンデーションを生成します。詳細については、「類似アイテムのレシピ」を参照してください。	2021 年 10 月 5 日
新規ドキュメント機能	Amazon Personalize デベロッパーガイドに、SDK for Java 2.x を使用した Amazon Personalize の利用に関する開始方法のチュートリアルが追加されました。詳細については、「開始方法 (SDK for Java 2.x)」を参照してください。	2021 年 8 月 25 日

新機能	Amazon Personalize は、Items データセットの非構造化テキストメタデータから有意義な情報を抽出できるようになりました。詳細については、「アイテムデータセット」を参照してください。	2021 年 6 月 9 日
新機能	Amazon Personalize は、ソリューションバージョンの作成を停止する (モデルのトレーニングを停止する) 機能をサポートするようになりました。詳細については、「ソリューションバージョンの作成の停止」を参照してください。	2021 年 5 月 20 日
新機能 (プレビューリリース)	Amazon Personalize は、収益の最大化などの関連性の最大化に加えて、目的に合わせてソリューションを最適化できるようになりました。この機能はプレビューリリースです。詳細については、「追加の目的のためのソリューションの最適化」を参照してください。	2021 年 5 月 18 日

新機能	Amazon Personalize は、分析と追跡のために、Amazon Personalize のデータセットのレコードを Amazon S3 バケットにエクスポートできるようになりました。詳細については、「データセットのエクスポート」を参照してください。	2021 年 4 月 26 日
新機能	Amazon Personalize は、User-Personalization でトレーニングした最新モデル (ソリューションバージョン) を 2 時間ごとに自動的に更新して、新しいデータを含めるようになりました。詳細については、「User-personalization レシピ」を参照してください。	2020 年 11 月 17 日
新機能	Amazon Personalize は、レコメンデーションを取得するときに指定した基準に基づいてレコメンデーションをフィルタリングできるようになりました。詳細については、「レコメンデーションのフィルタリング」を参照してください。	2020 年 11 月 10 日

新機能	Amazon Personalize は、ユーザーとアイテムを増分的にインポートする機能をサポートするようになりました。詳細については、「レコードの増分インポート」を参照してください。	2020 年 10 月 2 日
新機能	Amazon Personalize は、新しい USER_PERSONALIZATION レコメンデーションレシピをサポートするようになりました。USER_PERSONALIZATION 機能には、インプレッションデータのモデリング、アイテムの自動探索、コールドアイテムの自動選択が含まれます。詳細については、「User-personalization レシピ」を参照してください。	2020 年 8 月 5 日
新機能	Amazon Personalize は、カスタムフィルター式を使用して、アイテムとユーザーのメタデータに基づいてレコメンデーションをフィルタリングできるようになりました。詳細については、「レコメンデーションのフィルタリング」を参照してください。	2020 年 7 月 31 日

新機能	Amazon Personalize で、ユーザーが操作したアイテムに基づいて結果をフィルタリングできるようになりました。詳細については、「レコメンデーションのフィルタリング」を参照してください。	2020 年 6 月 3 日
新機能	Amazon Personalize は、推奨アイテムのスコアを公開するようになりました。スコアは、ユーザーが次に特定のアイテムを選択する Amazon Personalize モデルの確実性を表します。詳細については、「レコメンデーションの取得」を参照してください。	2020 年 4 月 3 日
新しいリージョン	Amazon Personalize は、アジアパシフィック (ソウル) リージョンのサポートを追加します。Amazon Personalize でサポートされている AWS リージョンの詳細なリストについては、アマゾン ウェブ サービスの全般のリファレンスの「AWS リージョンテーブル」または「AWS リージョンとエンドポイント」を参照してください。	2020 年 1 月 21 日

新機能	Amazon Personalize は、コンテキストメタデータに基づいてレコメンデーションを取得できるようになりました。詳細については、「レコメンデーションの取得」を参照してください。	2019 年 12 月 19 日
新しいリージョン	Amazon Personalize では、アジアパシフィック (ムンバイ)、アジアパシフィック (シドニー)、およびカナダ (中部) の各リージョンのサポートが追加されました。Amazon Personalize でサポートされている AWS リージョンの完全なリストについては、「Amazon Web Services 全般のリファレンス」のAWS「リージョンテーブル」またはAWS「リージョンとエンドポイント」を参照してください。	2019 年 12 月 18 日
新機能	Amazon Personalize は、バッチレコメンデーションワークフローをサポートするようになりました。詳細については、「バッチレコメンデーションの取得」を参照してください。	2019 年 11 月 14 日
Amazon Personalize の一般利用	Amazon Personalize が一般的にご利用いただけるようになりました。	2019 年 6 月 10 日

Amazon Personalize のプレビューのリリース	本書は Amazon Personalize ドキュメントのプレビュー版の初版リリースです。	2018 年 11 月 28 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。