aws

사용자 지정 레이블 가이드

Rekognition

Rekognition: 사용자 지정 레이블 가이드

Table of Contents

Amazon Rekognition Custom Labels란 무엇입니까?

Amazon Rekognition Custom Labels를 사용하면 비즈니스에 필요한 대로 이미지 안의 객체와 로고, 장면을 식별해 낼 수 있습니다. 예를 들어, 소셜 미디어 게시글에서 로고를 찾거나 매장에서 제품을 식별하거나 어셈블리 라인에서 기계 부품을 분류하거나 정상적으로 운영되는 공장과 결함이 있는 공장을 구별하거나 이미지에서 애니메이션 캐릭터를 탐지할 수 있습니다.

이미지 분석을 위한 사용자 지정 모델을 개발하는 것은 시간, 전문 지식 및 리소스가 필요한 중요한 작업입니다. 완료하는 데 몇 개월이 걸리는 경우가 많습니다. 또한 정확한 결정을 내리기에 충분한 데이터를 모델에 제공하려면 수작업으로 레이블을 지정한 이미지가 수천에서 수만 개나 필요할 수 있습니다. 이 데이터를 생성하는 데 몇 개월이 걸릴 수 있으며, 기계 학습에 사용할 수 있도록 준비하는 데에 레이블링 작업자가 다수 필요할 수 있습니다.

Amazon Rekognition Custom Labels는 이미 여러 카테고리에 걸쳐 수천만 개의 이미지에 대해 훈련된 Amazon Rekognition의 기존 기능을 확장합니다. 수천 개의 이미지 대신 사용 사례에 맞는 소량의 훈련 이미지 세트(일반적으로 수백 개 이하)만 업로드하면 됩니다. 이를 위해 사용하기 쉬운 콘솔을 사용할 수 있습니다. 이미지에 이미 레이블이 지정되어 있는 경우 Amazon Rekognition Custom Labels를 사용하면 짧은 시간 내에 모델 훈련을 시작할 수 있습니다. 그렇지 않은 경우 레이블 지정 인터페이스 내에서 이미지에 직접 레이블을 지정하거나 Amazon SageMaker Ground Truth를 사용하여 이미지에 레이블을 지정할 수 있습니다.

Amazon Rekognition Custom Labels가 이미지 세트에서 훈련을 시작하면 단 몇 시간만에 사용자 지정 이미지 분석 모델을 생성할 수 있습니다. Amazon Rekognition Custom Labels는 백그라운드에서 훈련 데이터를 자동으로 로드 및 검사하고, 적합한 기계 학습 알고리즘을 선택하고, 모델을 훈련하고, 모델 성능 지표를 제공합니다. 그런 다음 Amazon Rekognition Custom Labels API를 통해 사용자 지정 모델을 사용하고 이를 애플리케이션에 통합할 수 있습니다.

주제

주요 이점

간소화된 데이터 레이블링

Amazon Rekognition Custom Labels 콘솔은 이미지에 레이블을 빠르고 간단하게 레이블링할 수 있는 시각적 인터페이스를 제공합니다. 인터페이스를 통해 전체 이미지에 레이블을 적용할 수 있습니다. 또한 클릭 앤 드래그 인터페이스가 있는 경계 상자를 사용하여 이미지의 특정 객체를 식별하고 레이블을 지정할 수 있습니다. 또는 데이터 세트가 큰 경우 Amazon SageMaker Ground Truth를 사용하여 이미지에 효율적으로 레이블을 지정할 수 있습니다.

자동화된 기계 학습

사용자 지정 모델을 구축하는 데 기계 학습 전문 지식이 필요하지 않습니다. Amazon Rekognition Custom Labels에는 기계 학습을 대신 처리하는 자동 기계 학습(AutoML) 기능이 포함되어 있습니다. 훈련 이미지가 제공되면 Amazon Rekognition Custom Labels는 데이터를 자동으로 로드 및 검사하고, 적합한 기계 학습 알고리즘을 선택하고, 모델을 훈련하고, 모델 성능 지표를 제공할 수 있습니다.

간소화된 모델 평가, 추론 및 피드백

테스트 세트에서 사용자 지정 모델의 성능을 평가할 수 있습니다. 테스트 세트의 모든 이미지에 대해 모델의 예측과 할당된 레이블을 나란히 비교한 결과를 확인할 수 있습니다. 정밀도, 재현율, F1 점수, 신뢰도 점수와 같은 자세한 성능 지표를 검토할 수도 있습니다. 모델을 이미지 분석에 즉시 사용할 수도 있고, 더 많은 이미지로 새 버전을 반복하고 재훈련하여 성능을 개선할 수도 있습니다. 모델을 사용하기 시작한 후에는 예측을 추적하고 실수를 수정하고 피드백 데이터를 사용하여 새 모델 버전을 재훈련하고 성능을 개선합니다.

Amazon Rekognition Custom Labels 사용 선택

Amazon Rekognition은 이미지에서 레이블(객체, 장면 및 개념)을 찾는 데 사용할 수 있는 두 가지 기능을 제공합니다. 그것은 바로 Amazon Rekognition Custom Labels와 Amazon Rekognition Image 레이블 감지 기능입니다. 다음 정보를 통해 어떤 기능을 사용할지 결정하세요.

Amazon Rekognition Image 레이블 감지

Amazon Rekognition Image의 레이블 감지 기능을 사용하면 기계 학습 모델을 생성할 필요 없이 대규모로 이미지와 동영상의 공통 레이블을 식별, 분류 및 검색할 수 있습니다. 예를 들어 자동차, 트럭, 토마토, 농구공, 축구공과 같은 수천 개의 일반적인 물체를 쉽게 감지할 수 있습니다.

애플리케이션에서 공통 레이블을 찾아야 하는 경우 모델을 훈련할 필요가 없으므로 Amazon Rekognition Image 레이블 감지를 사용하는 것이 좋습니다. Amazon Rekognition Image 레이블 감지에서 찾은 레이블 목록을 가져오려면 레이블 감지를 참조하세요.

애플리케이션에서 Amazon Rekognition Image 레이블 감지로 찾을 수 없는 레이블(예: 조립 라인의 사용자 지정 기계 부품)을 찾아야 하는 경우에는 Amazon Rekognition Custom Labels를 사용하는 것이 좋습니다.

Amazon Rekognition Custom Labels

Amazon Rekognition Custom Labels를 사용하면 이미지에서 여러분의 비즈니스 요구 사항에 맞는 레이블(객체, 로고, 장면, 개념)을 찾는 기계 학습 모델을 쉽게 훈련할 수 있습니다.

Amazon Rekognition Custom Labels는 이미지를 분류(이미지 수준 예측)하거나 이미지에서 객체 위치를 감지(객체/경계 상자 수준 예측)할 수 있습니다.

Amazon Rekognition Custom Labels는 객체 및 장면의 유형을 더욱 유연하게 감지하도록 해줍니다. 예를 들어 Amazon Rekognition Image 레이블 감지 기능을 사용하여 식물과 나뭇잎을 찾을 수 있습니다. 건강한 식물, 손상된 식물, 감염된 식물을 구분하려면 Amazon Rekognition Custom Labels를 사용해야 합니다.

다음은 Amazon Rekognition Custom Labels 사용 방법의 예입니다.

- 선수 유니폼과 헬멧의 팀 로고 식별
- 조립 라인에서 특정 기계 부품 또는 제품을 구별
- 미디어 라이브러리의 만화 캐릭터 식별
- 소매점 진열대에서 특정 브랜드의 제품 찾기
- 농산물 품질 분류(예: 썩은 것, 익은 것, 익지 않은 것)

> **Note**
>
> Amazon Rekognition Custom Labels는 얼굴을 분석하거나, 텍스트를 감지하거나, 이미지에서 안전하지 않은 이미지 콘텐츠를 찾도록 설계되지 않았습니다. Amazon Rekognition Image를 사용하여 이러한 작업을 수행할 수 있습니다. 자세한 내용은 Amazon Rekognition이란 무엇입니까?를 참조하세요.

Amazon Rekognition Custom Labels를 처음 사용하시나요?

Amazon Rekognition Custom Labels를 처음 사용한다면, 다음 항목을 순서대로 읽어보기를 권장합니다.

1. Amazon Rekognition Custom Labels 설정: 이 항목에서는 계정 세부 정보를 설정합니다.
2. Amazon Rekognition Custom Labels의 이해: 이 항목에서는 모델 생성 워크플로에 대해 알아봅니다.
3. Amazon Rekognition Custom Labels 시작하기: 이 항목에서는 Amazon Rekognition Custom Labels에서 생성한 예제 프로젝트를 사용하여 모델을 훈련합니다.
4. 튜토리얼: 이미지 분류: 이 항목에서는 생성한 데이터 세트로 이미지를 분류하는 모델을 훈련하는 방법을 알아봅니다.

Amazon Rekognition Custom Labels 설정

다음 지침은 Amazon Rekognition Custom Labels 콘솔과 SDK를 설정하는 방법을 보여줍니다.

Amazon Rekognition Custom Labels 콘솔은 다음 브라우저에서 사용할 수 있습니다.

- Chrome: 버전 21 이상
- Firefox: 버전 27 이상
- Microsoft Edge: 버전 88 이상
- Safari: 버전 7 이상 또한 Amazon Rekognition Custom Labels 콘솔에서는 Safari를 사용하여 경계 상자를 그릴 수 없습니다. 자세한 정보는 경계 상자로 객체에 레이블 지정을 참조하세요.

Amazon Rekognition Custom Labels를 처음으로 사용하기 전에 다음 작업을 완료해야 합니다.

주제

1단계: AWS 계정 생성

이 단계에서는 AWS 계정을 만들고, 관리 사용자를 만들고, SDK에 프로그래밍 방식으로 액세스 권한을 부여하는 방법을 알아봅니다. AWS

주제

가입하여 AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

프로그래밍 방식 액세스

사용자가 AWS 외부 사용자와 상호 작용하려는 경우 프로그래밍 방식의 액세스가 필요합니다. AWS Management Console프로그래밍 방식의 액세스 권한을 부여하는 방법은 액세스하는 사용자 유형에 따라 다릅니다. AWS

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API	사용하고자 하는 인터페이스에 대한 지침을 따릅니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
(IAM Identity Center가 관리하는 사용자)	에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	• AWS CLI 에 대한 내용은 사용 설명서의 AWS CLI사용을 AWS IAM Identity Center위한 구성을 참조하십시오.AWS Command Line Interface • AWS SDK, 도구 및 AWS API의 경우 AWS SDK 및 도구 참조 안내서의 IAM ID 센터 인증을 참조하십시오.
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 방식 요청에 서명할 수 있습니다. AWS	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따르십시오.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 에 대한 내용은 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하십시오. AWS CLIAWS Command Line Interface • AWS SDK 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용한 인증을 참조하십시오.AWS • AWS API의 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하십시오.

2단계: Amazon Rekognition Custom Labels 콘솔 권한 설정

Amazon Rekognition 콘솔을 사용하려면 적절한 권한을 갖도록 추가해야 합니다. 훈련 파일을 콘솔 버킷이 아닌 다른 버킷에 저장하려면 추가 권한이 필요합니다.

주제

- 콘솔 액세스 허용
- 외부 Amazon S3 버킷에 액세스
- 권한 할당

콘솔 액세스 허용

Amazon Rekognition 사용자 지정 레이블 콘솔을 사용하려면 Amazon S3, Ground Truth 및 Amazon SageMaker Rekognition 사용자 지정 레이블을 다루는 다음과 같은 IAM 정책이 필요합니다. 권한을 할당하는 방법에 대한 자세한 내용은 권한 할당 항목을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "s3Policies",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:GetBucketVersioning",
```

```
                "s3:GetObjectVersionTagging",
                "s3:PutBucketCORS",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:PutBucketVersioning",
                "s3:PutObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::custom-labels-console-*"

            ]
        },
        {
            "Sid": "rekognitionPolicies",
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "groundTruthPolicies",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:*"
            ],
            "Resource": "*"
        }
    ]
}
```

외부 Amazon S3 버킷에 액세스

새 AWS 지역에서 Amazon Rekognition 사용자 지정 레이블 콘솔을 처음 열면 Amazon Rekognition 사용자 지정 레이블이 프로젝트 파일을 저장하는 데 사용되는 버킷 (콘솔 버킷) 을 생성합니다. 또는 자체 Amazon S3 버킷(외부 버킷)을 사용하여 이미지 또는 매니페스트 파일을 콘솔에 업로드할 수 있습니다. 외부 버킷을 사용하려면 이전 정책에 다음 정책 블록을 추가하세요. `my-bucket`을 버킷의 이름으로 바꿉니다.

```
        {
```

```
            "Sid": "s3ExternalBucketPolicies",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket*"
            ]
        }
```

권한 할당

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 해당 지역의 사용자 및 그룹: AWS IAM Identity Center

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.
- ID 제공자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

3단계: 콘솔 버킷 생성

Amazon Rekognition Custom Labels 프로젝트를 사용하여 모델을 생성하고 관리할 수 있습니다. 새 AWS 지역에서 Amazon Rekognition 사용자 지정 레이블 콘솔을 처음 열면 Amazon Rekognition 사용

자 지정 레이블이 Amazon S3 버킷 (콘솔 버킷) 을 생성하여 프로젝트를 저장합니다. AWS SDK 작업이나 콘솔 작업 (예: 데이터세트 생성) 에서 버킷 이름을 사용해야 할 수 있으므로 나중에 참조할 수 있는 곳에 콘솔 버킷 이름을 기록해 두어야 합니다.

버킷 이름의 형식은 `custom-labels-console-<region>-<random value>`입니다. 임의 값은 버킷 이름 간에 충돌이 발생하지 않도록 합니다.

콘솔 버킷을 생성하려면

1. 사용자에게 적절한 권한이 있는지 확인합니다. 자세한 정보는 콘솔 액세스 허용을 참조하세요.
2. 에 AWS Management Console 로그인하고 https://console.aws.amazon.com/rekognition/ 에서 Amazon Rekognition 콘솔을 엽니다.
3. Get started를 선택합니다.
4. 현재 AWS 리전에서 콘솔을 처음 연 경우 최초 설정 대화 상자가 표시됩니다.

   a. 표시된 Amazon S3 버킷의 이름을 복사합니다. 나중에 이 정보가 필요합니다.

   b. S3 버킷 생성을 선택하여 Amazon Rekognition Custom Labels가 사용자를 대신하여 Amazon S3 버킷(콘솔 버킷)을 생성하게 하세요.
5. 브라우저 창을 닫습니다.

4단계: 및 SDK 설정 AWS CLIAWS

Amazon Rekognition 사용자 지정 레이블은 () 및 AWS Command Line Interface SDK와 함께AWS CLI 사용할 수 있습니다. AWS 터미널에서 Amazon Rekognition Custom Labels 작업을 실행해야 하는 경우 AWS CLI를 설치하세요. 애플리케이션을 생성하는 경우 사용 중인 프로그래밍 언어용 AWS SDK를 다운로드하십시오.

주제

- SDKS 설치 AWS
- 프로그래밍 방식 액세스 권한 부여
- SDK 권한 설정
- Amazon Rekognition Custom Labels 작업을 직접 호출합니다.

SDKS 설치 AWS

단계에 따라 AWS SDK를 다운로드하고 구성합니다.

AWS CLI 및 SDK를 설정하려면 AWS

- 및 사용하려는 AWS SDK를 다운로드하여 설치합니다. AWS CLI 이 안내서는 AWS CLI, 자바, Python에 대한 예제를 제공합니다. AWS SDK 설치에 대한 자세한 내용은 Amazon Web Services 용 도구를 참조하십시오.

프로그래밍 방식 액세스 권한 부여

로컬 컴퓨터 또는 Amazon Elastic Compute Cloud 인스턴스와 같은 기타 AWS 환경에서 이 안내서의 AWS CLI 및 코드 예제를 실행할 수 있습니다. 예제를 실행하려면 예제에서 사용하는 AWS SDK 작업에 대한 액세스 권한을 부여해야 합니다.

주제

- 로컬 컴퓨터에서 코드 실행
- 환경에서 AWS 코드 실행

로컬 컴퓨터에서 코드 실행

로컬 컴퓨터에서 코드를 실행하려면 단기 자격 증명을 사용하여 사용자에게 AWS SDK 작업에 대한 액세스 권한을 부여하는 것이 좋습니다. 로컬 컴퓨터에서 AWS CLI 및 코드 예제를 실행하는 방법에 대한 자세한 내용은 을 참조하십시오로컬 컴퓨터에서 프로필 사용.

AWS 외부 사용자와 상호 작용하려는 사용자는 프로그래밍 방식의 액세스가 필요합니다. AWS Management Console프로그래밍 방식의 액세스 권한을 부여하는 방법은 액세스하는 사용자 유형에 따라 다릅니다. AWS

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI 에 대한 내용은 사용 설명서의 AWS CLI사용을 AWS IAM Identity Center위

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		한 구성을 참조하십시오.AWS Command Line Interface • AWS SDK, 도구 및 AWS API의 경우 AWS SDK 및 도구 참조 안내서의 IAM ID 센터 인증을 참조하십시오.
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 방식 요청에 서명할 수 있습니다. AWS	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따르십시오.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 에 대한 내용은 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하십시오. AWS CLIAWS Command Line Interface • AWS SDK 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용한 인증을 참조하십시오.AWS • AWS API의 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하십시오.

로컬 컴퓨터에서 프로필 사용

에서 생성한 단기 자격 증명으로 이 가이드의 예제를 AWS CLI 실행하고 코딩할 수 있습니다. 로컬 컴퓨터에서 코드 실행 보안 인증 정보 및 기타 설정 정보를 가져오려면, 예제에서는 `custom-labels-access` 이름의 프로필을 사용합니다. 예를 들어:

```
session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")
```

프로필이 나타내는 사용자는 Amazon Rekognition 사용자 지정 레이블 SDK 작업 AWS 및 예제에 필요한 기타 SDK 작업을 호출할 권한이 있어야 합니다. 자세한 정보는 SDK 권한 설정을 참조하세요. 권한을 할당하려면 SDK 권한 설정 항목을 참조하세요.

AWS CLI 및 코드 예제와 호환되는 프로필을 생성하려면 다음 중 하나를 선택하십시오. 생성한 프로필의 이름이 `custom-labels-access`인지 확인하세요.

- IAM으로 관리하는 사용자 — IAM 역할로 전환(AWS CLI)의 지침을 따르세요.
- 인력 ID (사용자 관리 대상 AWS IAM Identity Center) — 사용할 AWS CLI 구성의 지침을 따르십시오. AWS IAM Identity Center코드 예제의 경우 IAM ID 센터를 통한 인증을 지원하는 AWS 툴킷을 지원하는 통합 개발 환경(IDE)을 사용하는 것이 좋습니다. Java 예제는 Java로 구축 시작을 참조하세요. Python 예제는 Python으로 구축 시작을 참조하세요. 자세한 정보는 IAM Identity Center 보안 인증 정보를 참조하세요.

> **Note**
>
> 코드를 사용하여 단기 보안 인증 정보를 얻을 수 있습니다. 자세한 내용은 IAM 역할로 전환(API)을 참조하세요. IAM Identity Center의 경우 CLI 액세스를 위한 IAM 역할 자격 증명 가져오기의 지침에 따라 역할에 대한 단기 보안 인증 정보를 받으세요.

환경에서 AWS 코드 실행

AWS Lambda 함수에서 실행되는 프로덕션 코드와 같은 AWS 환경에서는 사용자 자격 증명을 사용하여 AWS SDK 호출에 서명해서는 안 됩니다. 대신 코드에 필요한 권한을 정의하는 역할을 구성합니다. 그런 다음 코드가 실행되는 환경에 역할을 연결합니다. 역할을 연결하고 임시 보안 인증 정보를 사용할 수 있게 하는 방법은 코드가 실행되는 환경에 따라 달라집니다.

- AWS Lambda 함수 — Lambda 함수의 실행 역할을 맡을 때 Lambda가 함수에 자동으로 제공하는 임시 자격 증명을 사용합니다. 보안 인증 정보는 Lambda 환경 변수에서 사용할 수 있습니다. 프로필을 지정할 필요가 없습니다. 자세한 내용을 알아보려면 Lambda 실행 역할을 참조하세요.
- Amazon EC2 - Amazon EC2 인스턴스 메타데이터 엔드포인트 보안 인증 인증 정보를 사용합니다. 공급자는 Amazon EC2 인스턴스에 연결한 Amazon EC2 인스턴스 프로파일을 사용하여 자동으로

보안 인증 정보를 생성하고 새로 고칩니다. 자세한 내용은 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

- Amazon Elastic 컨테이너 서비스 - 컨테이너 보안 인증 정보 공급자를 사용하세요. Amazon ECS는 메타데이터 엔드포인트로 보안 인증 정보를 전송하고 새로 고칩니다. 지정한 작업 IAM 역할은 애플리케이션이 사용하는 보안 인증 정보를 관리하기 위한 전략을 제공합니다. 자세한 내용은 AWS 서비스와 상호 작용을 참조하세요.

보안 인증 정보 공급자에 대한 자세한 정보는 표준화된 보안 인증 정보 공급자를 참조하세요.

SDK 권한 설정

Amazon Rekognition Custom Labels SDK 작업을 사용하려면 Amazon Rekognition Custom Labels API 및 모델 훈련에 사용되는 Amazon S3 버킷에 대한 액세스 권한이 필요합니다.

주제

- SDK 작업 권한 부여
- AWS SDK 사용을 위한 정책 업데이트
- 권한 할당

SDK 작업 권한 부여

작업을 수행하는 데 필요한 권한만 부여하는 것을 권장합니다(최소 권한). 예를 들어 DetectCustomLabels, 호출하려면 수행 권한이 필요합니다. `rekognition:DetectCustomLabels` 작업에 대한 권한을 찾으려면 API 참조를 확인하세요.

애플리케이션을 막 시작하는 경우 필요한 특정 권한을 모를 수 있으므로 더 광범위한 권한으로 시작할 수 있습니다. AWS 관리형 정책은 시작하는 데 도움이 되는 권한을 제공합니다. `AmazonRekognitionCustomLabelsFullAccess` AWS 관리형 정책을 사용하여 Amazon Rekognition 사용자 지정 레이블 API에 대한 전체 액세스 권한을 얻을 수 있습니다. 자세한 내용은 AWS 관리형 정책을 참조하십시오 AmazonRekognitionCustomLabelsFullAccess. 애플리케이션에 필요한 권한을 알고 있는 경우, 사용자에게 필요한 고객 관리형 정책을 정의하여 권한을 줄이세요. 자세한 내용은 고객 관리형 정책을 참조하세요.

권한을 할당하려면 권한 할당 항목을 참조하세요.

AWS SDK 사용을 위한 정책 업데이트

Amazon Rekognition 사용자 지정 레이블의 최신 릴리스와 함께 AWS SDK를 사용하려면 더 이상 Amazon Rekognition 사용자 지정 레이블에 교육 및 테스트 이미지가 포함된 Amazon S3 버킷에 액세스할 수 있는 권한을 부여할 필요가 없습니다. 이전에 권한을 추가한 경우 권한을 제거할 필요가 없습니다. 원하는 경우 보안 주체에 대한 서비스가 `rekognition.amazonaws.com`인 버킷에서 정책을 제거하세요. 예:

```
"Principal": {
    "Service": "rekognition.amazonaws.com"
}
```

자세한 내용은 버킷 정책 사용을 참조하세요.

권한 할당

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 대상 사용자 및 그룹: AWS IAM Identity Center

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.
- ID 제공자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

Amazon Rekognition Custom Labels 작업을 직접 호출합니다.

다음 코드를 실행하여 Amazon Rekognition Custom Labels API를 직접 호출할 수 있는지 확인합니다. 코드에는 현재 AWS 지역의 AWS 계정 내 프로젝트가 나열됩니다. 이전에 프로젝트를 생성하지 않은 경우 응답은 비어 있지만 `DescribeProjects` 작업을 직접 호출할 수 있다는 확인은 표시됩니다.

일반적으로 예제 함수를 직접 호출하려면 AWS SDK Rekognition 클라이언트와 기타 필수 파라미터가 필요합니다. AWS SDK 클라이언트는 기본 함수에서 선언됩니다.

코드가 실패할 경우 사용하는 사용자에게 올바른 권한이 있는지 확인하세요. 또한 Amazon Rekognition 사용자 지정 레이블을 모든 AWS 지역에서 사용할 수 있는 것은 아니므로 사용 중인 지역도 확인하십시오. AWS

Amazon Rekognition Custom Labels 작업을 직접 호출하려면

1. 아직 설치하지 않았다면 및 SDK를 설치하고 구성하십시오 AWS CLI . AWS 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 다음 예제 코드를 사용하여 프로젝트를 확인하십시오.

   CLI

   `describe-projects` 명령어를 사용하여 계정에 있는 프로젝트를 나열합니다.

   ```
   aws rekognition describe-projects \
   --profile custom-labels-access
   ```

   Python

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   This example shows how to describe your Amazon Rekognition Custom Labels
    projects.
   If you haven't previously created a project in the current AWS Region,
   the response is an empty list, but does confirm that you can call an
   Amazon Rekognition Custom Labels operation.
   """
   from botocore.exceptions import ClientError
   import boto3

   def describe_projects(rekognition_client):
       """
       Lists information about the projects that are in in your AWS account
       and in the current AWS Region.
   ```

```
    : param rekognition_client: A Boto3 Rekognition client.
    """
    try:
        response = rekognition_client.describe_projects()
        for project in response["ProjectDescriptions"]:
            print("Status: " + project["Status"])
            print("ARN: " + project["ProjectArn"])
            print()
        print("Done!")
    except ClientError as err:
        print(f"Couldn't describe projects. \n{err}")
        raise


def main():
    """
    Entrypoint for script.
    """

    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    describe_projects(rekognition_client)


if __name__ == "__main__":
    main()
```

Java V2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class Hello {

    public static final Logger logger = Logger.getLogger(Hello.class.getName());

    public static void describeMyProjects(RekognitionClient rekClient) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument

        List<String> projectNames = new ArrayList<String>();


        descProjects = DescribeProjectsRequest.builder().build();

        // Display useful information for each project.

        DescribeProjectsResponse resp =
 rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
 {

            System.out.println("ARN: " + projectDescription.projectArn());
            System.out.println("Status: " +
 projectDescription.statusAsString());
            if (projectDescription.hasDatasets()) {
                for (DatasetMetadata datasetDescription :
 projectDescription.datasets()) {
                    System.out.println("\tdataset Type: " +
 datasetDescription.datasetTypeAsString());
                    System.out.println("\tdataset ARN: " +
 datasetDescription.datasetArn());
```

```
                    System.out.println("\tdataset Status: " +
 datasetDescription.statusAsString());
                }
            }
            System.out.println();
        }

    }

    public static void main(String[] args) {

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

5단계: (선택 사항) 훈련 파일 암호화

다음 옵션 중 하나를 선택하여 콘솔 버킷 또는 외부 Amazon S3 버킷에 있는 Amazon Rekognition Custom Labels 매니페스트 파일 및 이미지 파일을 암호화할 수 있습니다.

- Amazon S3 관리형 키(SSE-S3)를 사용하세요.

- 를 사용하세요. AWS KMS key

> **Note**
>
> 직접 호출하는 IAM 보안 주체에는 파일을 해독할 수 있는 권한이 필요합니다. 자세한 정보는 로 암호화된 파일을 복호화하고 있습니다. AWS Key Management Service을 참조하세요.

Amazon S3 버킷 암호화에 대한 자세한 내용은 Amazon S3 버킷에 대한 Amazon S3 기본 서버 측 암호화 동작 설정을 참조하세요.

로 암호화된 파일을 복호화하고 있습니다. AWS Key Management Service

AWS Key Management Service (KMS) 를 사용하여 Amazon Rekognition 사용자 지정 레이블 매니페스트 파일 및 이미지 파일을 암호화하는 경우 Amazon Rekognition 사용자 지정 레이블을 호출하는 IAM 보안 주체를 KMS 키의 키 정책에 추가하십시오. 이렇게 하면 Amazon Rekognition Custom Labels가 교육 전에 매니페스트와 이미지 파일을 해독할 수 있습니다. 자세한 정보는 사용자 지정 AWS KMS 키를 사용하는 기본 암호화가 있는 내 Amazon S3 버킷을 참조하세요. 사용자가 버킷에서 다운로드하고 버킷에 업로드하도록 허용하려면 어떻게 해야 합니까?

IAM 보안 주체에는 KMS 키에 대한 다음 권한이 필요합니다.

- kms: GenerateDataKey
- kms:Decrypt

자세한 내용은 AWS Key Management Service에 저장된 KMS 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.

복사한 훈련 및 테스트 이미지 암호화

모델을 훈련하기 위해 Amazon Rekognition Custom Labels는 소스 훈련 및 테스트 이미지의 사본을 만듭니다. 기본적으로 복사된 이미지는 AWS가 소유하고 관리하는 키로 암호화됩니다. 사용자의 AWS KMS key를 사용하는 방법도 있습니다. 자체 KMS 키를 사용하는 경우 KMS 키에 다음 권한이 필요합니다.

- kms: CreateGrant
- kms: DescribeKey

콘솔로 모델을 훈련하거나 `CreateProjectVersion` 작업을 직접 호출할 때 KMS 키를 지정할 수도 있습니다. 사용하는 KMS 키는 Amazon S3 버킷의 매니페스트 및 이미지 파일을 암호화하는 데 사용하는 KMS 키와 같지 않아도 됩니다. 자세한 정보는 5단계: (선택 사항) 훈련 파일 암호화을 참조하세요.

자세한 내용은 AWS Key Management Service 개념을 참조하세요. 소스 이미지는 영향을 받지 않습니다.

모델 훈련에 대한 자세한 내용은 Amazon Rekognition Custom Labels 모델 훈련 항목을 참조하세요.

6단계: (선택 사항) 이전 데이터 세트를 새 프로젝트와 연결

Amazon Rekognition Custom Labels는 이제 프로젝트와 함께 데이터 세트를 관리합니다. 생성한 이전 데이터 세트는 읽기 전용이므로 프로젝트에 연결해야 사용할 수 있습니다. 콘솔에서 프로젝트의 세부 정보 페이지를 열면 프로젝트 모델의 최신 버전을 훈련한 데이터 세트가 프로젝트에 자동으로 연결됩니다. SDK를 사용하는 경우 데이터셋과 프로젝트가 자동으로 연결되지 않습니다. AWS

연결되지 않은 이전 데이터 세트는 모델 훈련에 사용된 적이 없거나 이전 버전의 모델을 훈련하는 데 사용된 적이 없습니다. 이전 데이터 세트 페이지에는 관련 데이터 세트와 연결되지 않은 데이터 세트가 모두 표시됩니다.

연결되지 않은 이전 데이터 세트를 사용하려면 이전 데이터 세트 페이지에서 새 프로젝트를 만드세요. 데이터 세트는 새 프로젝트의 훈련 데이터 세트가 됩니다. 이전 데이터 세트에는 여러 연결이 있을 수 있으므로 이미 연결된 데이터 세트에 대한 프로젝트를 만들 수도 있습니다.

이전 데이터 세트를 새 프로젝트에 연결하는 방법

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다.
3. 왼쪽 탐색 창에서 이전 데이터 세트를 선택합니다.
4. 데이터 세트 보기에서 프로젝트와 연결하려는 이전 데이터 세트를 선택합니다.
5. 데이터 세트로 프로젝트 만들기를 선택합니다.
6. 프로젝트 생성 페이지에서 프로젝트 이름에 새 프로젝트의 이름을 입력합니다.
7. 프로젝트를 생성하려면 프로젝트 생성을 선택합니다. 프로젝트를 만드는 데 시간이 걸릴 수 있습니다.
8. 프로젝트를 사용합니다. 자세한 정보는 Amazon Rekognition Custom Labels의 이해을 참조하세요.

이전 데이터 세트를 테스트 데이터 세트로 사용

먼저 이전 데이터 세트를 새 프로젝트에 연결하여 이전 데이터 세트를 기존 프로젝트의 테스트 데이터 세트로 사용할 수 있습니다. 그런 다음 새 프로젝트의 훈련 데이터 세트를 기존 프로젝트의 테스트 데이터 세트에 복사합니다.

이전 데이터 세트를 테스트 데이터 세트로 사용하려면

1. 6단계: (선택 사항) 이전 데이터 세트를 새 프로젝트와 연결의 지침에 따라 이전 데이터 세트를 새 프로젝트에 연결하세요.
2. 새 프로젝트의 훈련 데이터 세트를 복사하여 기존 프로젝트에서 테스트 데이터 세트를 생성합니다. 자세한 정보는 기존 데이터 세트을 참조하세요.
3. Amazon Rekognition Custom Labels 프로젝트 삭제(콘솔)의 지침에 따라 새 프로젝트를 삭제하세요.

또는 이전 데이터 세트의 매니페스트 파일을 사용하여 테스트 데이터 세트를 만들 수도 있습니다. 자세한 내용은 매니페스트 파일 생성을(를) 참조하세요.

Amazon Rekognition Custom Labels의 이해

이 섹션에서는 콘솔 및 SDK를 통해 Amazon Rekognition 사용자 지정 레이블 모델을 교육하고 사용하는 워크플로의 개요를 제공합니다. AWS

> **Note**
>
> Amazon Rekognition Custom Labels는 이제 프로젝트 내에서 데이터 세트를 관리합니다. 콘솔과 SDK를 사용하여 프로젝트용 데이터세트를 생성할 수 있습니다. AWS 이전에 Amazon Rekognition Custom Labels를 사용한 적이 있는 경우 이전 데이터 세트를 새 프로젝트와 연결해야 할 수 있습니다. 자세한 정보는 6단계: (선택 사항) 이전 데이터 세트를 새 프로젝트와 연결섹션을 참조하십시오.

주제

모델 유형 결정

먼저 비즈니스 목표에 따라 훈련할 모델 유형을 결정합니다. 예를 들어 소셜 미디어 게시물에서 로고를 찾거나, 매장 진열대에서 제품을 식별하거나, 조립 라인에서 기계 부품을 분류하도록 모델을 훈련할 수 있습니다.

Amazon Rekognition Custom Labels는 다음 유형의 모델을 훈련할 수 있습니다.

- 객체, 장면 및 개념 찾기
- 객체 위치 찾기
- 브랜드 위치 찾기

Amazon Rekognition Custom Labels는 훈련할 모델 유형을 결정하는 데 도움이 되도록 사용할 수 있는 예제 프로젝트를 제공합니다. 자세한 정보는 Amazon Rekognition Custom Labels 시작하기을 참조하세요.

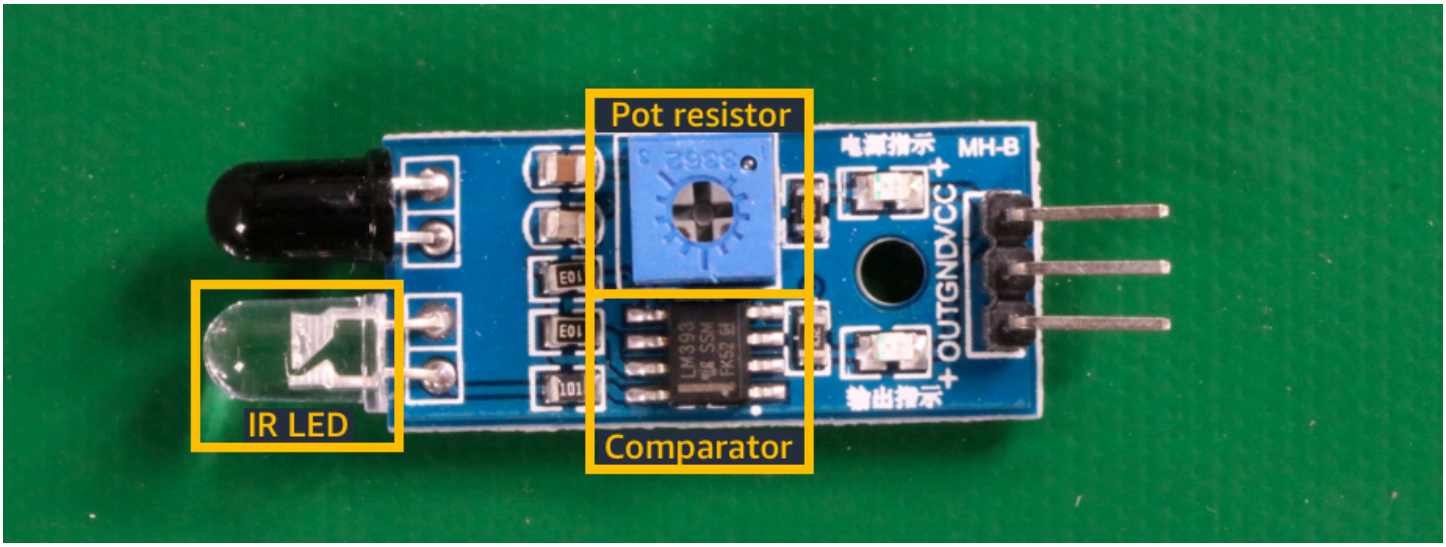
객체, 장면 및 개념 찾기

모델은 전체 이미지와 관련된 객체, 장면 및 개념의 분류를 예측합니다. 예를 들어 이미지에 관광 명소가 포함되어 있는지 여부를 판단하는 모델을 훈련할 수 있습니다. 예제 프로젝트에 관해서는 이미지 분류 항목을 참조하세요. 다음 호수 이미지는 사물, 풍경, 개념을 인식할 수 있는 이미지 종류의 예입니다.

또는 이미지를 여러 범주로 분류하는 모델을 훈련할 수도 있습니다. 예를 들어, 이전 이미지에는 하늘색, 반사 또는 호수와 같은 범주가 있을 수 있습니다. 예제 프로젝트에 관해서는 다중 레이블 이미지 분류 항목을 참조하세요.

객체 위치 찾기

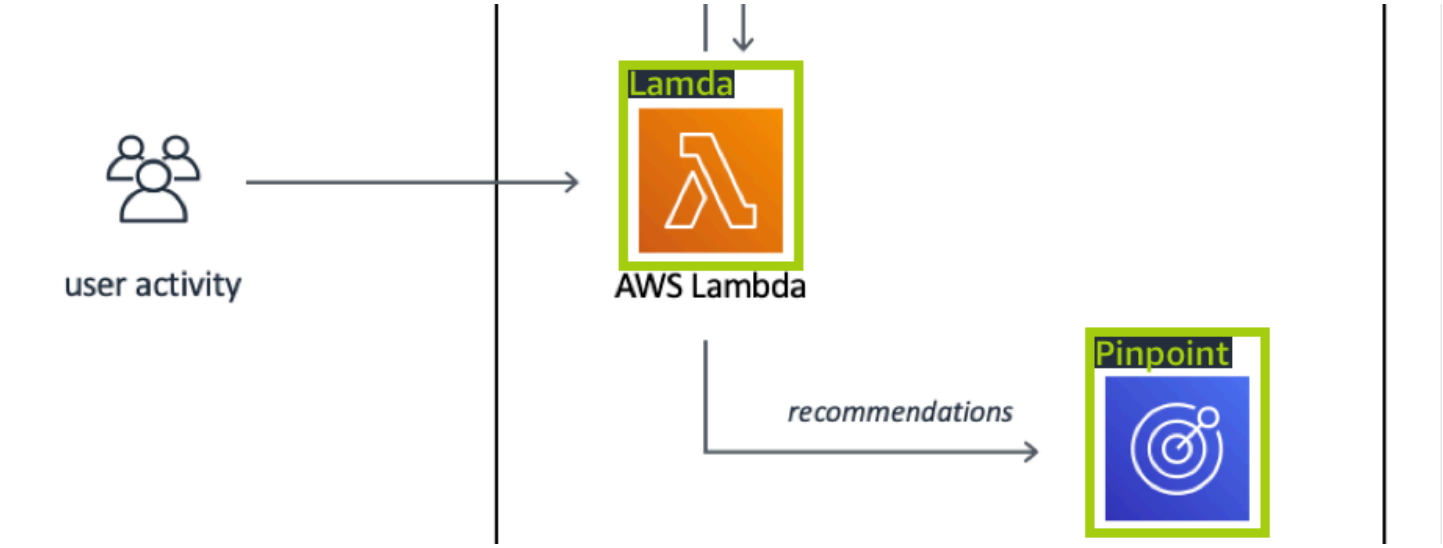
모델은 이미지 상의 객체 위치를 예측합니다. 예측에는 객체 위치에 대한 경계 상자 정보와 경계 상자 내의 객체를 식별하는 레이블이 포함됩니다. 예를 들어, 다음 이미지는 콤퍼레이터 또는 포트 저항기와 같은 회로 기판의 다양한 부분 주위에 있는 경계 상자를 보여줍니다.



객체 위치 파악 예제 프로젝트는 Amazon Rekognition Custom Labels가 레이블이 있는 경계 상자를 사용하여 객체 위치를 찾는 모델을 훈련하는 방법을 보여줍니다.

브랜드 위치 찾기

Amazon Rekognition Custom Labels는 이미지에서 브랜드 위치(예: 로고)를 찾는 모델을 훈련할 수 있습니다. 예측에는 브랜드 위치에 대한 경계 상자 정보와 경계 상자 내의 객체를 식별하는 레이블이 포함됩니다. 예제 프로젝트에 관해서는 브랜드 감지 항목을 참조하세요. 다음 이미지는 모델이 감지할 수 있는 일부 브랜드의 예입니다.



모델 생성

모델을 만드는 단계는 프로젝트 생성, 훈련 및 테스트 데이터 세트 생성, 모델 훈련입니다.

프로젝트 생성

Amazon Rekognition Custom Labels 프로젝트는 모델을 생성하고 관리하는 데 필요한 리소스의 모음입니다. 프로젝트는 다음을 관리합니다.

- 데이터 세트: 모델 훈련에 사용되는 이미지 및 이미지 레이블. 프로젝트에는 훈련 데이터 세트와 테스트 데이터 세트가 있습니다.
- 모델: 사용자의 비즈니스에 필요한 개념, 장면, 객체를 찾을 수 있게 훈련하는 소프트웨어입니다. 한 프로젝트에 여러 버전의 모델을 넣어 둘 수 있습니다.

하나의 프로젝트에는 하나의 용도만 지정해서 사용하시는 것을 권장합니다. 예를 들어 회로판에서 회로판 부품을 찾는 용도로만 사용하는 프로젝트처럼 말입니다.

Amazon Rekognition 사용자 지정 라벨 콘솔과 API를 사용하여 프로젝트를 생성할 수 있습니다. CreateProject 자세한 정보는 프로젝트 생성을 참조하세요.

훈련 및 테스트 데이터 세트 생성

데이터 세트는 해당 이미지를 설명하는 이미지와 레이블의 집합입니다. 프로젝트 내에서 Amazon Rekognition Custom Labels가 모델을 훈련하고 테스트하는 데 사용하는 교육 데이터 세트와 테스트 데이터 세트를 생성합니다.

레이블은 이미지에서 객체, 장면, 개념 또는 객체 주위의 경계 상자를 식별합니다. 레이블은 전체 이미지(이미지 수준)에 지정되거나 이미지에서 객체를 둘러싸는 경계 상자에 지정됩니다.

> **Important**
>
> 데이터 세트의 이미지에 레이블을 지정하는 방식에 따라 Amazon Rekognition Custom Labels가 생성하는 모델 유형이 결정됩니다. 예를 들어 객체, 장면 및 개념을 찾는 모델을 훈련하려면 훈련 및 테스트 데이터 세트의 이미지에 이미지 수준 레이블을 지정합니다. 자세한 정보는 데이터 세트 목적 설정을 참조하세요.

이미지는 PNG 및 JPEG 형식이어야 하며 입력 이미지 권장 사항을 따라야 합니다. 자세한 정보는 이미지 준비을 참조하세요.

훈련 및 테스트 데이터 세트 생성(콘솔)

단일 데이터 세트 또는 별도의 훈련 및 테스트 데이터 세트로 프로젝트를 시작할 수 있습니다. 단일 데이터 세트로 시작하는 경우 Amazon Rekognition Custom Labels는 훈련 중에 데이터 세트를 분할하여 프로젝트에 사용할 훈련 데이터 세트(80%)와 테스트 데이터 세트(20%)를 생성합니다. Amazon Rekognition Custom Labels가 훈련 및 테스트에 사용할 이미지를 결정하게 하려면 단일 데이터 세트로 시작하세요. 훈련, 테스트 및 성능 튜닝을 완벽하게 제어하려면 별도의 훈련 및 테스트 데이터 세트로 프로젝트를 시작하는 것이 좋습니다.

프로젝트의 데이터 세트를 만들려면 다음 방법 중 하나로 이미지를 가져옵니다.

- 로컬 컴퓨터에서 이미지 가져오기
- S3 버킷에서 이미지 가져오기 Amazon Rekognition Custom Labels는 이미지가 포함된 폴더 이름을 사용하여 이미지에 레이블을 지정할 수 있습니다.
- Amazon SageMaker Ground Truth 매니페스트 파일을 가져옵니다.
- 기존 Amazon Rekognition Custom Labels 데이터 세트를 복사합니다.

자세한 정보는 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성을 참조하세요.

이미지를 가져온 위치에 따라 이미지에 레이블이 지정되지 않을 수 있습니다. 예를 들어 로컬 컴퓨터에서 가져온 이미지에는 레이블이 지정되지 않습니다. Amazon SageMaker Ground Truth 매니페스트 파일에서 가져온 이미지에는 레이블이 지정됩니다. Amazon Rekognition Custom Labels 콘솔을 사용하여 레이블을 추가, 변경 및 할당할 수 있습니다. 자세한 정보는 이미지 레이블 지정을 참조하세요.

콘솔을 사용하여 훈련 및 테스트 데이터 세트를 생성하려면 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성 항목을 참조하세요. 훈련 및 테스트 데이터 세트 생성이 포함된 튜토리얼에 관해서는 튜토리얼: 이미지 분류 항목을 참조하세요.

훈련 및 테스트 데이터 세트 생성(SDK)

훈련 및 테스트 데이터 세트를 만들려면 `CreateDataset` API를 사용합니다. Amazon Sagemaker 형식 매니페스트 파일을 사용하거나 기존 Amazon Rekognition Custom Labels 데이터세트를 복사하여 데이터 세트를 생성할 수 있습니다. 자세한 내용은 훈련 및 테스트 데이터 세트 생성(SDK) 섹션을 참조하세요. 필요한 경우 직접 매니페스트 파일을 생성할 수 있습니다. 자세한 정보는 the section called "매니페스트 파일 생성"을 참조하세요.

모델 훈련하기

훈련 데이터 세트를 사용하여 모델을 훈련하세요. 모델을 훈련할 때마다 새 버전의 모델이 생성됩니다. Amazon Rekognition Custom Labels는 훈련 중에 훈련된 모델의 성능을 테스트합니다. 그 결과를 사용하여 모델을 평가하고 개선할 수 있습니다. 훈련을 완료하는 데 시간이 걸립니다. 모델 훈련을 성공적으로 완료한 경우에만 비용이 청구됩니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 훈련을 참조하세요. 모델 훈련이 실패할 경우 Amazon Rekognition Custom Labels는 사용할 수 있는 디버깅 정보를 제공합니다. 자세한 정보는 실패한 모델 훈련 디버깅을 참조하세요.

모델 훈련(콘솔)

콘솔을 사용하여 모델을 훈련하려면 모델 훈련(콘솔) 항목을 참조하세요.

모델 훈련(SDK)

버전을 호출하여 Amazon Rekognition 사용자 지정 레이블 모델을 학습시킵니다. CreateProject 자세한 정보는 모델 훈련(SDK)을 참조하세요.

모델 개선

Amazon Rekognition Custom Labels는 테스트 중에 훈련된 모델을 개선하는 데 사용할 수 있는 평가 지표를 생성합니다.

모델 평가

테스트 중에 만든 성능 지표를 사용하여 모델의 성능을 평가하세요. F1, 정밀도, 재현율과 같은 성능 지표를 통해 훈련된 모델의 성능을 이해하고 프로덕션에 사용할 준비가 되었는지 결정할 수 있습니다. 자세한 정보는 모델 평가를 위한 지표을 참조하세요.

모델 평가(콘솔)

성능 지표를 보려면 평가 지표 액세스(콘솔) 항목을 참조하세요.

모델 평가(SDK)

성능 지표를 가져오려면 Versions를 호출하여 DescribeProject테스트 결과를 얻어야 합니다. 자세한 정보는 Amazon Rekognition Custom Labels 평가 지표에 액세스(SDK)을 참조하세요. 테스트 결과에는 콘솔에서는 이용할 수 없는 지표(예: 분류 결과에 대한 오차 행렬)가 포함됩니다. 테스트 결과는 다음과 같은 형식으로 반환됩니다.

- F1 점수: 모델의 정밀도 및 재현율의 전반적인 성능을 나타내는 단일 값입니다. 자세한 정보는 F1을 참조하세요.
- 요약 파일 위치: 테스트 개요에는 전체 테스트 데이터 세트에 대한 집계된 평가 지표와 각 개별 레이블에 대한 지표가 포함됩니다. `DescribeProjectVersions`는 개요 파일의 S3 버킷 및 폴더 위치를 반환합니다. 자세한 정보는 요약 파일을 참조하세요.
- 평가 매니페스트 스냅샷 위치: 스냅샷에는 신뢰도 등급 및 바이너리 분류 테스트 결과(예: 오탐지)를 비롯한 테스트 결과에 대한 세부 정보가 포함됩니다. `DescribeProjectVersions`는 스냅샷 파일의 S3 버킷 및 폴더 위치를 반환합니다. 자세한 정보는 평가 매니페스트 스냅샷을 참조하세요.

모델 개선

개선이 필요한 경우 훈련 이미지를 더 추가하거나 데이터 세트 레이블 지정을 개선할 수 있습니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 개선을 참조하세요. 또한 모델의 예측에 대한 피드백을 제공하고 이를 사용하여 모델을 개선할 수 있습니다. 자세한 정보는 모델 피드백 솔루션을 참조하세요.

모델 개선(콘솔)

데이터 세트에 이미지를 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요. 레이블을 추가하거나 변경하려면 the section called “이미지 레이블 지정” 항목을 참조하세요.

모델을 재훈련하려면 모델 훈련(콘솔) 항목을 참조하세요.

모델 개선(SDK)

데이터 세트에 이미지를 추가하거나 이미지의 레이블을 변경하려면 `UpdateDatasetEntries` API를 사용하세요. `UpdateDatasetEntries`는 매니페스트 파일에 JSON 라인을 업데이트하거나 추가

합니다. 각 JSON 라인에는 지정된 레이블 또는 경계 상자 정보와 같은 단일 이미지에 대한 정보가 들어 있습니다. 자세한 정보는 더 많은 이미지 추가(SDK)을 참조하세요. 데이터 세트의 항목을 보려면 `ListDatasetEntries` API를 사용하세요.

모델을 재훈련하려면 모델 훈련(SDK) 항목을 참조하세요.

모델 시작

모델을 사용하려면 먼저 Amazon Rekognition Custom Labels 콘솔 또는 `StartProjectVersion` API를 사용하여 모델을 시작하세요. 모델을 실행하는 시간만큼 요금이 부과됩니다. 자세한 정보는 훈련된 모델 실행을 참조하세요.

모델 시작(콘솔)

콘솔을 사용하여 모델을 시작하려면 Amazon Rekognition Custom Labels 모델 시작(콘솔) 항목을 참조하세요.

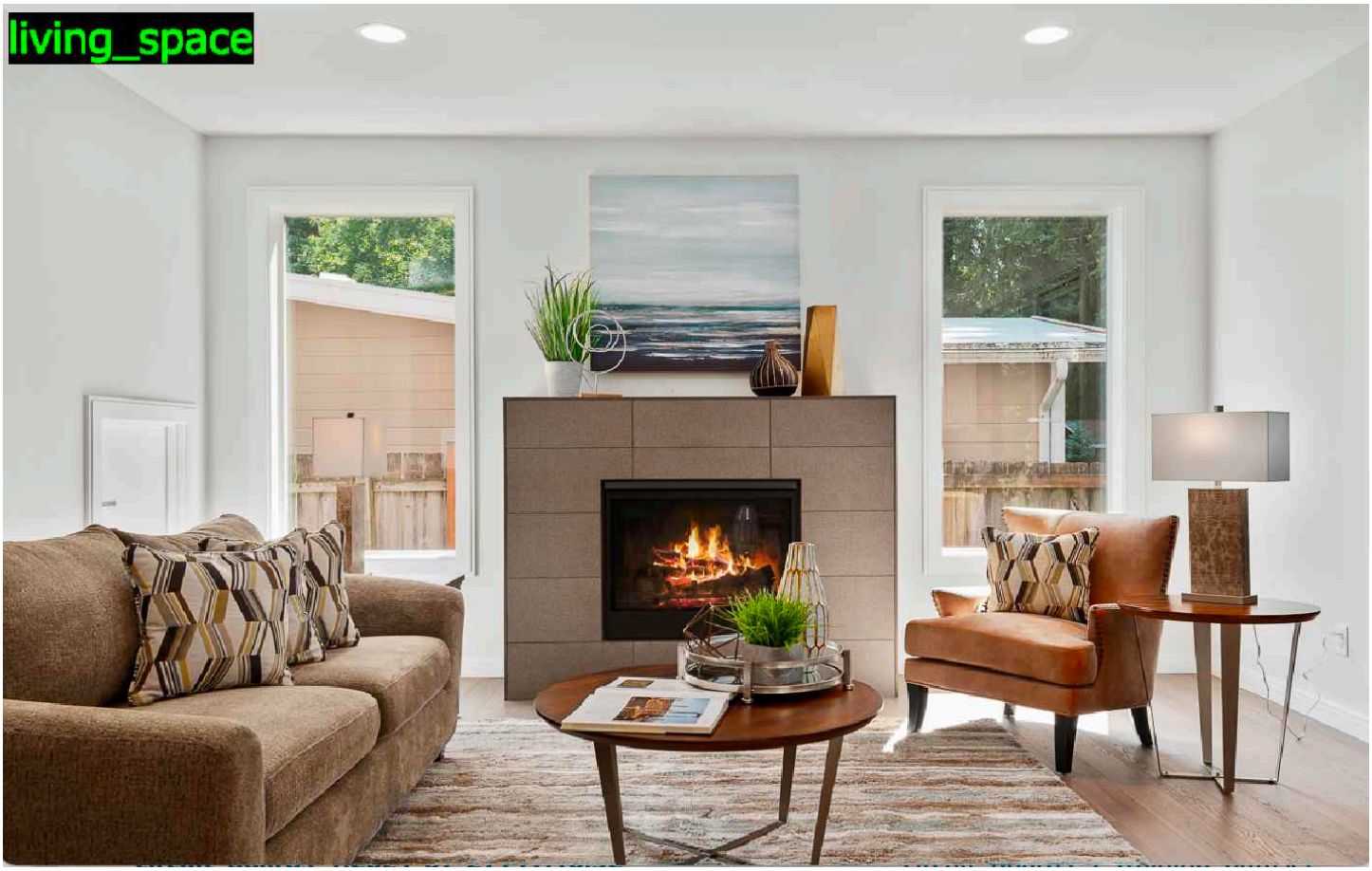
모델 시작

StartProjectVersion을 호출하여 모델을 시작합니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 시작(SDK)을 참조하세요.

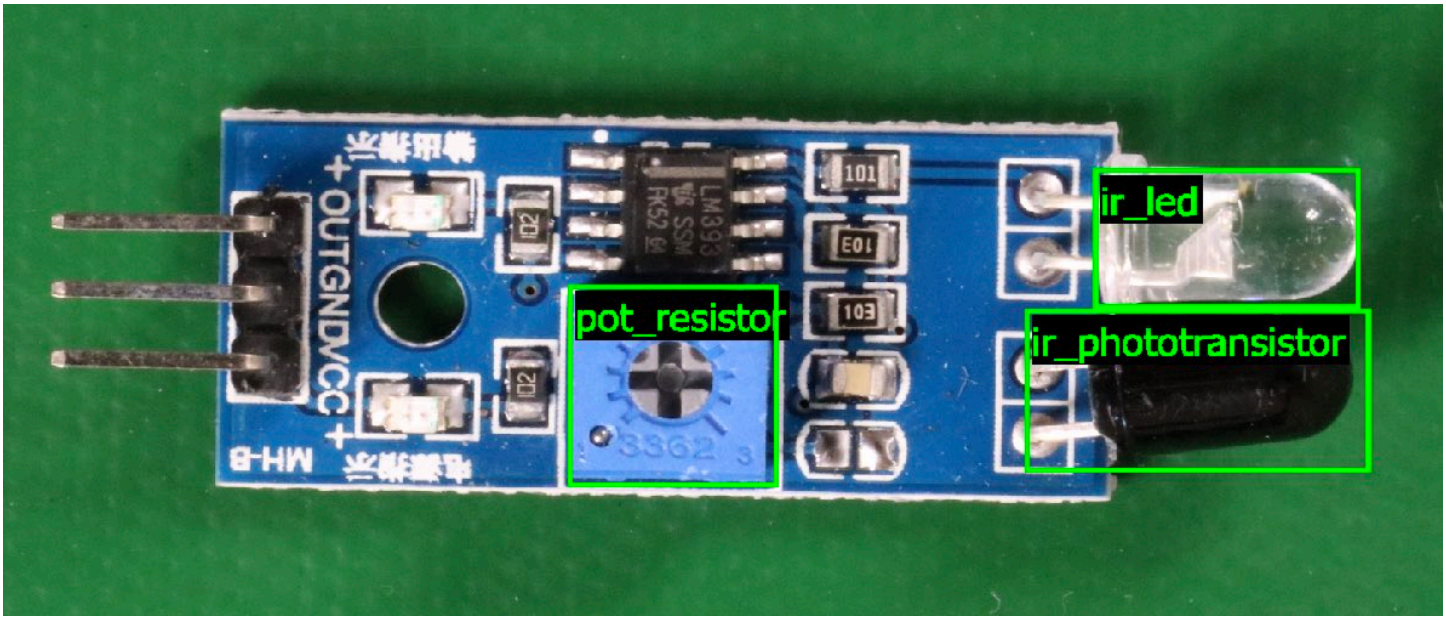
이미지 분석

모델로 이미지를 분석하려면 `DetectCustomLabels` API를 사용합니다. 로컬 이미지 또는 S3 버킷에 저장된 이미지를 지정할 수 있습니다. 작업을 수행하려면 사용하려는 모델의 Amazon 리소스 이름(ARN)도 필요합니다.

모델이 객체, 장면, 개념을 찾은 경우 응답에는 이미지에서 찾은 이미지 수준 레이블 목록이 포함됩니다. 예를 들어 다음 이미지는 방 예제 프로젝트를 사용하여 찾은 이미지 수준 레이블을 보여줍니다.



모델이 객체 위치를 찾은 경우 응답에는 이미지에서 찾은 레이블이 지정된 경계 상자 목록이 포함됩니다. 경계 상자는 이미지에서 객체의 위치를 나타냅니다. 테두리 상자 정보를 사용하여 객체 주위에 테두리 상자를 그릴 수 있습니다. 예를 들어, 다음 이미지는 회로판 예제 프로젝트를 사용하여 찾은 회로판 부품 주위의 경계 상자를 보여줍니다.



자세한 정보는 훈련된 모델을 사용한 이미지 분석을 참조하세요.

모델 중지

모델을 실행하는 시간만큼 요금이 부과됩니다. 모델을 더 이상 사용하지 않는 경우 Amazon Rekognition Custom Labels 콘솔을 사용하거나 `StopProjectVersion` API를 사용하여 모델을 중지하세요. 자세한 정보는 Amazon Rekognition Custom Labels 모델 중지을 참조하세요.

모델 중지(콘솔)

콘솔에서 실행 중인 모델을 중지하려면 Amazon Rekognition Custom Labels 모델 중지(콘솔) 항목을 참조하세요.

모델 중지(SDK)

실행 중인 모델을 중지하려면 Version을 호출하십시오. StopProject 자세한 내용은 Amazon Rekognition Custom Labels 모델 중지(SDK)을(를) 참조하세요.

Amazon Rekognition Custom Labels 시작하기

시작하기 지침 전에 Amazon Rekognition Custom Labels의 이해 항목을 먼저 읽어 보는 것이 좋습니다.

Amazon Rekognition Custom Labels를 사용하여 기계 학습 모델을 교육합니다. 훈련된 모델은 이미지를 분석하여 사용자의 비즈니스에 필요한 객체, 장면 및 개념을 찾습니다. 예를 들어, 주택 이미지를 분류하도록 모델을 훈련하거나 인쇄 회로 기판에서 전자 부품의 위치를 찾을 수 있습니다.

Amazon Rekognition Custom Labels에는 사용자를 위한 튜토리얼 동영상과 예제 프로젝트가 포함되어 있습니다.

> **Note**
>
> Amazon Rekognition 사용자 지정 레이블이 지원하는 AWS 지역 및 엔드포인트에 대한 자세한 내용은 Rekognition 엔드포인트 및 할당량을 참조하십시오.

튜토리얼 동영상

이 동영상은 Amazon Rekognition Custom Labels를 사용하여 모델을 훈련하고 사용하는 방법을 보여줍니다.

튜토리얼 동영상을 보려면

1. 에 AWS Management Console 로그인하고 https://console.aws.amazon.com/rekognition/ 에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다. 사용자 지정 레이블 사용이 표시되지 않으면 Amazon Rekognition Custom Labels가 현재 계신 지역 AWS 지역에서 지원되는지 확인하세요.
3. 탐색 창에서 시작하기를 선택하세요.
4. Amazon Rekognition Custom Labels이란 무엇입니까?에서 동영상을 선택하여 개요 동영상을 시청하세요.
5. 탐색 창에서 튜토리얼을 선택합니다.
6. 튜토리얼 페이지에서 보고 싶은 튜토리얼 동영상을 선택합니다.

예제 프로젝트

Amazon Rekognition Custom Labels는 다음과 같은 예제 프로젝트를 제공합니다.

이미지 분류

이미지 분류 프로젝트(방)는 이미지에서 가구 위치를 하나 이상 찾는 모델(예: 뒷마당, 주방, 파티오)을 훈련합니다. 훈련 및 테스트 이미지는 하나의 위치를 나타냅니다. 각 이미지에는 주방, 파티오, 거실 같은 하나의 이미지 수준 레이블이 지정되어 있습니다. 분석된 이미지의 경우 훈련된 모델은 학습에 사용된 이미지 수준 레이블 집합에서 일치하는 레이블을 하나 이상 반환합니다. 예를 들어, 모델은 다음 이미지에서 거실이라는 레이블을 찾을 수 있습니다. 자세한 정보는 객체, 장면 및 개념 찾기을 참조하세요.

다중 레이블 이미지 분류

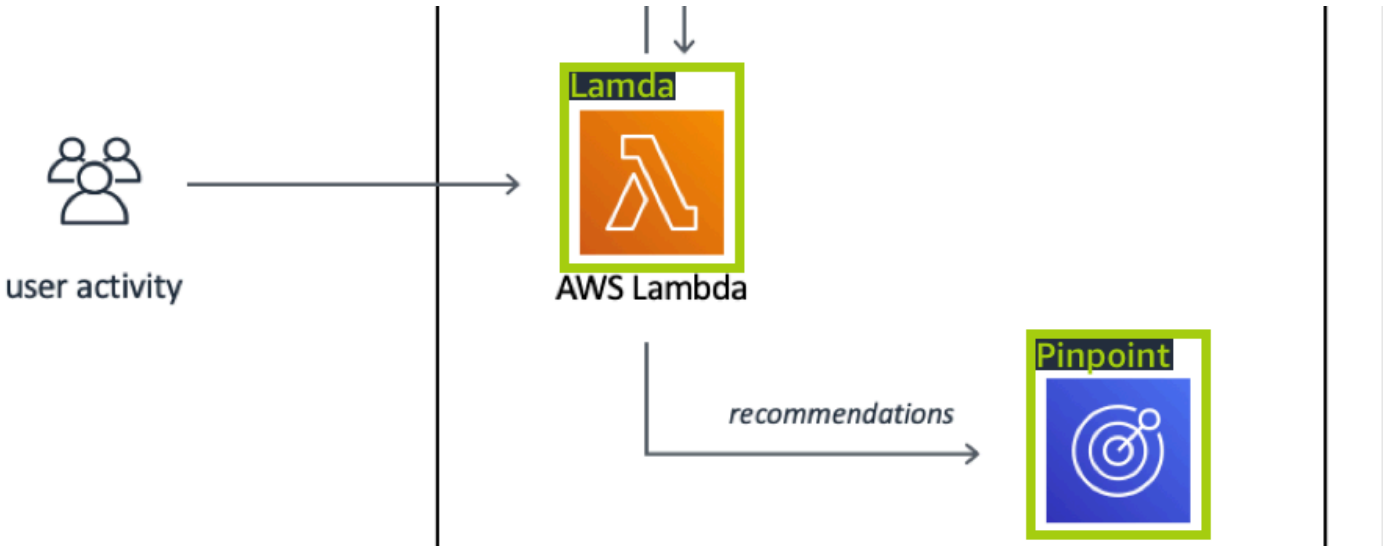
다중 레이블 이미지 분류 프로젝트(꽃)는 꽃 이미지를 세 가지 개념(꽃 유형, 잎 유무, 성장 단계)으로 분류하는 모델을 훈련합니다.

훈련 및 테스트 이미지에는 각 개념에 대한 이미지 수준 레이블이 있습니다. 예를 들어 꽃 유형에는 camellia, 잎이 있는 꽃의 경우 with_leaves, 완전히 자란 꽃의 경우 fully_grown이 있습니다.

분석된 이미지의 경우 훈련된 모델은 훈련에 사용된 이미지 수준 레이블 집합에서 일치하는 레이블을 반환합니다. 예를 들어, 모델은 다음 이미지에 대해 mediterranean_spurge 및 with_leaves라는 레이블을 반환합니다. 자세한 정보는 객체, 장면 및 개념 찾기을 참조하세요.

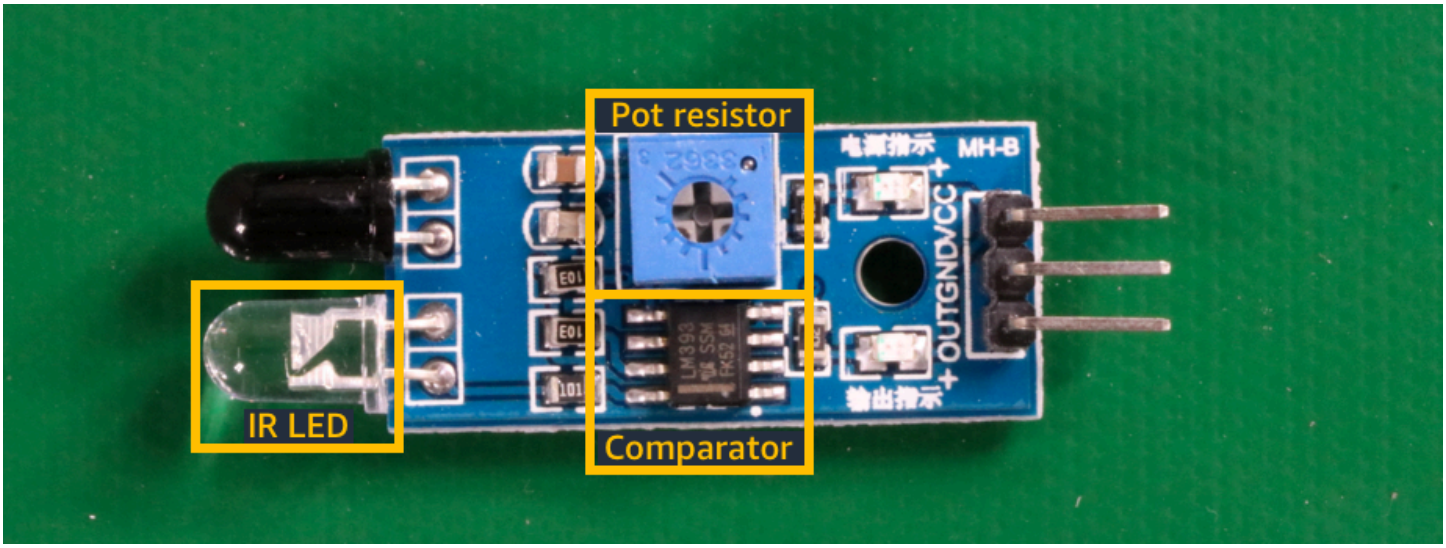
브랜드 감지

브랜드 감지 프로젝트 (Logos) 는 모델이 Amazon Textract와 AWS Lambda와 같은 특정 AWS 로고의 위치를 찾는 모델을 교육합니다. 훈련 이미지는 로고만을 대상으로 하며 lambda나 textract와 같은 하나의 이미지 수준 레이블이 있습니다. 브랜드 위치를 나타내는 경계 상자가 있는 훈련 이미지를 사용하여 브랜드 감지 모델을 훈련할 수도 있습니다. 테스트 이미지에는 아키텍처 다이어그램과 같이 자연스러운 위치에서의 로고 위치를 나타내는 레이블이 지정된 경계 상자가 있습니다. 훈련된 모델은 로고를 찾고 발견된 각 로고에 대해 레이블이 지정된 경계 상자를 반환합니다. 자세한 정보는 브랜드 위치 찾기을 참조하세요.



객체 위치 파악

객체 위치 파악 프로젝트(회로 기판)는 비교기 또는 적외선 발광 다이오드와 같은 인쇄 회로 기판에서 부품의 위치를 찾는 모델을 훈련합니다. 훈련 및 테스트 이미지에는 회로 기판 부품을 둘러싸는 경계 상자와 경계 상자 내의 부품을 식별하는 레이블이 포함되어 있습니다. 다음 예제 이미지에서 라벨 이름은 ir_포토트랜지스터, ir_led, pot_resistor, 콤퍼레이터입니다. 훈련된 모델은 회로 기판 부품을 찾아서 찾은 각 회로 부품에 대해 레이블이 지정된 경계를 반환합니다. 자세한 정보는 객체 위치 찾기을 참조하세요.



예제 프로젝트 사용

이 시작하기 지침은 Amazon Rekognition Custom Labels가 생성한 예제 프로젝트를 사용하여 모델을 훈련하는 방법을 보여줍니다. 또한 모델을 시작하고 이를 사용하여 이미지를 분석하는 방법도 보여줍니다.

예제 프로젝트 생성

시작하려면 사용할 프로젝트를 결정하세요. 자세한 내용은 1단계: 예제 프로젝트 선택 섹션을 참조하세요.

Amazon Rekognition Custom Labels는 데이터 세트를 사용하여 모델을 훈련 및 평가(테스트)합니다. 데이터 세트는 이미지와 이미지 내용을 식별하는 레이블을 관리합니다. 예제 프로젝트에는 모든 이미지에 레이블이 지정된 테스트 데이터 세트와 훈련 데이터 세트가 포함되어 있습니다. 모델을 훈련하기 전에 아무것도 변경할 필요가 없습니다. 예제 프로젝트는 Amazon Rekognition Custom Labels가 레이블을 사용하여 다양한 유형의 모델을 훈련하는 두 가지 방법을 보여줍니다.

- 이미지 수준: 레이블은 전체 이미지를 나타내는 객체, 장면 또는 개념을 식별합니다.
- 경계 상자: 레이블은 경계 상자의 내용을 식별합니다. 경계 상자는 이미지에서 객체를 둘러싸는 이미지 좌표의 집합입니다.

나중에 자체 이미지로 프로젝트를 만들 때는 훈련 및 테스트 데이터 세트를 만들고 이미지에 레이블도 지정해야 합니다. 자세한 내용은 모델 유형 결정 섹션을 참조하세요.

모델 훈련

Amazon Rekognition Custom Labels가 예제 프로젝트를 생성한 후 모델을 훈련할 수 있습니다. 자세한 내용은 2단계: 모델 훈련 섹션을 참조하세요. 일반적으로 훈련이 끝나면 모델의 성능을 평가합니다. 예제 데이터 세트의 이미지는 이미 고성능 모델을 생성하므로 모델을 실행하기 전에 모델을 평가할 필요가 없습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 섹션을 참조하세요.

모델 사용

다음으로 모델을 시작합니다. 자세한 내용은 3단계: 모델 시작 섹션을 참조하세요.

모델 실행을 시작한 후 모델을 사용하여 새 이미지를 분석할 수 있습니다. 자세한 내용은 4단계: 모델을 사용하여 이미지 분석 섹션을 참조하세요.

모델을 실행하는 시간만큼 요금이 부과됩니다. 예제 모델 사용을 마치면 모델을 중지해야 합니다. 자세한 내용은 5단계: 모델 중지 섹션을 참조하세요.

다음 단계

준비되었으면 프로젝트를 직접 만들 수 있습니다. 자세한 내용은 6단계: 다음 단계 섹션을 참조하세요.

1단계: 예제 프로젝트 선택

이 단계에서는 예제 프로젝트를 선택합니다. 그러면 Amazon Rekognition Custom Labels가 자동으로 프로젝트와 데이터 세트를 생성합니다. 프로젝트는 모델 훈련에 사용되는 파일을 관리합니다. 자세한 내용은 Amazon Rekognition Custom Labels 프로젝트 관리 섹션을 참조하세요. 데이터 세트에는 모델을 훈련하고 테스트하는 데 사용하는 이미지, 할당된 레이블, 경계 상자가 포함되어 있습니다. 자세한 내용은 the section called “데이터 세트 관리” 섹션을 참조하세요.

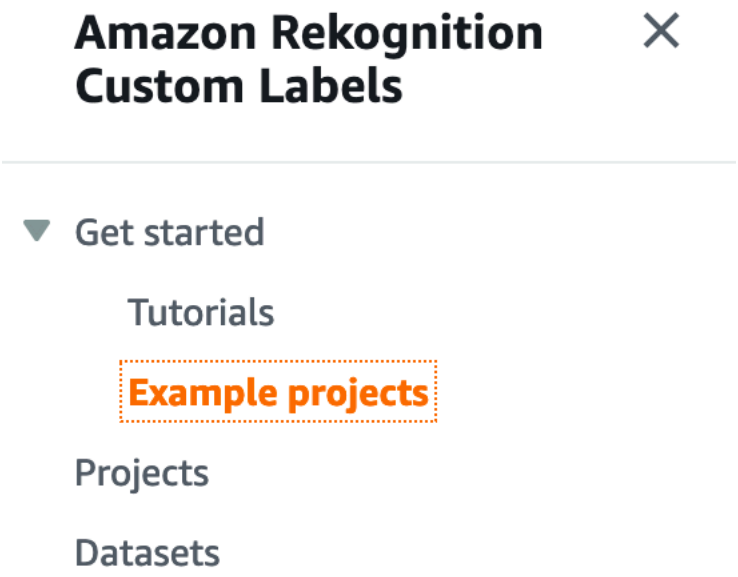
예제 프로젝트에 대한 자세한 내용은 예제 프로젝트 항목을 참조하세요.

예제 프로젝트 선택

1. 에 AWS Management Console 로그인하고 https://console.aws.amazon.com/rekognition/ 에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다. 사용자 지정 레이블 사용이 표시되지 않으면 Amazon Rekognition Custom Labels가 현재 계신 지역 AWS 지역에서 지원되는지 확인하세요.

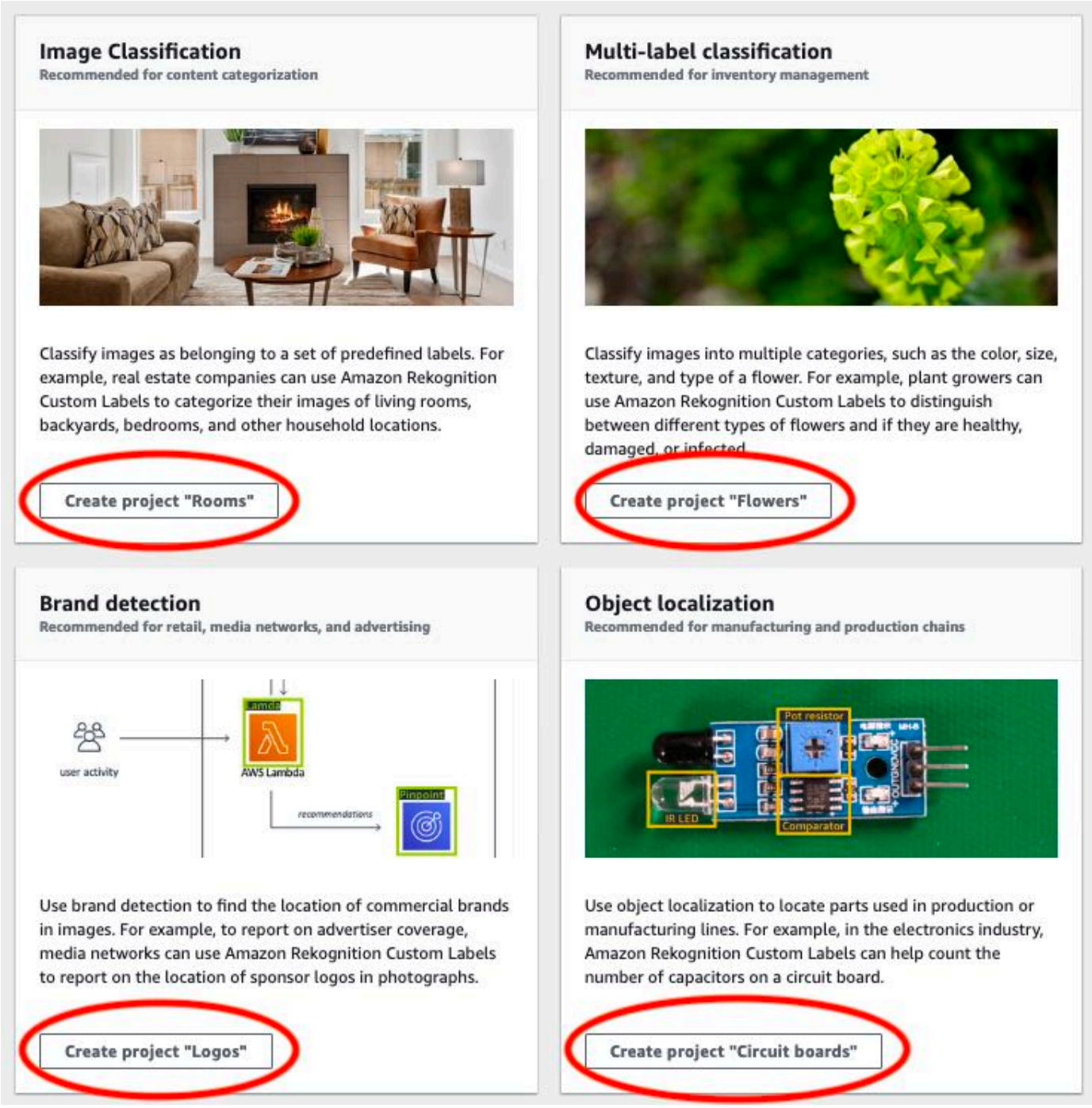
3. Get started를 선택합니다.

   시작하기, “예제 프로젝트”가 강조 표시된 자습서, 프로젝트 및 데이터세트를 보여주는 Amazon Rekognition 사용자 지정 레이블 섹션.

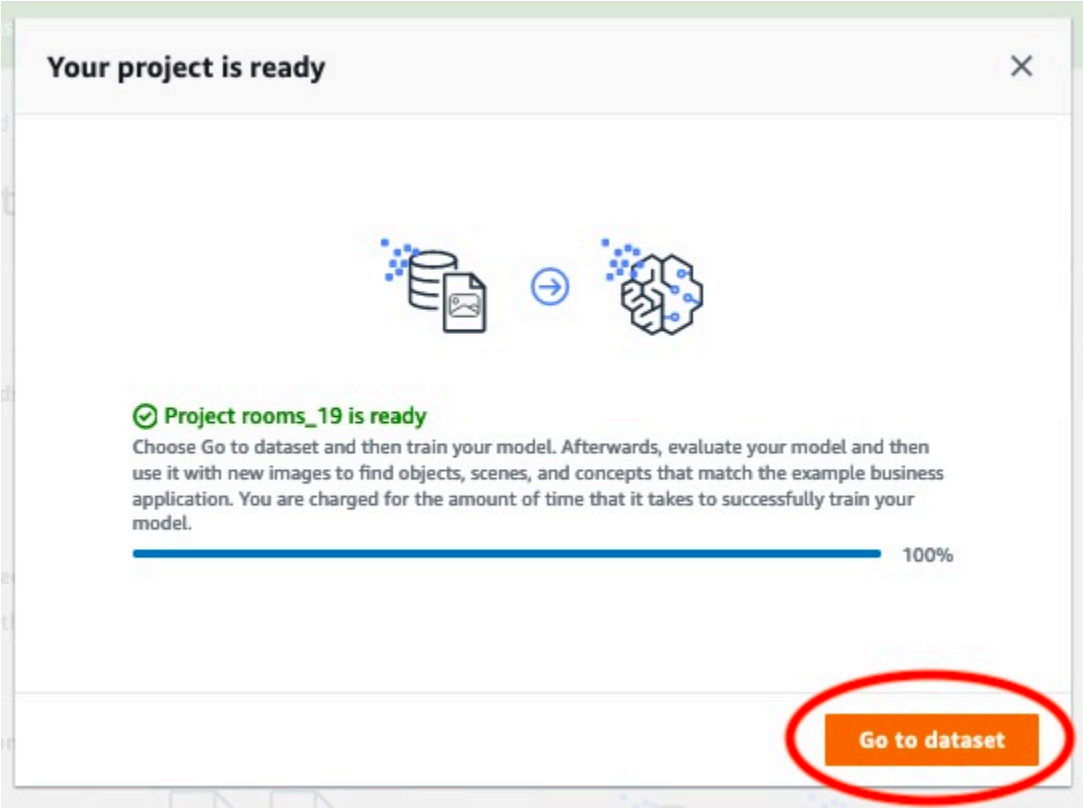


4. 예제 프로젝트 탐색에서 예제 프로젝트 체험을 선택합니다.
5. 사용할 프로젝트를 결정하고 예제 항목에서 “**#### ##**” 프로젝트 만들기를 선택합니다. 그러면 Amazon Rekognition Custom Labels가 예제 프로젝트를 생성합니다.

> **Note**
>
> 현재 AWS 지역에서 콘솔을 처음 연 경우 최초 설정 대화 상자가 표시됩니다. 다음을 따릅니다.
>
> 1. 표시된 Amazon S3 버킷의 이름을 기록해 둡니다.
> 2. 계속을 선택하여 Amazon Rekognition Custom Labels가 사용자를 대신하여 Amazon S3 버킷(콘솔 버킷)을 생성하게 하세요. 아래 콘솔 이미지는 이미지 분류 (방), 다중 라벨 분류 (꽃), 브랜드 감지 (로고), 개체 로컬라이제이션 (회로판) 을 위한 “프로젝트 만들기” 버튼이 있는 예를 보여줍니다.



6. 프로젝트가 준비되면 데이터 세트로 이동을 선택합니다. 다음 이미지는 프로젝트가 준비되었을 때 프로젝트 패널이 어떻게 보이는지 보여줍니다.

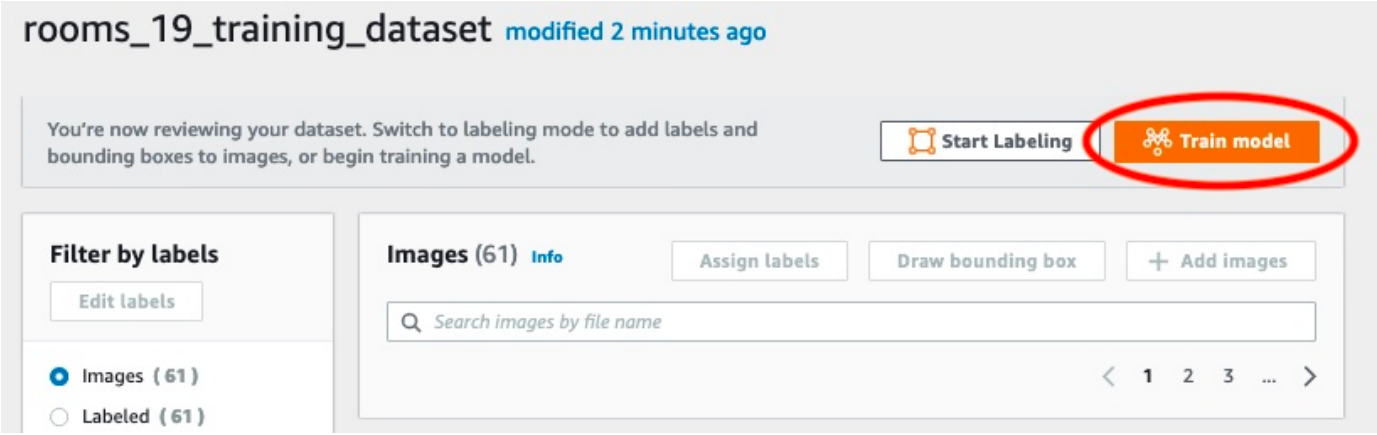


2단계: 모델 훈련

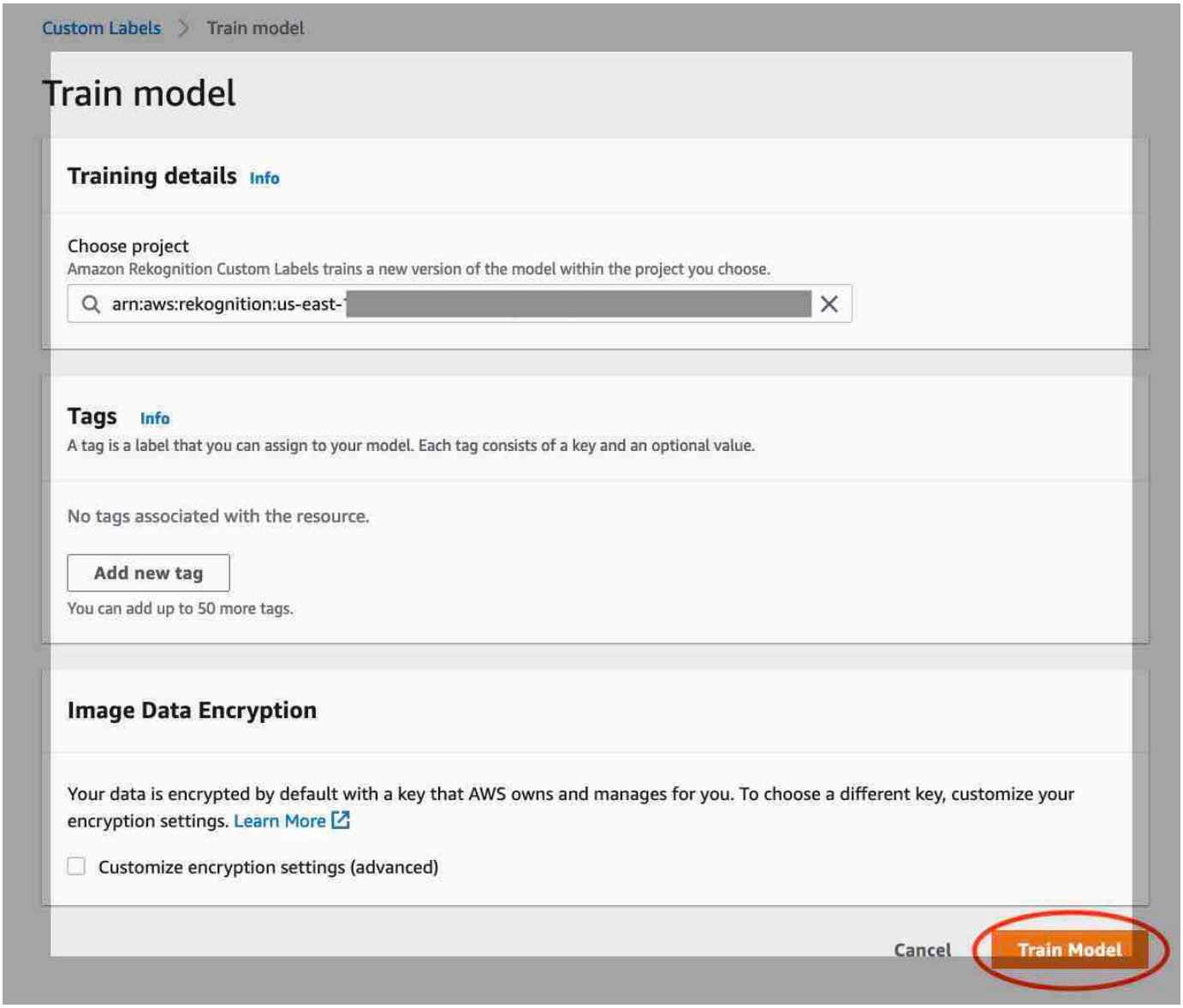
이 단계에서는 모델을 훈련합니다. 훈련 및 테스트 데이터 세트는 자동으로 구성됩니다. 훈련이 성공적으로 완료되면 전체 평가 결과와 개별 테스트 이미지에 대한 평가 결과를 볼 수 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 훈련 섹션을 참조하세요.

모델을 훈련하려면

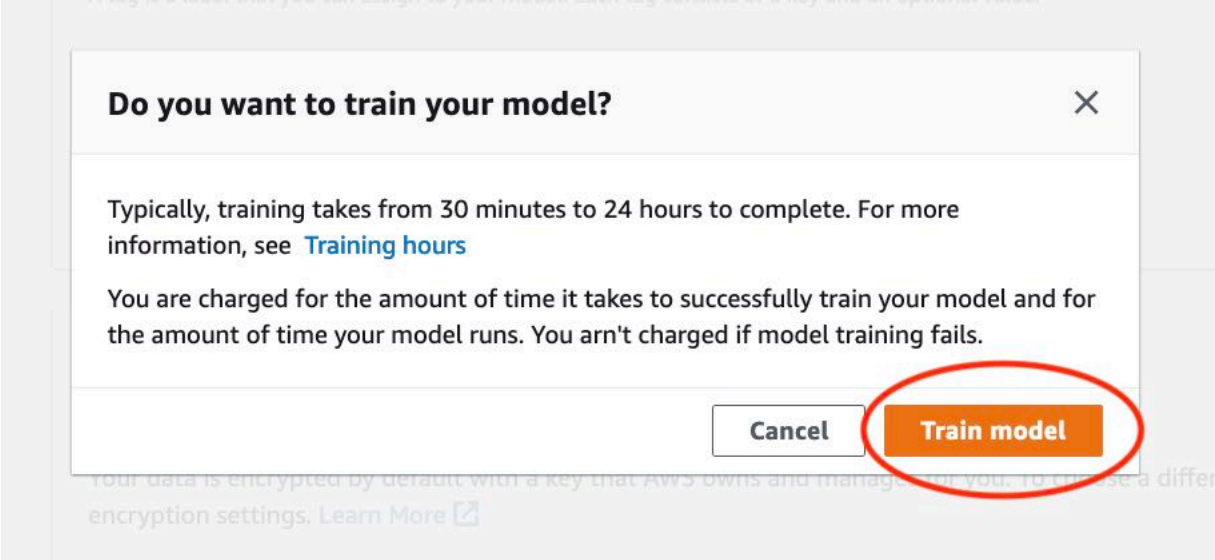
1. 데이터세트 페이지에서 Train 모델을 선택합니다. 다음 이미지는 기차 모델 버튼이 있는 콘솔을 보여줍니다.



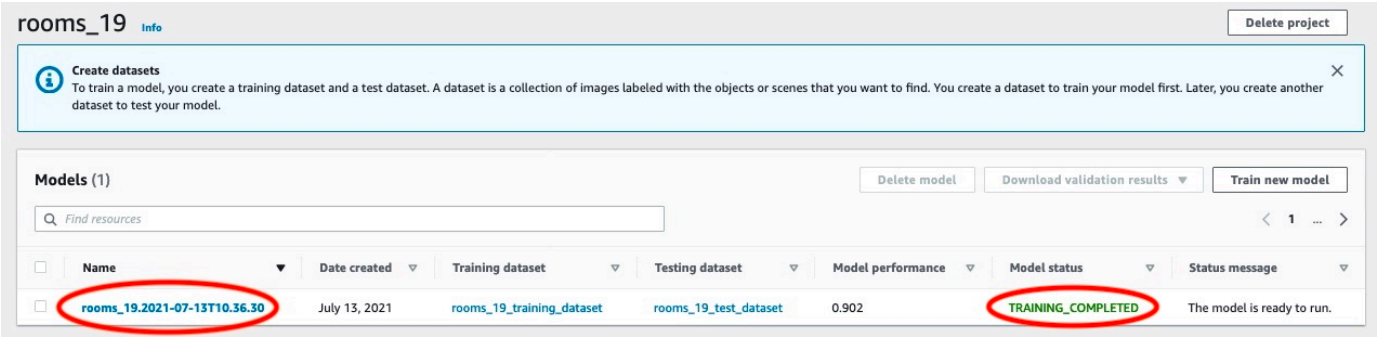
2. 모델 훈련 페이지에서 모델 훈련을 선택합니다. 아래 이미지는 모델 학습 버튼을 보여줍니다. 프로젝트의 Amazon 리소스 이름 (ARN) 이 프로젝트 선택 편집 상자에 있는 것을 확인할 수 있습니다.



3. 모델을 학습시키고 싶으신가요? 에서 다음 이미지에 표시된 대화 상자에서 Train model을 선택합니다.

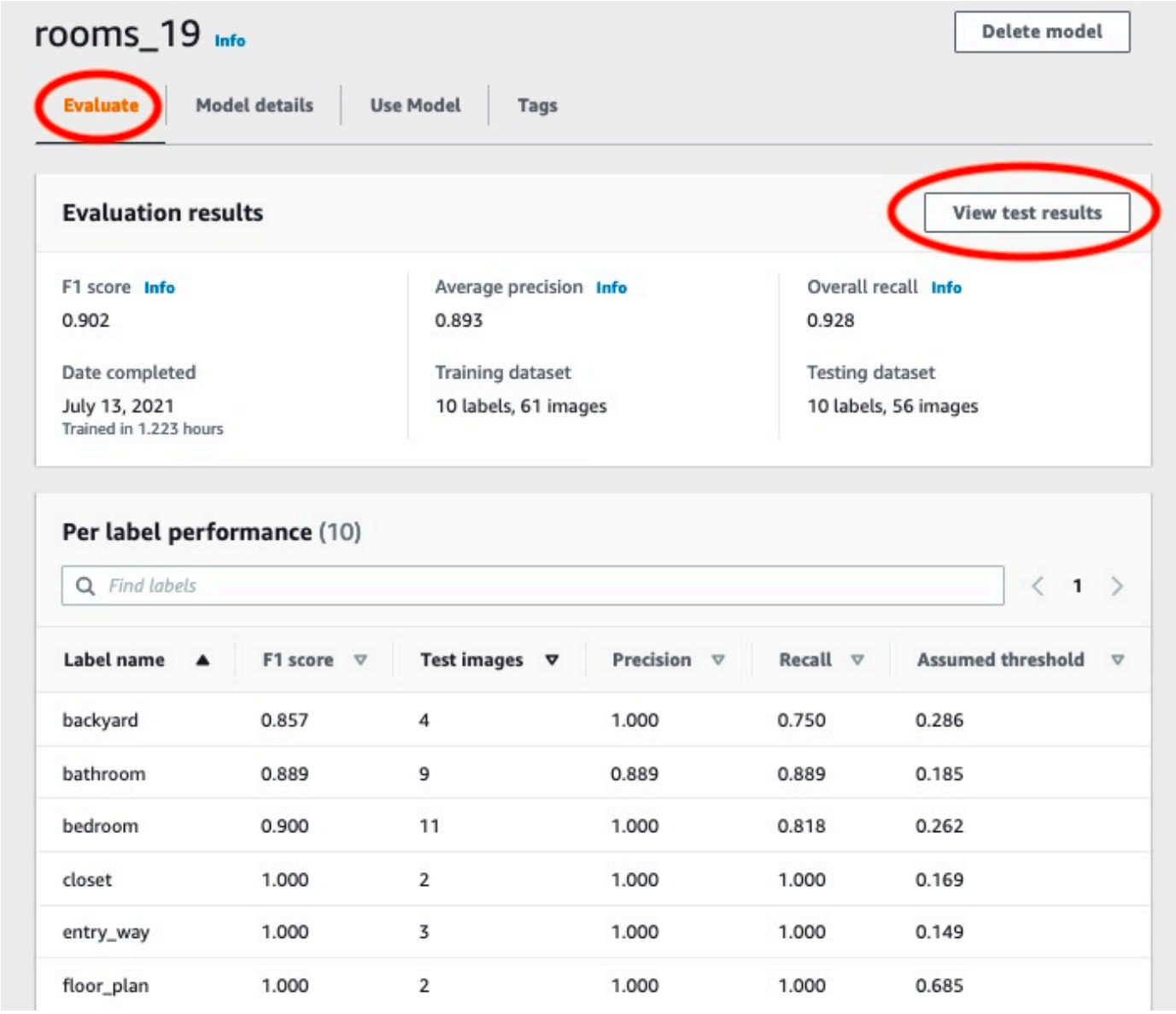


4. 훈련이 완료되면 모델 이름을 선택합니다. 다음 콘솔 스크린샷에서 볼 수 있듯이 모델 상태가 TRAINING_COMPLETED 일 때 학습이 완료됩니다.

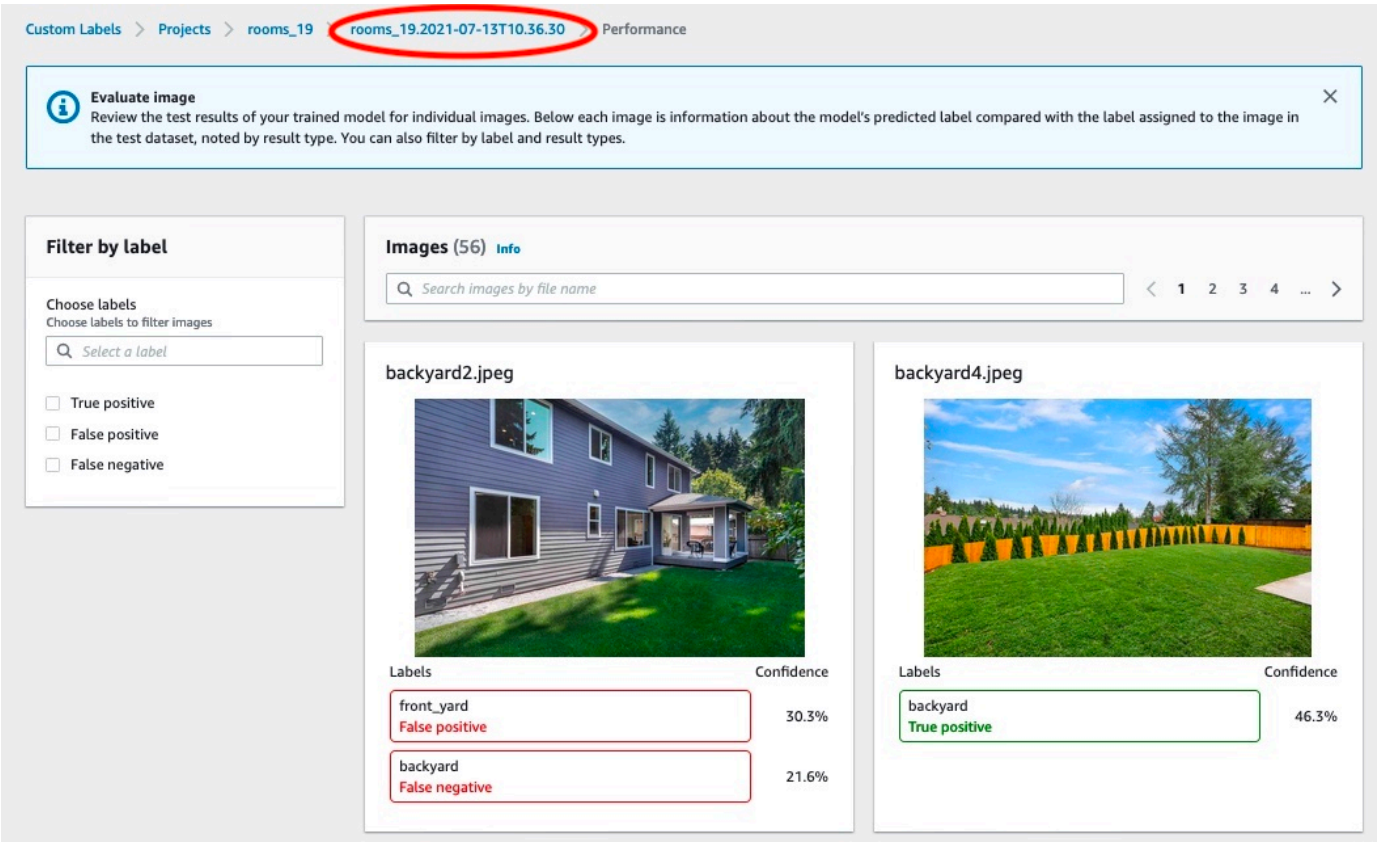


5. 평가 버튼을 선택하면 평가 결과를 볼 수 있습니다. 모델 평가에 대한 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 항목을 참조하세요.

6. 개별 테스트 이미지의 결과를 보려면 테스트 결과 보기를 선택합니다. 다음 스크린샷에서 볼 수 있듯이 평가 대시보드에는 각 라벨에 대한 F1 점수, 정밀도, 재현율 등의 지표가 테스트 이미지 수와 함께 표시됩니다. 평균, 정밀도, 재현율과 같은 전체 지표도 표시됩니다.



7. 테스트 결과를 확인한 후 모델 이름을 선택하여 모델 페이지로 돌아가세요. 다음은 성능 대시보드의 스크린샷입니다. 클릭하면 모델 페이지로 돌아갈 수 있습니다.



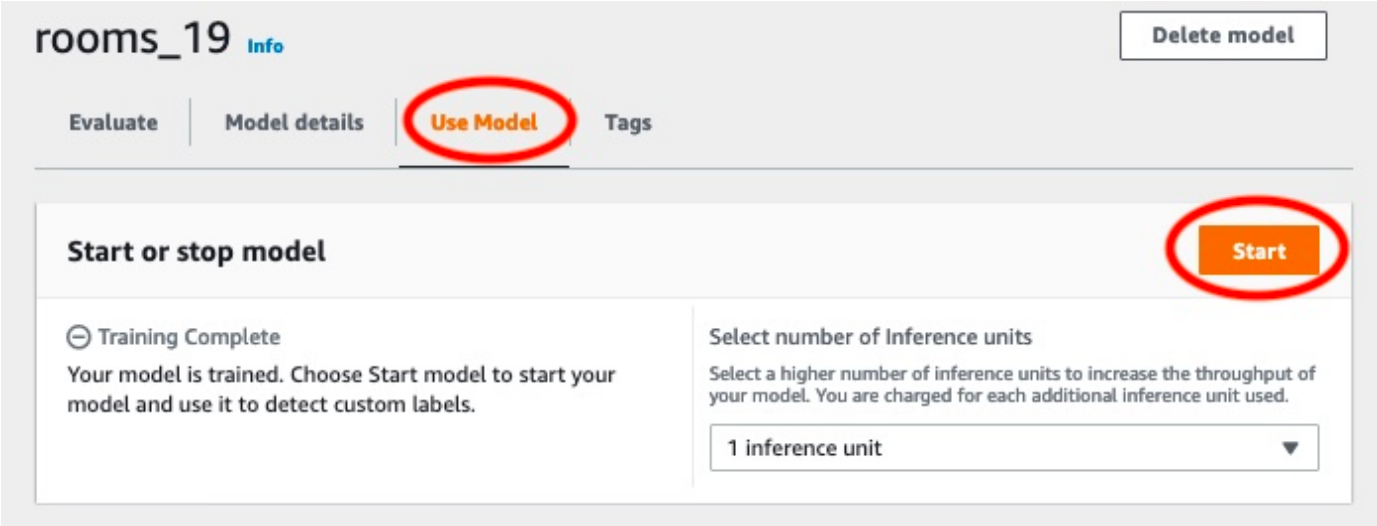
3단계: 모델 시작

이 단계에서는 모델을 시작합니다. 모델이 시작되면 모델을 사용하여 이미지를 분석할 수 있습니다.

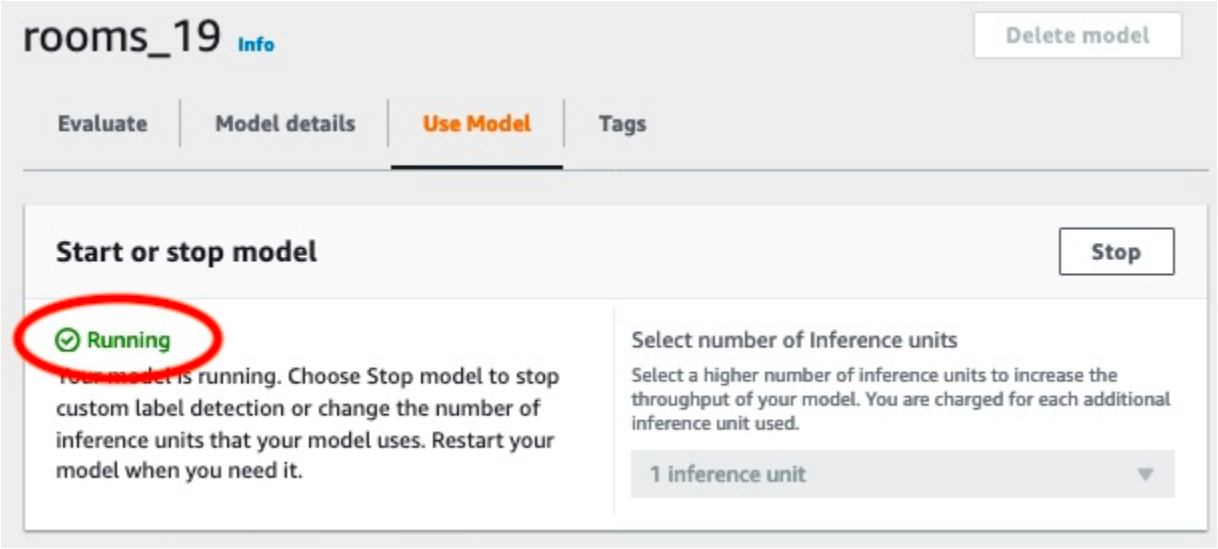
모델을 실행하는 시간만큼 요금이 부과됩니다. 이미지를 분석할 필요가 없는 경우 모델을 중지하세요. 나중에 모델을 다시 시작할 수 있습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 실행 섹션을 참조하세요.

모델을 시작하려면

1. 모델 페이지에서 모델 사용 탭을 선택합니다.
2. 모델 시작 또는 중지 항목에서 다음을 수행하세요.
   a. 시작을 선택합니다.
   b. 모델 시작 대화 상자에서 시작을 선택합니다. 다음 이미지는 모델 제어판의 시작 버튼을 보여 줍니다.



3. 모델이 실행될 때까지 기다리세요. 다음 스크린샷은 모델이 실행 중인 동안의 콘솔을 보여줍니다. 여기서 모델 시작 또는 중지 섹션의 상태는 Running 입니다.



4. 모델을 사용하여 이미지를 분류합니다. 자세한 내용은 4단계: 모델을 사용하여 이미지 분석 섹션을 참조하세요.

4단계: 모델을 사용하여 이미지 분석

DetectCustom레이블 API를 호출하여 이미지를 분석합니다. 이 단계에서는 `detect-custom-labels` AWS Command Line Interface (AWS CLI) 명령을 사용하여 예제 이미지를 분석합니다. Amazon Rekognition 사용자 지정 라벨 콘솔에서 AWS CLI 명령을 받을 수 있습니다. 콘솔은 모델을 사

용하도록 AWS CLI 명령을 구성합니다. 사용자는 Amazon S3 버킷에 저장된 이미지만 제공하면 됩니다. 이 주제는 각 예제 프로젝트에 사용할 수 있는 이미지를 제공합니다.

> **Note**
>
> 콘솔은 Python 예제 코드도 제공합니다.

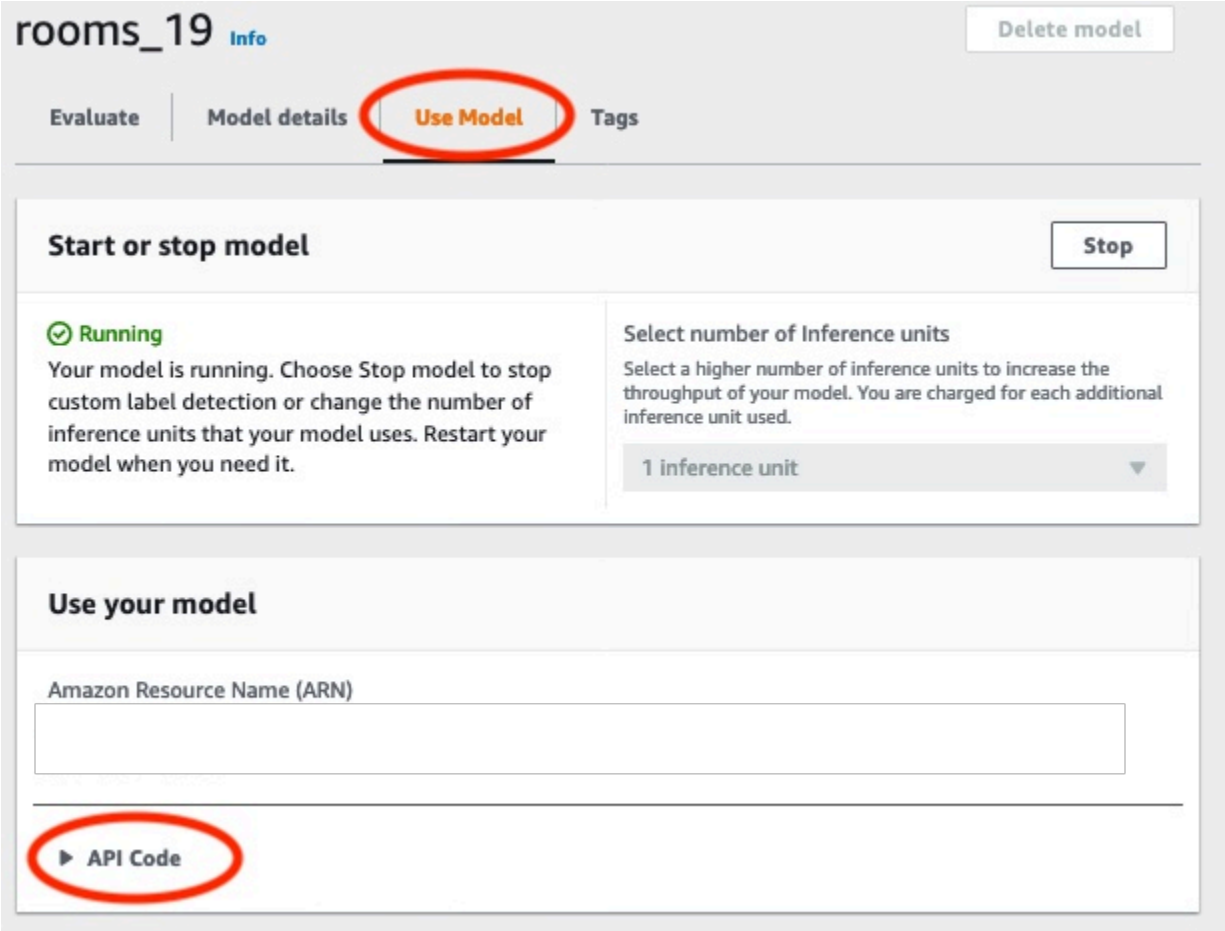
`detect-custom-labels`의 출력에는 이미지에서 찾은 레이블 목록, 경계 상자(모델이 객체 위치를 찾는 경우), 예측 정확도에 대한 모델의 신뢰도가 포함됩니다.

자세한 내용은 훈련된 모델을 사용한 이미지 분석 섹션을 참조하세요.

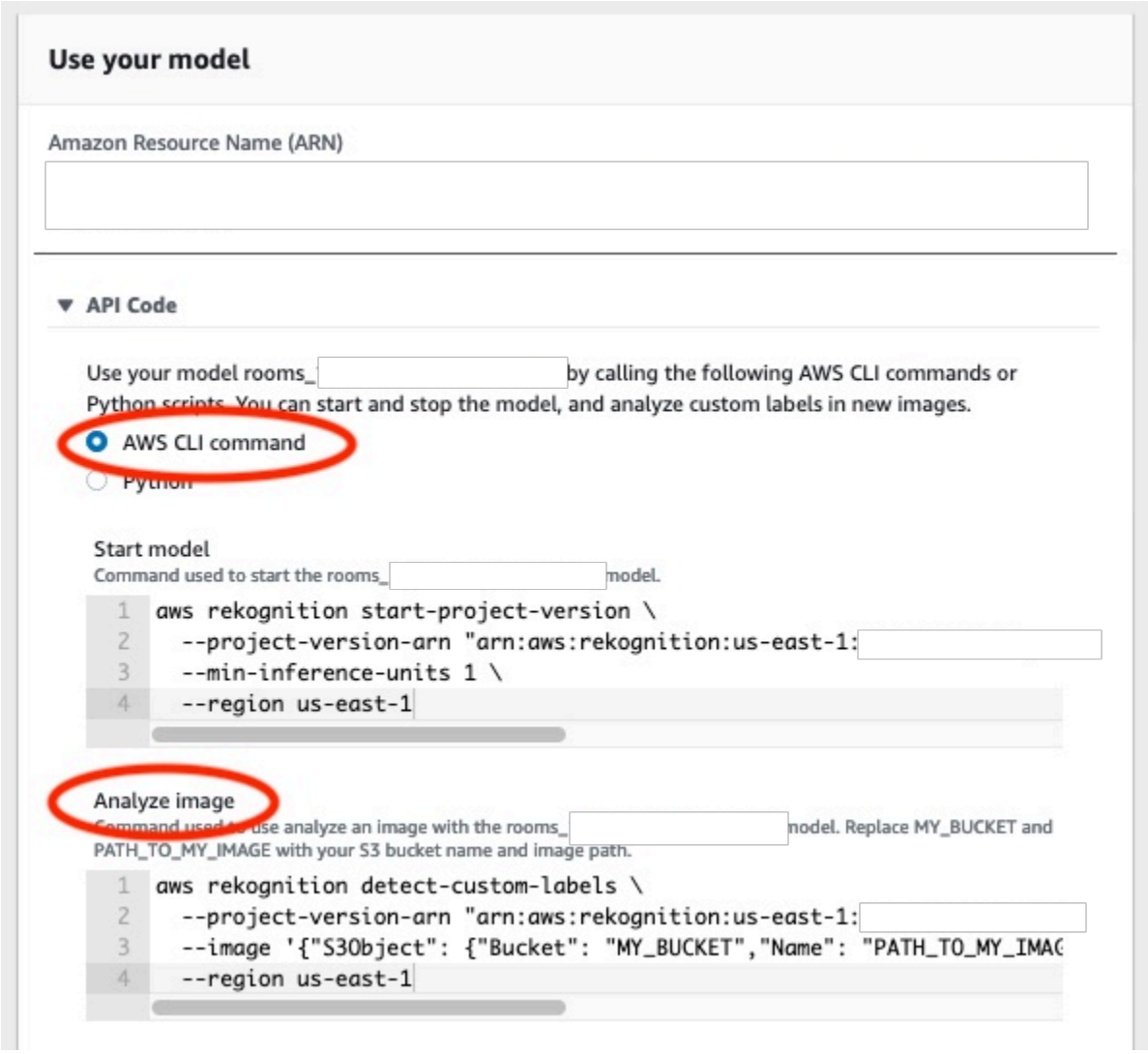
이미지를 분석하려면(콘솔)

1. <textobject><phrase>모델 상태가 실행 중으로 표시되며 실행 중인 모델을 중지하려면 중지 버튼이 있습니다. </phrase></textobject>

   아직 설정하지 않았다면 설정하십시오 AWS CLI. 지침은 the section called “4단계: 및 SDK 설정 AWS CLIAWS” 섹션을 참조하세요.
2. 아직 하지 않았다면 모델을 실행하세요. 자세한 내용은 3단계: 모델 시작 섹션을 참조하세요.
3. 모델 사용 탭을 선택한 다음 API 코드를 선택합니다. 아래 표시된 모델 상태 패널에는 모델이 실행 중으로 표시되며, 실행 중인 모델을 중지하는 중지 버튼과 API를 표시하는 옵션이 있습니다.



4. AWS CLI 명령을 선택합니다.
5. 이미지 분석 섹션에서 호출하는 AWS CLI 명령을 `detect-custom-labels` 복사합니다. Rekognition 콘솔의 다음 이미지는 기계 학습 모델을 사용하여 이미지의 사용자 지정 레이블을 탐지하는 AWS CLI 명령이 포함된 “이미지 분석” 섹션과 모델을 시작하고 이미지 세부 정보를 제공하는 지침을 보여줍니다.



6. Amazon S3 버킷에 예제 이미지를 업로드합니다. 지침은 예제 이미지 가져오기 섹션을 참조하세요.
7. 명령 프롬프트에 이전 단계에서 복사한 AWS CLI 명령을 입력합니다. 다음 예제와 같아야 합니다.

   `--project-version-arn`의 값은 모델의 Amazon 리소스 이름(ARN)이어야 합니다. `--region`의 값은 모델을 생성한 AWS 리전이어야 합니다.

   `MY_BUCKET` 및 `PATH_TO_MY_IMAGE`를 이전 단계에서 사용한 Amazon S3 버킷과 이미지로 변경합니다.

   사용자 지정 레이블 액세스 프로필을 사용하여 자격 증명을 가져오려는 경우 `--profile custom-labels-access` 파라미터를 추가하세요.

```
aws rekognition detect-custom-labels \
```

```
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

모델이 객체, 장면 및 개념을 찾으면 AWS CLI 명령의 JSON 출력이 다음과 같이 표시됩니다. Name은 모델이 찾은 이미지 수준 레이블의 이름입니다. Confidence는 예측 정확도에 대한 모델의 신뢰도입니다.

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

모델이 객체 위치를 찾거나 브랜드를 찾으면 레이블이 지정된 경계 상자가 반환됩니다. BoundingBox는 객체를 둘러싸고 있는 상자의 위치가 포함되어 있습니다. Name은 모델이 경계 상자에서 찾은 객체입니다. Confidence는 경계 상자에 객체가 포함되어 있는지 여부에 대한 모델의 신뢰도입니다.

```
{
    "CustomLabels": [
        {
            "Name": "textract",
            "Confidence": 87.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

8. 모델을 계속 사용하여 다른 이미지를 분석하세요. 더 이상 사용하지 않을 경우 모델을 중지하세요. 자세한 내용은 5단계: 모델 중지 섹션을 참조하세요.

예제 이미지 가져오기

`DetectCustomLabels` 작업에 다음 이미지를 사용할 수 있습니다. 각 프로젝트에 이미지가 하나씩 있습니다. 이미지를 사용하려면 S3 버킷에 업로드합니다.

예제 이미지를 사용하려면

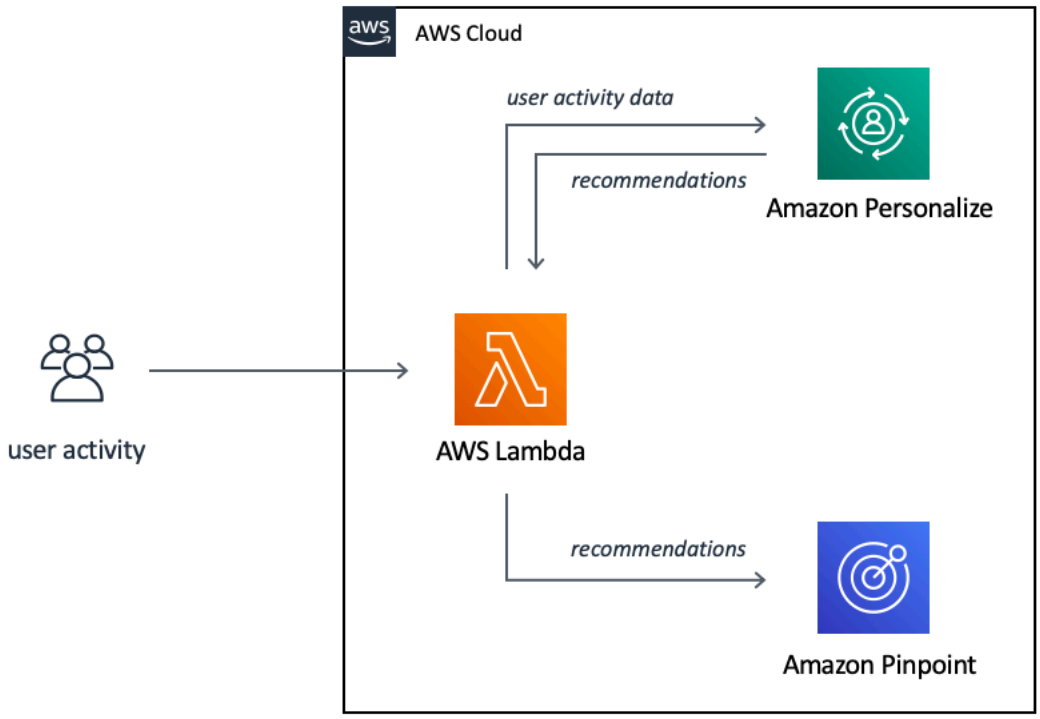
1. 사용 중인 예제 프로젝트와 일치하는 다음 이미지를 마우스 오른쪽 버튼으로 클릭합니다. 그런 다음 이미지 저장을 선택하여 이미지를 컴퓨터에 저장합니다. 사용 중인 브라우저에 따라 메뉴 옵션이 다를 수 있습니다.
2. Amazon Rekognition 사용자 지정 레이블을 사용하고 있는 AWS 지역과 동일한 지역에 있는 사용자 AWS 계정 소유의 Amazon S3 버킷에 이미지를 업로드합니다.

   이에 관한 지침은 Amazon Simple Storage Service 사용 설명서에서 Amazon S3에 객체 업로드를 참조하세요.

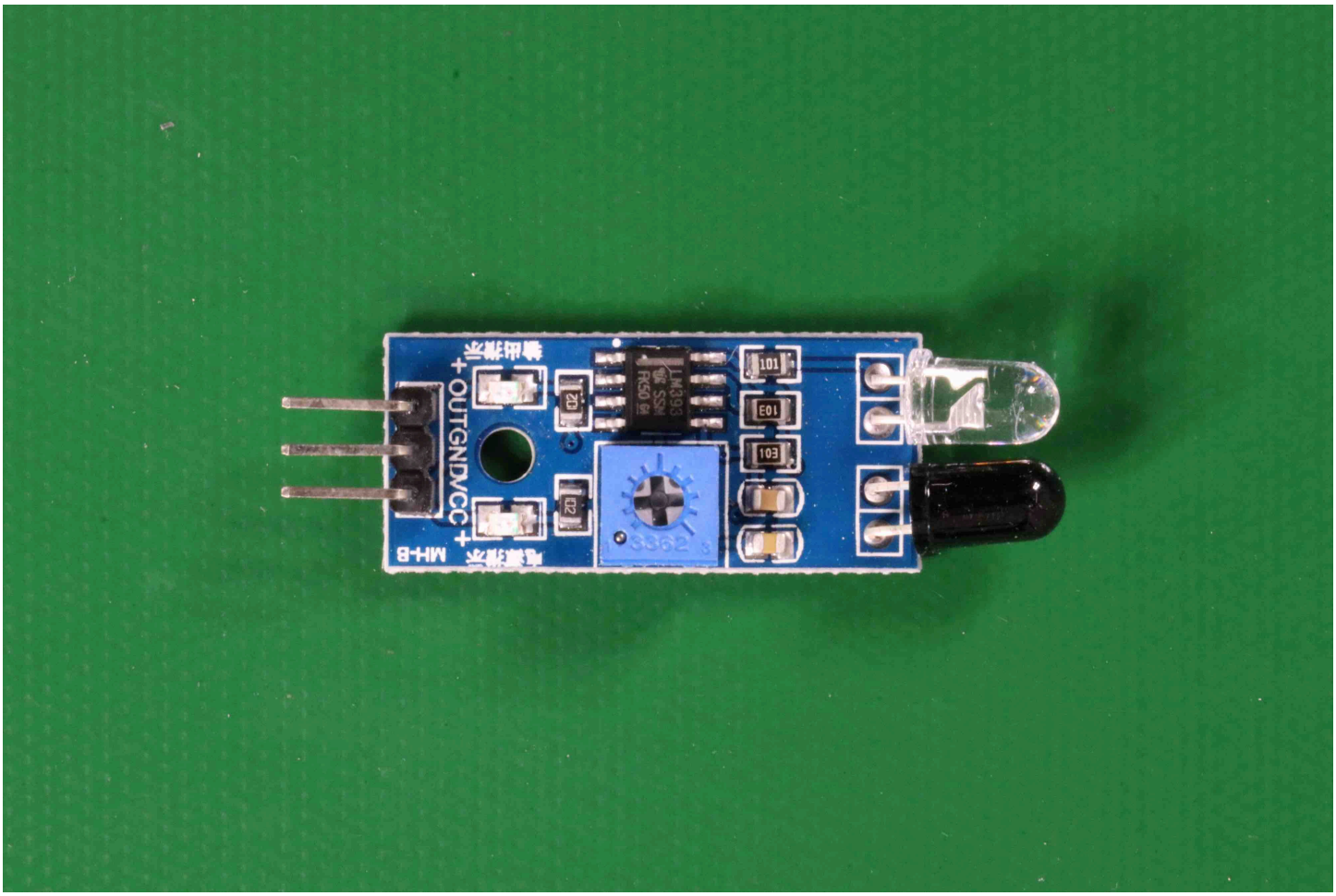
이미지 분류

다중 레이블 분류

브랜드 감지



객체 위치 파악

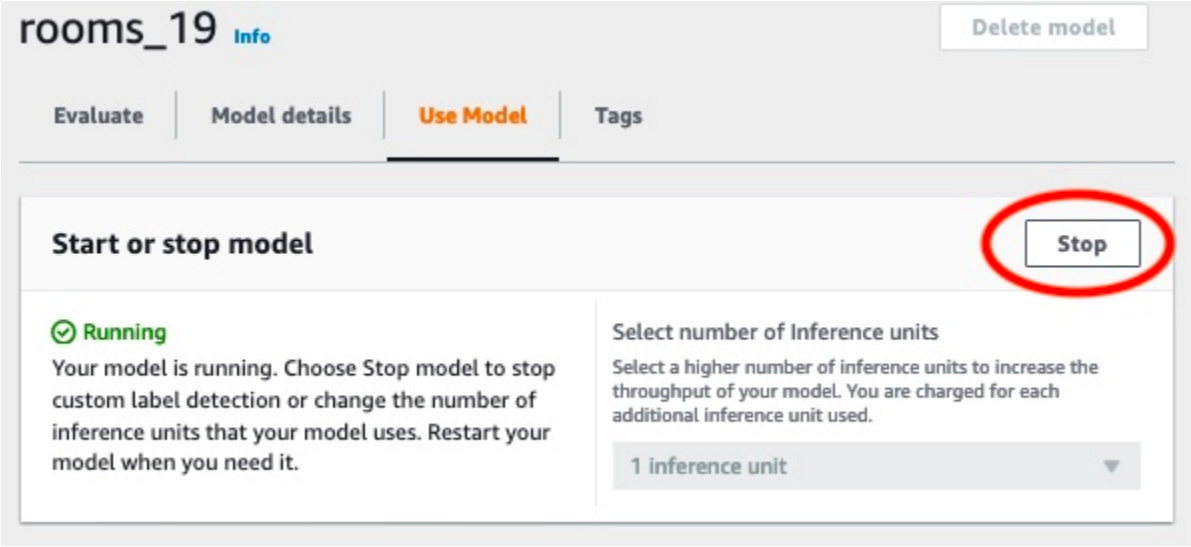


5단계: 모델 중지

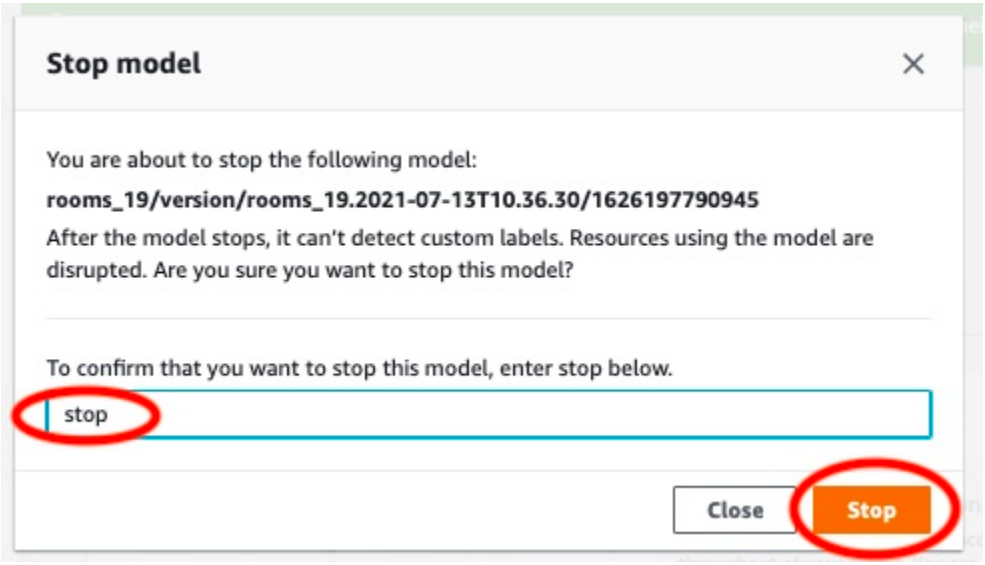
이 단계에서는 모델 실행을 중단합니다. 모델을 실행하는 시간만큼 요금이 부과됩니다. 모델 사용을 마쳤다면 사용을 중지해야 합니다.

모델을 중지하려면

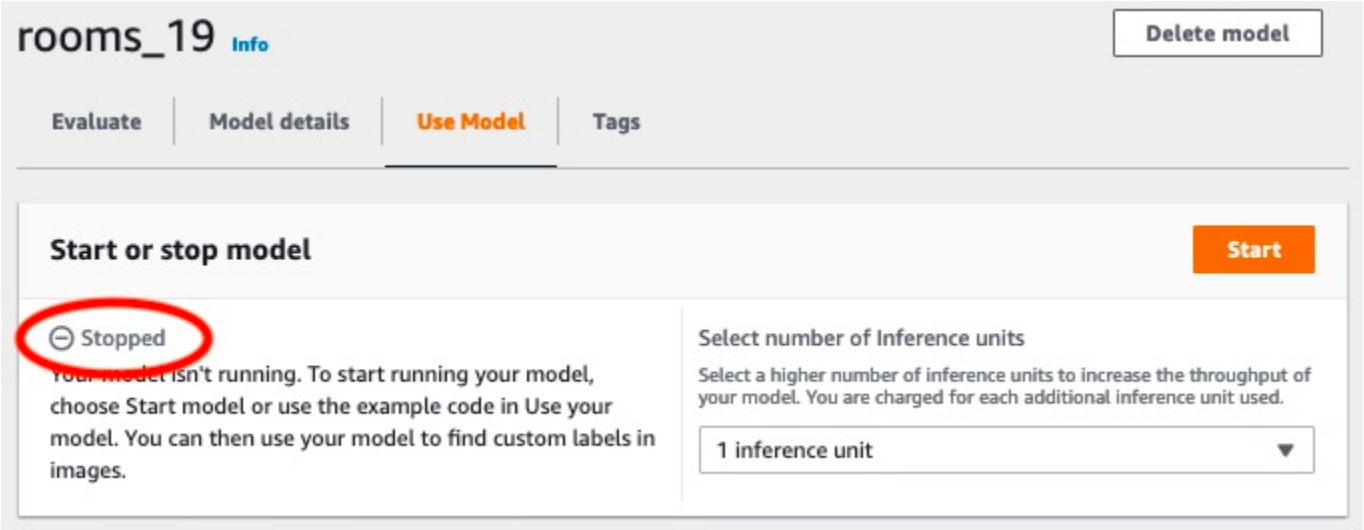
1. 모델 시작 또는 중지 항목에서 중지를 선택합니다.



2. 모델 중지 대화 상자에서 중지를 입력하여 모델 중지를 확인합니다.



3. 중지를 선택하여 모델을 중지합니다. 모델 시작 또는 중지 항목의 상태가 중지됨이면 모델이 중지된 것입니다. 다음 스크린샷의 사용자 인터페이스 섹션에는 기계 학습 모델을 시작하거나 중지할 수 있는 옵션이 있습니다. 모델 상태는 “중지됨”으로 표시되며, 모델을 시작하는 “시작” 버튼과 추론 단위 수를 선택할 수 있는 드롭다운이 나타납니다.



6단계: 다음 단계

예제 프로젝트 체험을 마친 후에는 자체 이미지와 데이터 세트를 사용하여 자체 모델을 생성할 수 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels의 이해 섹션을 참조하세요.

다음 표의 레이블 지정 정보를 사용하여 예제 프로젝트와 유사한 모델을 훈련하세요.

예	훈련 이미지	테스트 이미지
이미지 분류(방)	이미지당 이미지 수준 레이블 1개	이미지당 이미지 수준 레이블 1개
다중 레이블 분류(꽃)	이미지당 이미지 수준 레이블 여러 개	이미지당 이미지 수준 레이블 여러 개
브랜드 감지(로고)	이미지 수준 레이블(레이블이 지정된 경계 상자도 사용할 수 있음)	레이블이 지정된 경계 상자
이미지 위치 파악(회로 기판)	레이블이 지정된 경계 상자	레이블이 지정된 경계 상자

튜토리얼: 이미지 분류 항목은 이미지 분류 모델을 위한 프로젝트, 데이터 세트, 모델을 생성하는 방법을 보여줍니다.

데이터 세트 및 훈련 모델 생성에 대한 자세한 내용은 Amazon Rekognition Custom Labels 모델 생성 항목을 참조하세요.

튜토리얼: 이미지 분류

이 튜토리얼은 이미지에서 찾은 객체, 장면, 개념을 분류하는 모델의 프로젝트와 데이터 세트를 만드는 방법을 보여줍니다. 모델은 전체 이미지를 분류합니다. 이 튜토리얼을 따르면 예를 들어 거실이나 주방과 같은 가정 위치를 인식하도록 모델을 훈련할 수 있습니다. 이 튜토리얼은 모델을 사용하여 이미지를 분석하는 방법도 보여줍니다.

튜토리얼을 시작하기 전에 Amazon Rekognition Custom Labels의 이해 항목을 읽어 보시면 좋습니다.

이 튜토리얼에서는 로컬 컴퓨터에서 이미지를 업로드하여 훈련 및 테스트 데이터 세트를 만듭니다. 그 후 훈련 및 테스트 데이터 세트의 이미지에 이미지 수준 레이블을 지정합니다.

생성한 모델은 사용자가 훈련 데이터 세트 이미지에 지정한 이미지 수준 레이블 세트에 따라 이미지를 분류합니다. 예를 들어 훈련 데이터 세트의 이미지 수준 레이블 집합이 `kitchen`, `living_room`, `patio`, `backyard`인 경우 모델은 단일 이미지에서 이러한 모든 이미지 수준 레이블을 찾을 수 있습니다.

> **Note**
>
> 이미지에서 객체의 위치를 찾는 등 다양한 목적으로 모델을 만들 수 있습니다. 자세한 내용은 모델 유형 결정 섹션을 참조하세요.

1단계: 이미지 수집

두 세트의 이미지가 필요합니다. 훈련 데이터 세트에 추가할 세트 하나. 그리고 테스트 데이터 세트에 추가할 세트 하나. 이미지는 모델이 분류할 대상, 장면, 개념을 나타내야 합니다. 이미지는 PNG 또는 JPEG 형식이어야 합니다. 자세한 내용은 이미지 준비 섹션을 참조하세요.

훈련 데이터 세트에는 최소 10개, 테스트 데이터 세트에는 10개 이상의 이미지가 있어야 합니다.

아직 이미지가 없는 경우 Rooms 예제 분류 프로젝트의 이미지를 사용하세요. 프로젝트를 생성한 후 훈련 및 테스트 이미지는 다음 Amazon S3 버킷 위치에 있습니다.

- 훈련 이미지: `s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`
- 테스트 이미지: `s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`

region 항목은 사용자가 Amazon Rekognition Custom Labels 콘솔을 사용하고 있는 AWS 리전입니다. numbers 항목은 콘솔이 버킷 이름에 할당하는 값입니다. Version number 항목은 1부터 시작하는 예제 프로젝트의 버전 번호입니다.

다음 절차는 Rooms 프로젝트의 이미지를 컴퓨터의 로컬 폴더에 training 및 test 이름으로 저장합니다.

Rooms 예제 프로젝트 이미지 파일을 다운로드하려면

1. Rooms 프로젝트를 생성하세요. 자세한 내용은 1단계: 예제 프로젝트 선택 섹션을 참조하세요.
2. 명령 프롬프트를 열고 다음 명령을 입력하여 훈련 이미지를 다운로드합니다.

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
 number_training_dataset/ training --recursive
```

3. 명령 프롬프트에서 다음 명령을 입력하여 테스트 이미지를 다운로드합니다.

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
 number_test_dataset/ test --recursive
```

4. 훈련 폴더의 이미지 두 개를 선택한 별도의 폴더로 옮깁니다. 해당 이미지는 훈련시킨 모델을 9단계: 모델을 사용하여 이미지 분석에서 시험하는 데 사용됩니다.

2단계: 분류 결정

모델로 찾고자 하는 분류의 목록을 생성합니다. 예를 들어, 집 안의 방을 인식하도록 모델을 훈련하는 경우 다음 이미지를 living_room 항목으로 분류할 수 있습니다.

각 분류는 이미지 수준 레이블에 매핑됩니다. 그 후 훈련 및 테스트 데이터 세트의 이미지에 이미지 수준 레이블을 지정합니다.

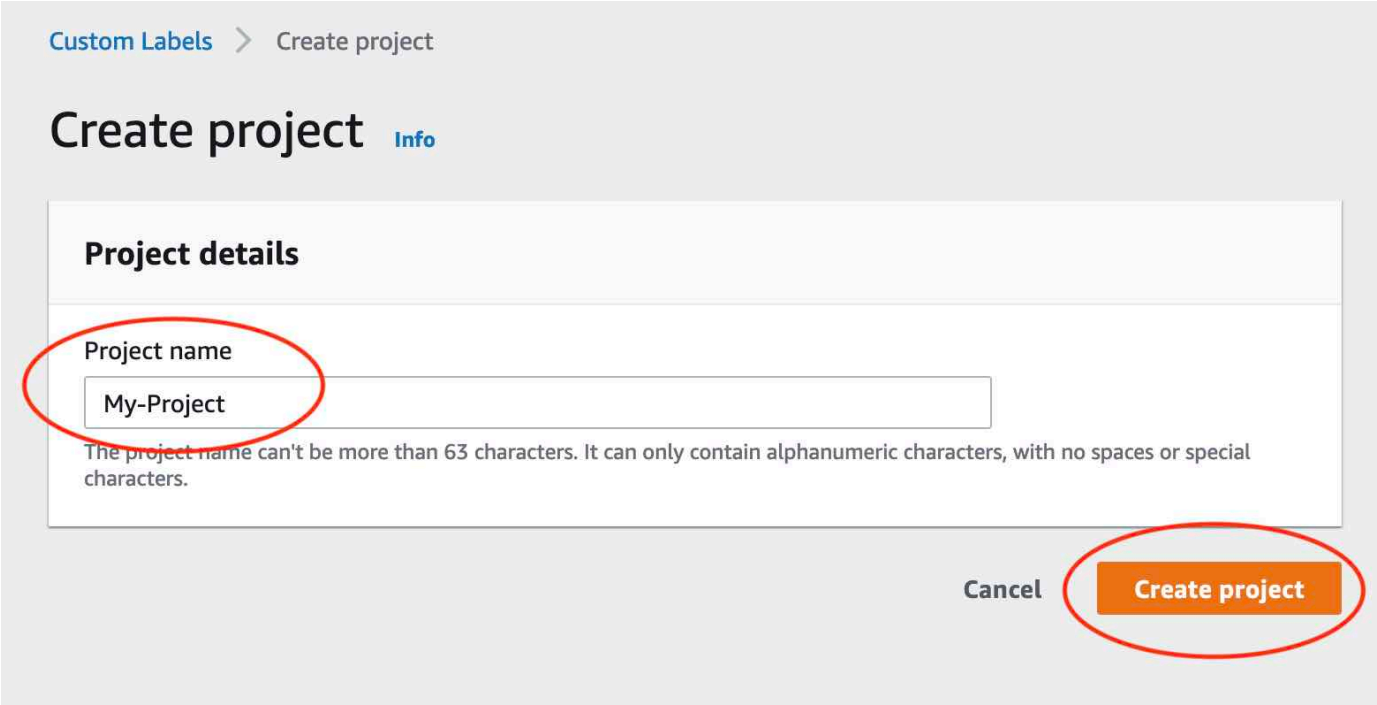
Rooms 예제 프로젝트의 이미지를 사용하는 경우 이미지 수준 레이블은 뒷마당, 욕실, 침실, 옷장, 진입로, 평면도, 앞마당, 주방, 거실, 파티오입니다.

3단계: 프로젝트 생성

데이터 세트와 모델을 관리하기 위해 프로젝트를 생성합니다. 각 프로젝트는 집 안의 방 인식과 같은 단일 용도로 사용되어야 합니다.

프로젝트 생성 방법(콘솔)

1. 아직 하지 않았다면 Amazon Rekognition Custom Labels 콘솔을 설정합니다. 자세한 내용은 Amazon Rekognition Custom Labels 설정 섹션을 참조하세요.
2. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
3. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다.
4. Amazon Rekognition Custom Labels 랜딩 페이지에서 시작하기를 선택합니다.
5. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
6. 프로젝트 페이지에서 프로젝트 생성을 선택합니다.
7. 프로젝트 이름에 프로젝트의 이름을 입력합니다.
8. 프로젝트를 생성하려면 프로젝트 생성을 선택합니다.



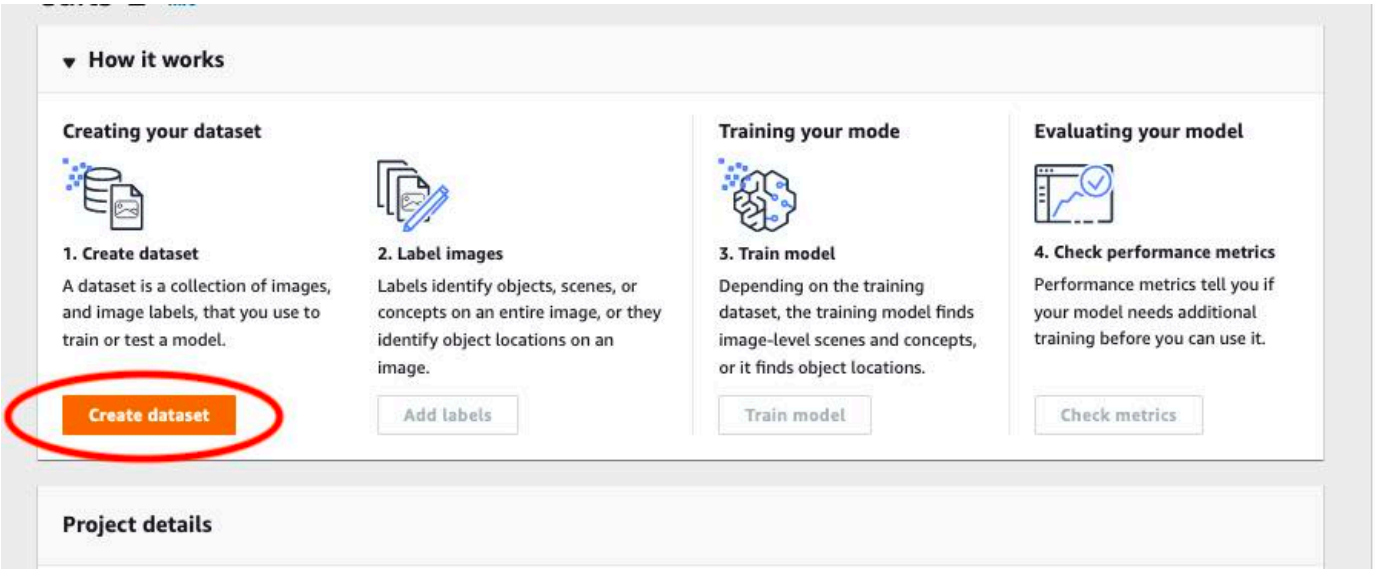
4단계: 훈련 및 테스트 데이터 세트 생성

이 단계에서는 로컬 컴퓨터에서 이미지를 업로드하여 훈련 및 테스트 데이터 세트를 만듭니다. 한 번에 최대 30개의 이미지를 업로드할 수 있습니다. 업로드할 이미지가 많은 경우 Amazon S3 버킷에서 이미지를 가져와서 데이터 세트를 생성하는 것을 고려해 보세요. 자세한 내용은 Amazon S3 버킷 섹션을 참조하세요.

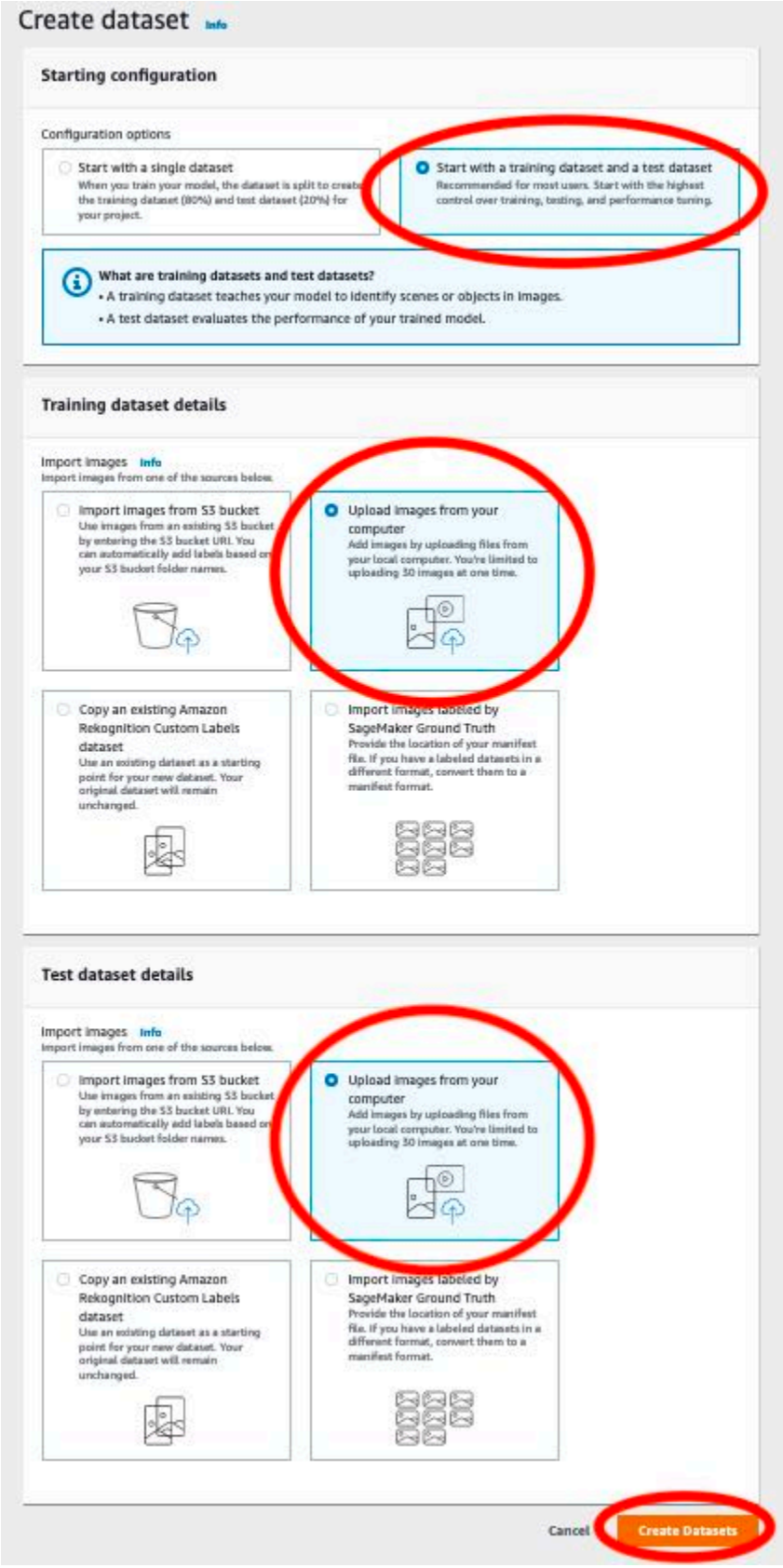
데이터 세트에 관한 자세한 내용은 데이터 세트 관리 항목을 참조하세요.

로컬 컴퓨터의 이미지를 사용하여 데이터 세트를 만들려면(콘솔)

1. 프로젝트 세부 정보 페이지에서 데이터 세트 생성을 선택합니다.

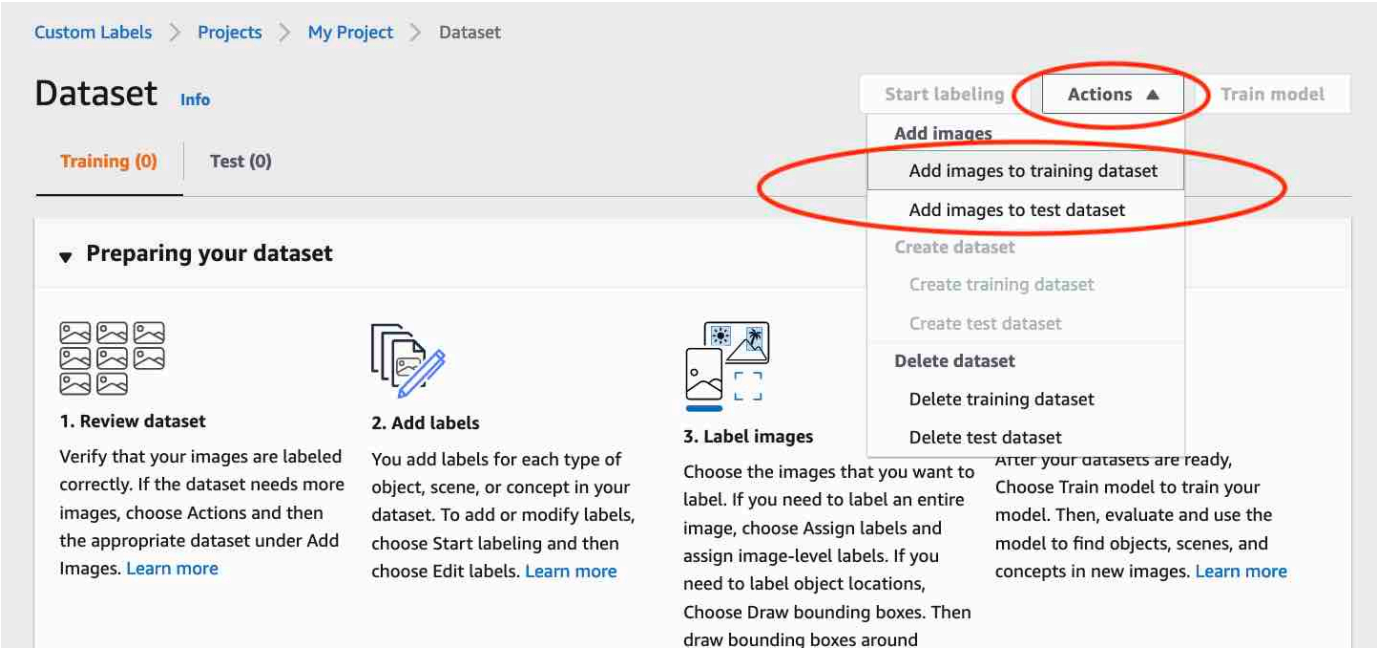


2. 시작 구성 항목에서 훈련 데이터 세트와 테스트 데이터 세트로 시작을 선택합니다.
3. 훈련 데이터 세트 세부 정보 항목에서 컴퓨터에서 이미지 업로드를 선택합니다.
4. 테스트 데이터 세트 세부 정보 항목에서 컴퓨터에서 이미지 업로드를 선택합니다.
5. 데이터 세트 생성을 선택합니다.

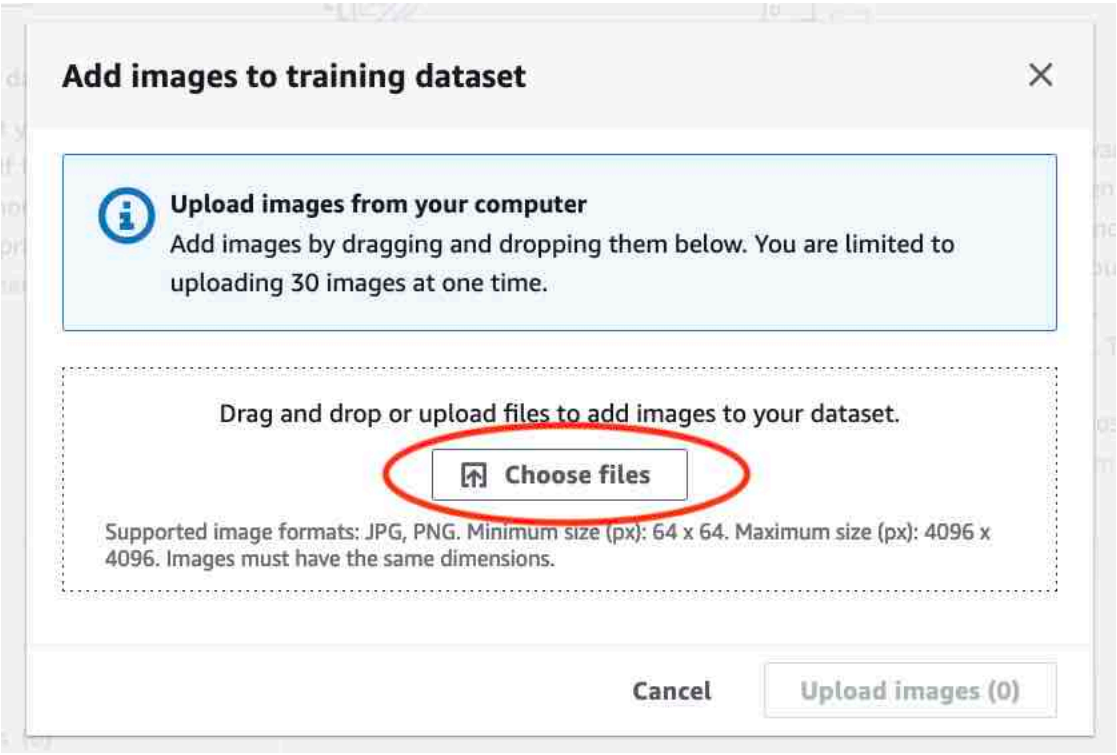


6. 각 데이터 세트에 대한 훈련 탭과 테스트 탭이 있는 데이터 세트 페이지가 나타납니다.

7. 데이터 세트 페이지에서 훈련 탭을 선택합니다.
8. 작업을 선택한 다음 훈련 데이터 세트에 이미지 추가를 선택합니다.

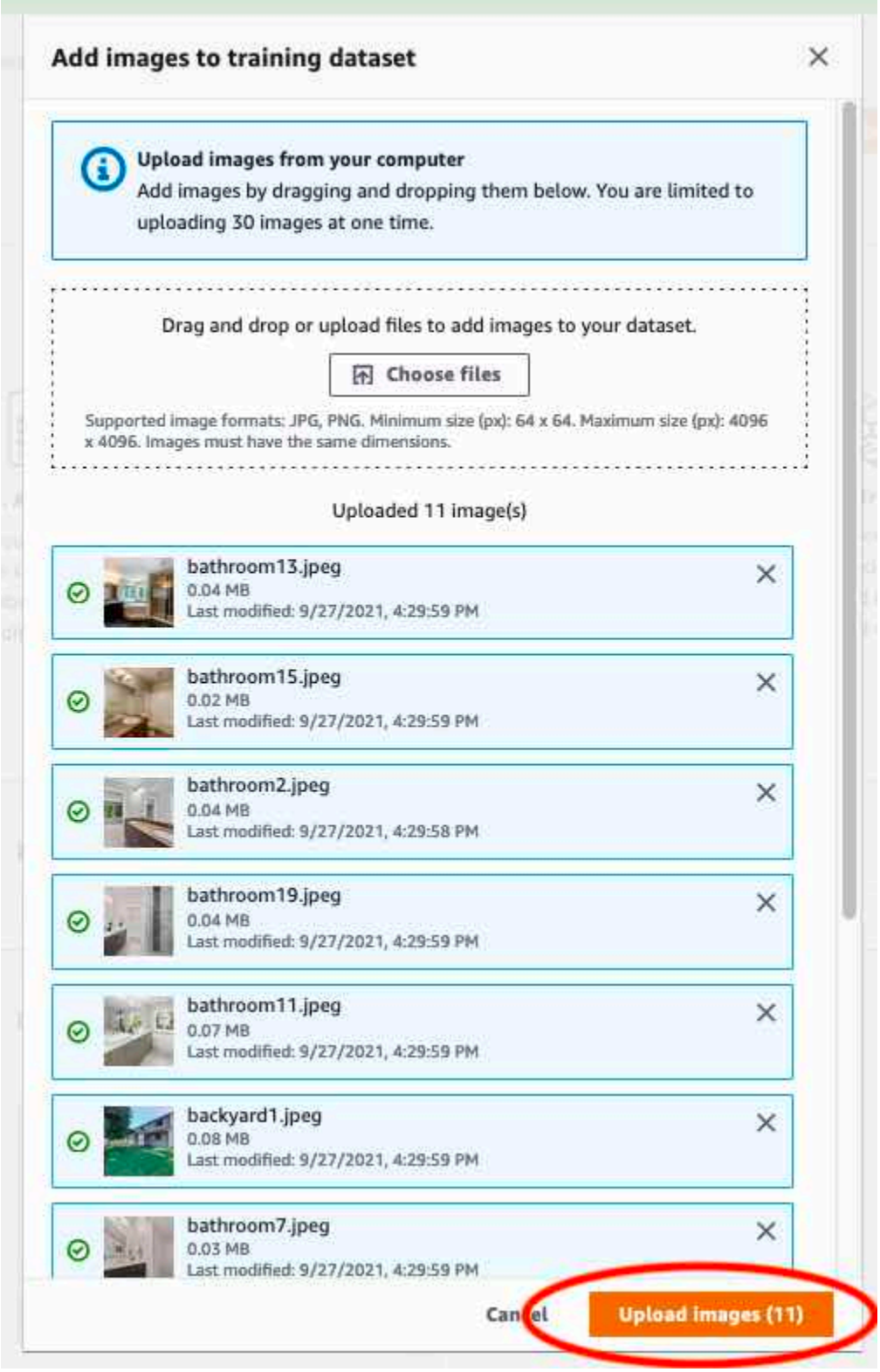


9. 훈련 데이터 세트에 이미지 추가 대화 상자에서 파일 선택을 선택합니다.



10. 데이터 세트에 업로드할 이미지를 선택합니다. 한 번에 최대 30개의 이미지를 업로드할 수 있습니다.

11. 이미지 업로드를 선택합니다. Amazon Rekognition Custom Labels가 이미지를 데이터 세트에 추가하는 데 몇 초 정도 걸릴 수 있습니다.



12. 훈련 데이터 세트에 추가할 이미지가 더 있으면 9~12단계를 반복합니다.

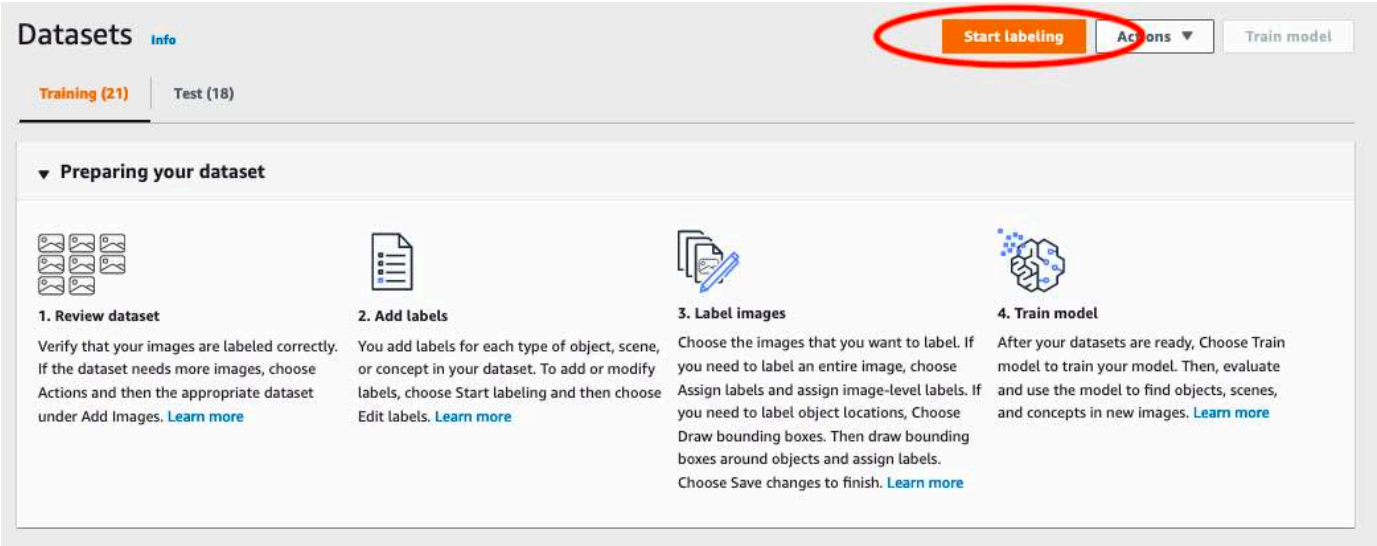
13. 테스트 탭을 선택합니다.
14. 8~12단계를 반복하여 테스트 데이터 세트에 이미지를 추가합니다. 8단계에서는 작업을 선택한 다음 테스트 데이터 세트에 이미지 추가를 선택합니다.

5단계: 프로젝트에 레이블 추가

이 단계에서는 2단계: 분류 결정단계에서 식별한 각 분류에 대해 레이블을 프로젝트에 추가합니다.

새 레이블을 추가하려면(콘솔)

1. 데이터 세트 갤러리 페이지에서 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.



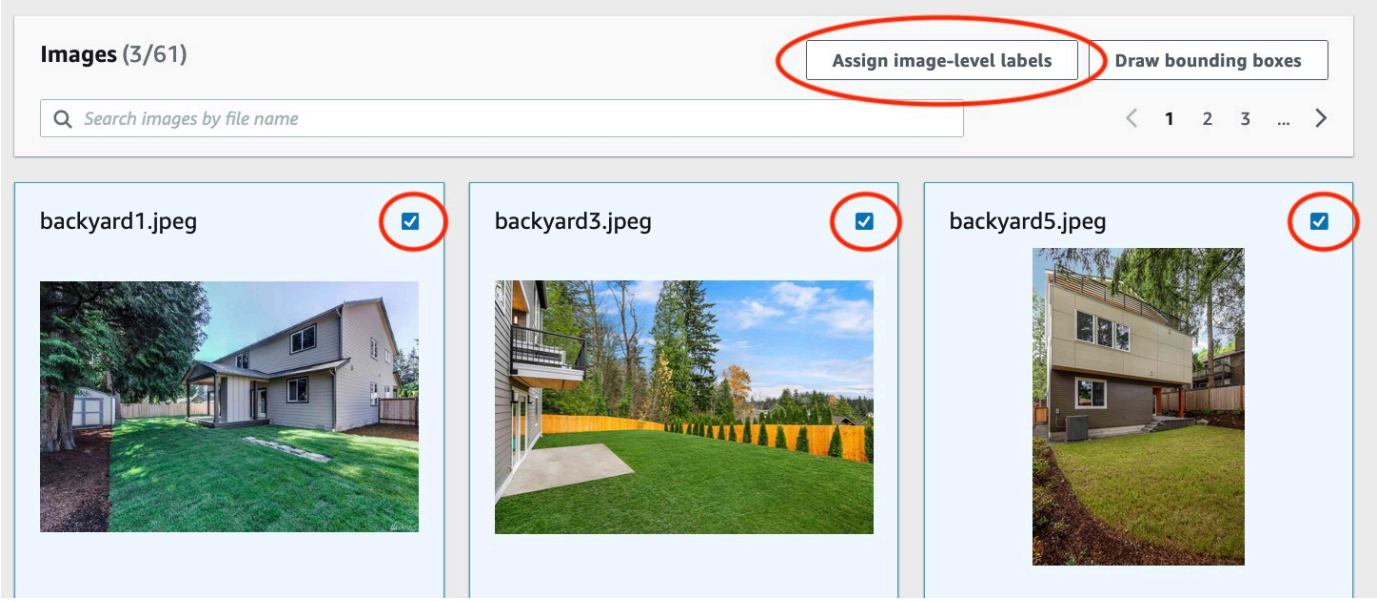
2. 데이터 세트 갤러리의 레이블 항목에서 레이블 편집을 선택하여 레이블 관리 대화 상자를 엽니다.
3. 편집 상자에 새 레이블 이름을 입력합니다.
4. 레이블 추가를 선택합니다.
5. 필요한 레이블을 모두 만들 때까지 3단계와 4단계를 반복합니다.
6. 저장을 선택하여 추가한 레이블을 저장합니다.

6단계: 훈련 및 테스트 데이터 세트에 이미지 수준 레이블 지정

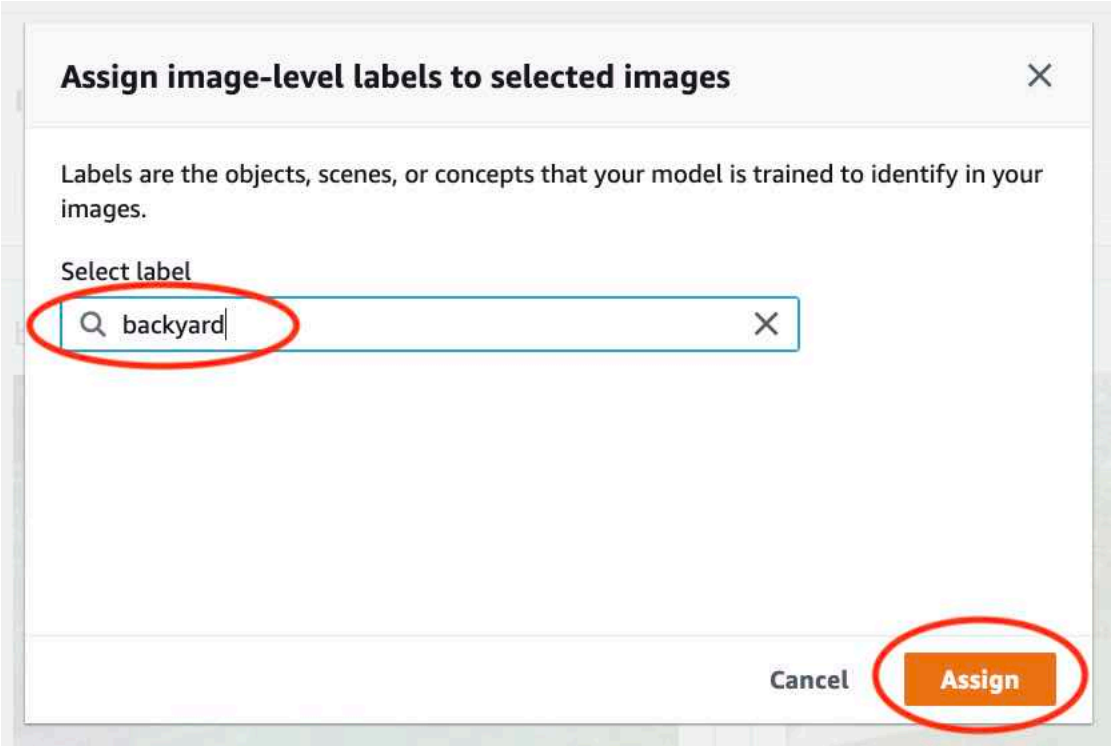
이 단계에서는 훈련 및 테스트 데이터 세트의 각 이미지에 하나의 이미지 수준을 지정합니다. 이미지 수준 레이블은 각 이미지가 나타내는 분류입니다.

이미지에 이미지 수준 레이블을 지정하려면(콘솔)

1. 데이터 세트 페이지에서 훈련 탭을 선택합니다.
2. 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.
3. 레이블을 추가할 이미지를 하나 이상 선택합니다. 한 번에 하나의 페이지에 있는 이미지만 선택할 수 있습니다. 한 페이지에서 연속된 이미지 범위를 선택하려면:
   a. 첫 번째 이미지를 선택합니다.
   b. Shift 키를 길게 누릅니다.
   c. 두 번째 이미지를 선택합니다. 첫 번째 이미지와 두 번째 이미지 사이의 이미지도 선택됩니다.
   d. Shift 키를 놓습니다.
4. 이미지 수준 레이블 지정을 선택합니다.



5. 선택한 이미지에 이미지 수준 레이블 지정 대화 상자에서 이미지에 지정하려는 레이블을 선택합니다.
6. 지정을 선택하여 이미지에 레이블을 지정합니다.



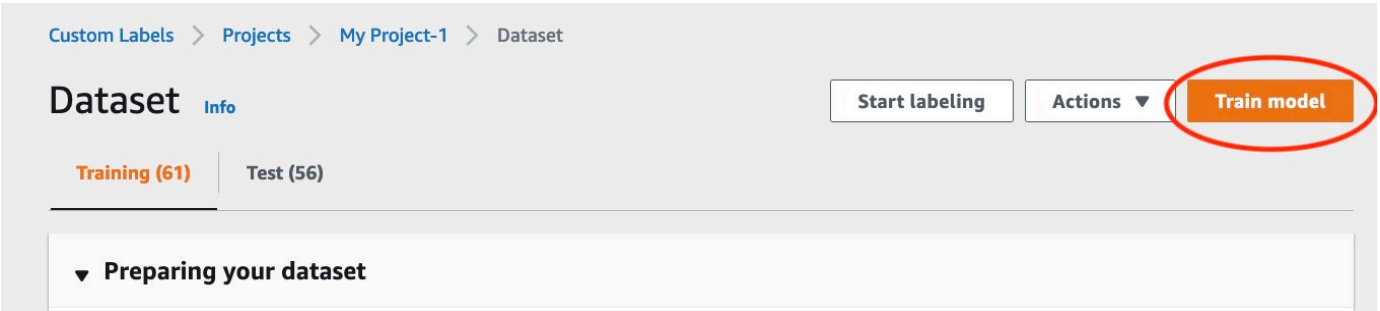
7. 모든 이미지에 필요한 레이블이 주석으로 달릴 때까지 레이블 지정을 반복합니다.
8. 테스트 탭을 선택합니다.
9. 단계를 반복하여 테스트 데이터 세트 이미지에 이미지 수준 레이블을 지정합니다.

7단계: 모델 훈련

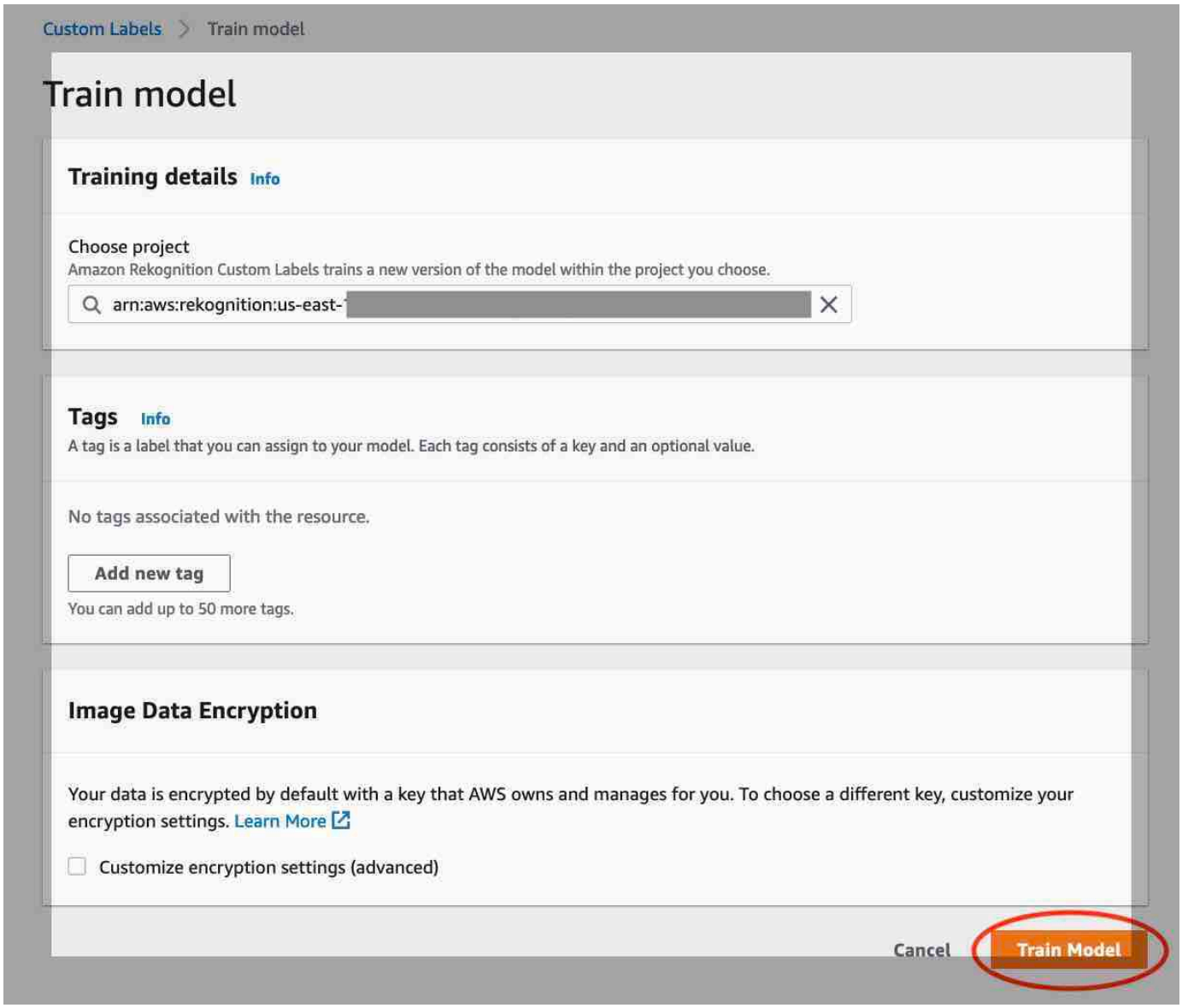
다음 단계를 사용하여 모델을 훈련하세요. 자세한 내용은 Amazon Rekognition Custom Labels 모델 훈련 섹션을 참조하세요.

모델을 훈련하려면(콘솔)

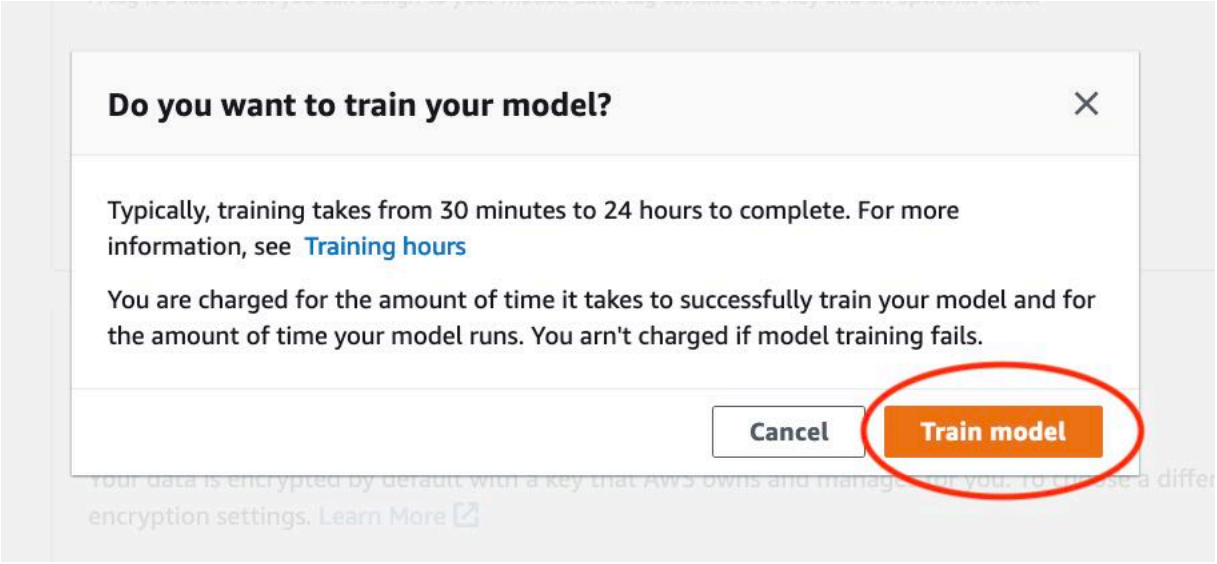
1. 데이터 세트 페이지에서 모델 훈련을 선택합니다.



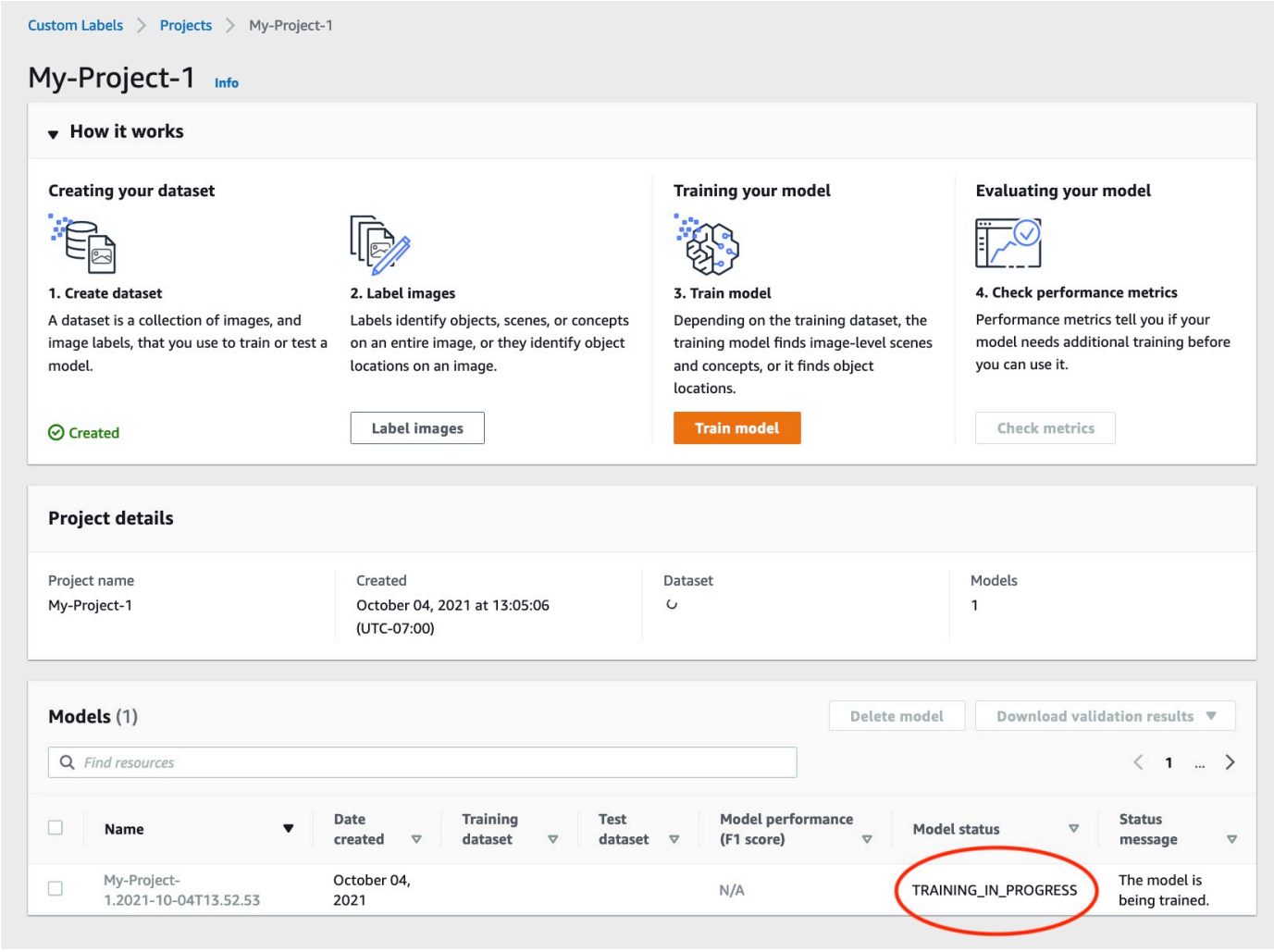
2. 모델 훈련 페이지에서 모델 훈련을 선택합니다. 프로젝트의 Amazon 리소스 이름(ARN)은 프로젝트 선택 편집 상자에 있습니다.



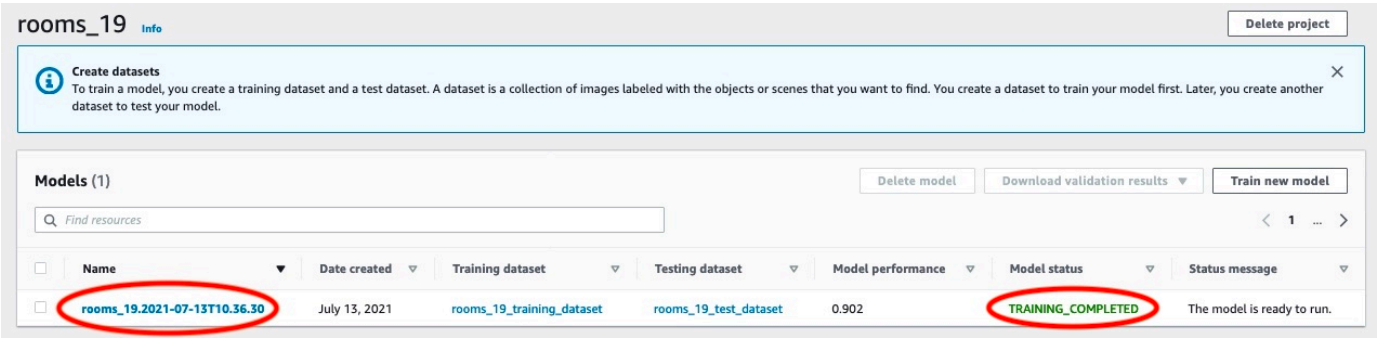
3. 모델을 훈련하고 싶으신가요? 대화 상자에서 모델 훈련을 선택합니다.



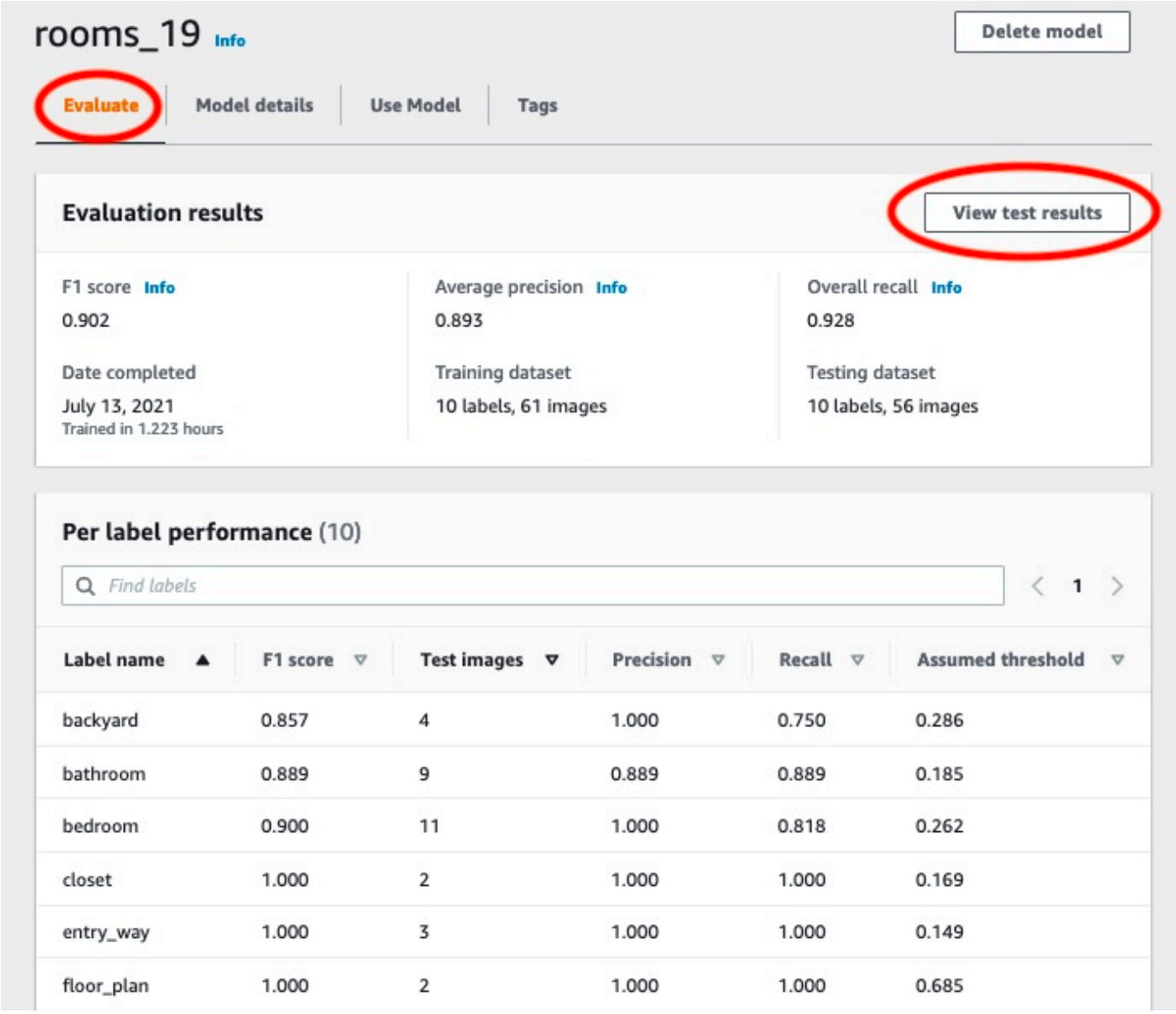
4. 프로젝트 페이지의 모델 항목에서 훈련이 진행 중임을 확인할 수 있습니다. 모델 버전의 `Model Status` 열을 보면 현재 상태를 확인할 수 있습니다. 모델 훈련은 완료하는 데 다소 시간이 걸립니다.



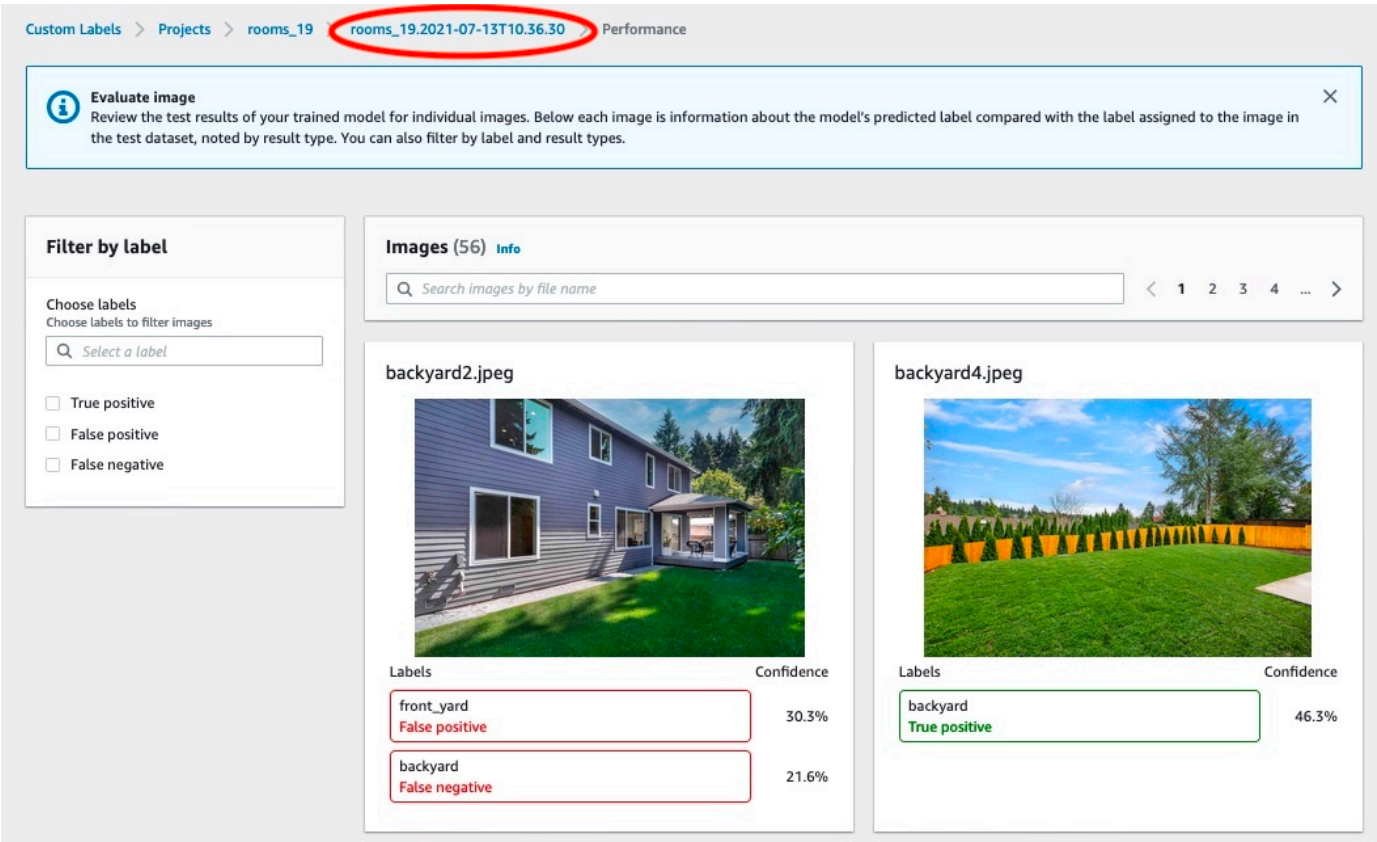
5. 훈련이 완료되면 모델 이름을 선택합니다. 모델 상태가 TRAINING_COMPLETED로 표시되면 훈련이 끝났다는 뜻입니다.



6. 평가 버튼을 선택하면 평가 결과를 볼 수 있습니다. 모델 평가에 대한 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 항목을 참조하세요.
7. 개별 테스트 이미지의 결과를 보려면 테스트 결과 보기를 선택합니다. 자세한 내용은 모델 평가를 위한 지표 섹션을 참조하세요.



8. 테스트 결과를 확인한 후 모델 이름을 선택하여 모델 페이지로 돌아가세요.



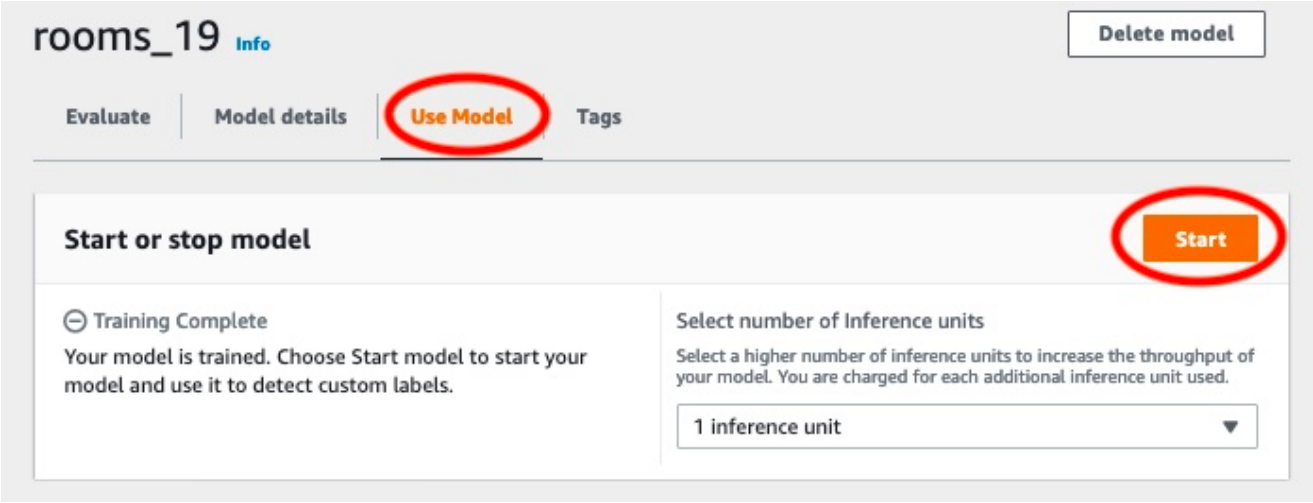
8단계: 모델 시작

이 단계에서는 모델을 시작합니다. 모델이 시작되면 모델을 사용하여 이미지를 분석할 수 있습니다.

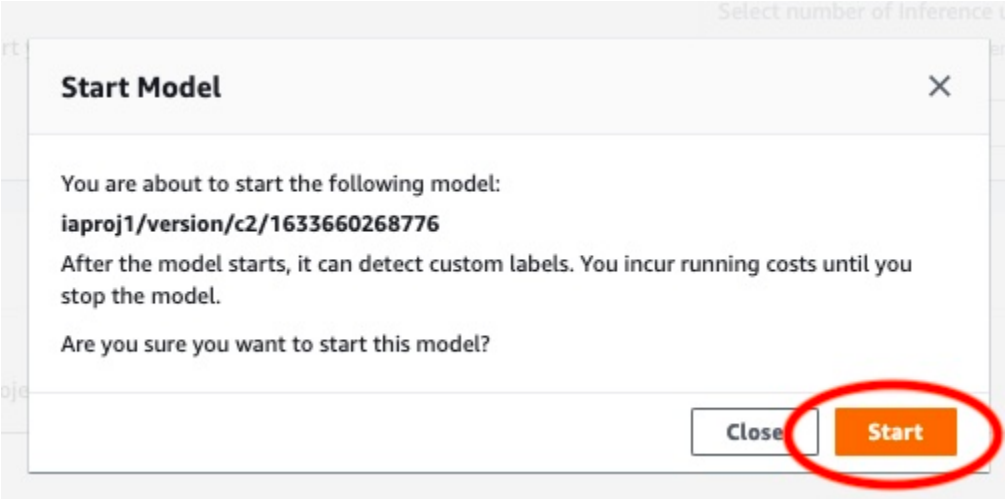
모델을 실행하는 시간만큼 요금이 부과됩니다. 이미지를 분석할 필요가 없는 경우 모델을 중지하세요. 나중에 모델을 다시 시작할 수 있습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 실행 섹션을 참조하세요.

모델을 시작하려면

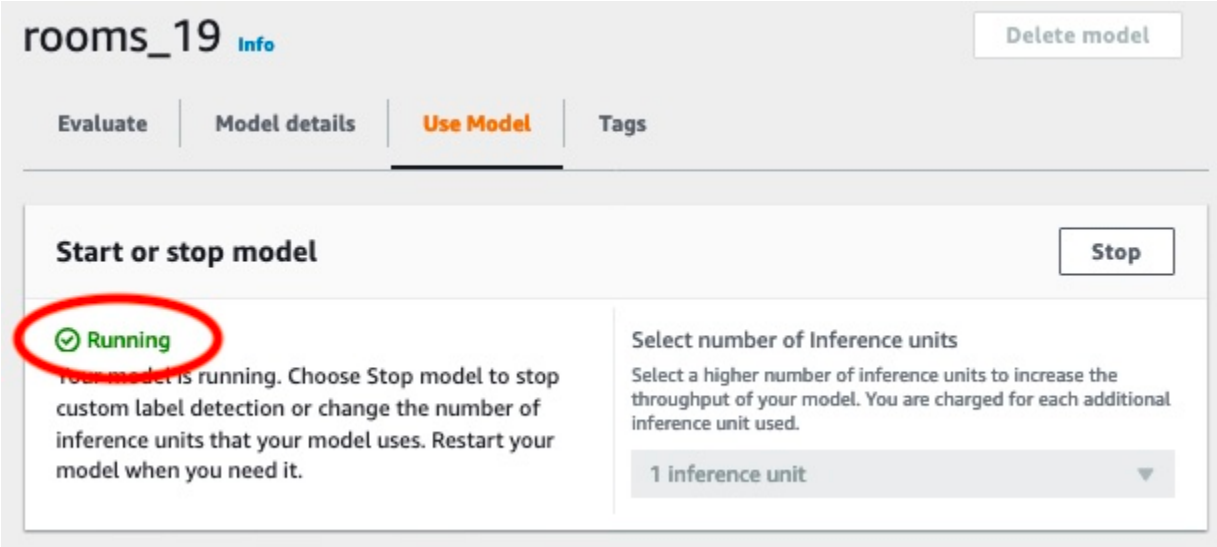
1. 모델 페이지에서 모델 사용 탭을 선택합니다.
2. 모델 시작 또는 중지 항목에서 다음을 수행하세요.
    a. 시작을 선택합니다.



b. 모델 시작 대화 상자에서 시작을 선택합니다.



3. 모델이 실행될 때까지 기다리세요. 모델 시작 또는 중지 항목의 상태가 실행 중이면 모델이 실행되고 있다는 뜻입니다.



9단계: 모델을 사용하여 이미지 분석

DetectCustomLabels API를 호출하여 이미지를 분석합니다. 이 단계에서는 `detect-custom-labels` AWS Command Line Interface(AWS CLI) 명령을 사용하여 예제 이미지를 분석합니다. Amazon Rekognition Custom Labels 콘솔에서 AWS CLI 명령을 가져올 수 있습니다. 콘솔은 모델을 사용하도록 AWS CLI 명령을 구성합니다. 사용자는 Amazon S3 버킷에 저장된 이미지만 제공하면 됩니다.

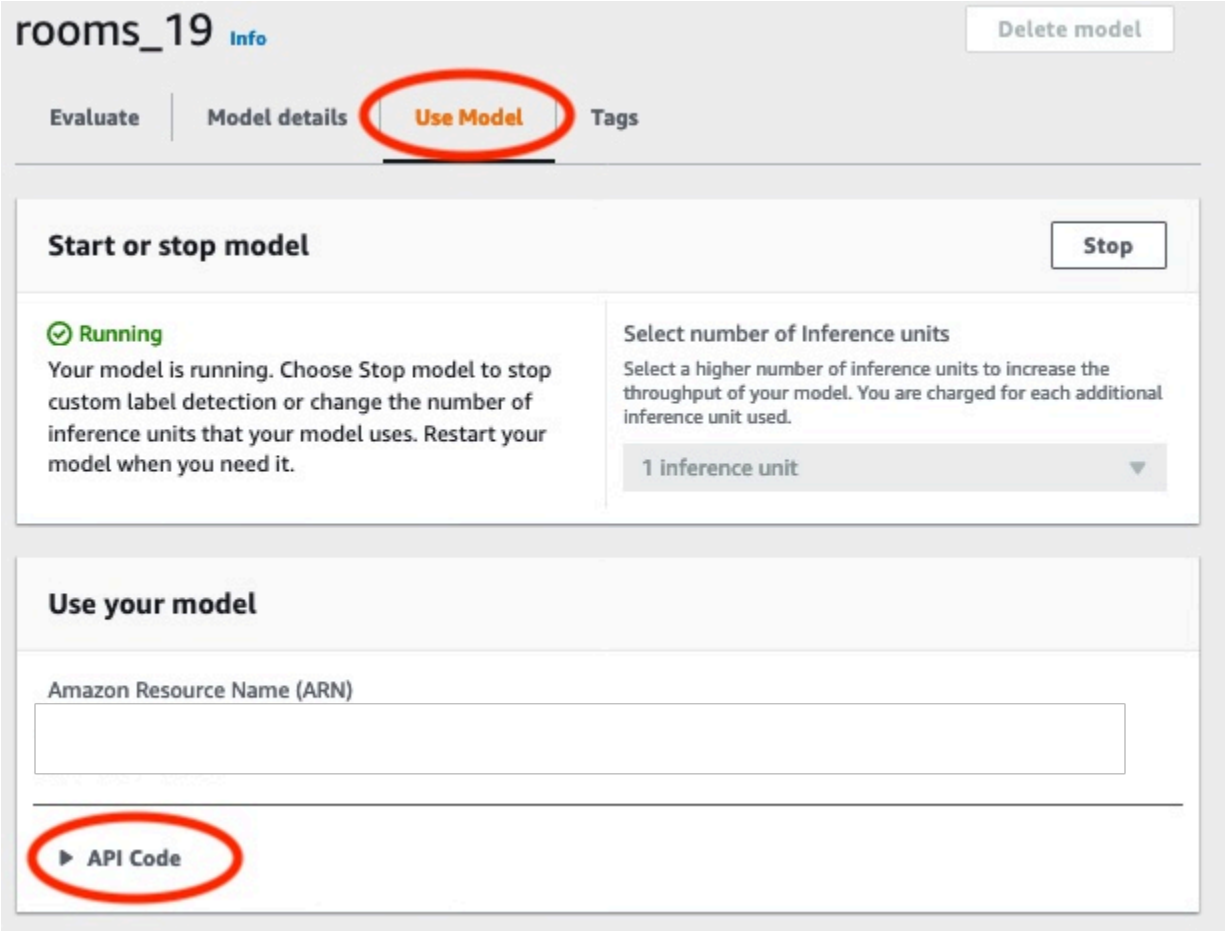
> **Note**
>
> 콘솔은 Python 예제 코드도 제공합니다.

`detect-custom-labels`의 출력에는 이미지에서 찾은 레이블 목록, 경계 상자(모델이 객체 위치를 찾는 경우), 예측 정확도에 대한 모델의 신뢰도가 포함됩니다.

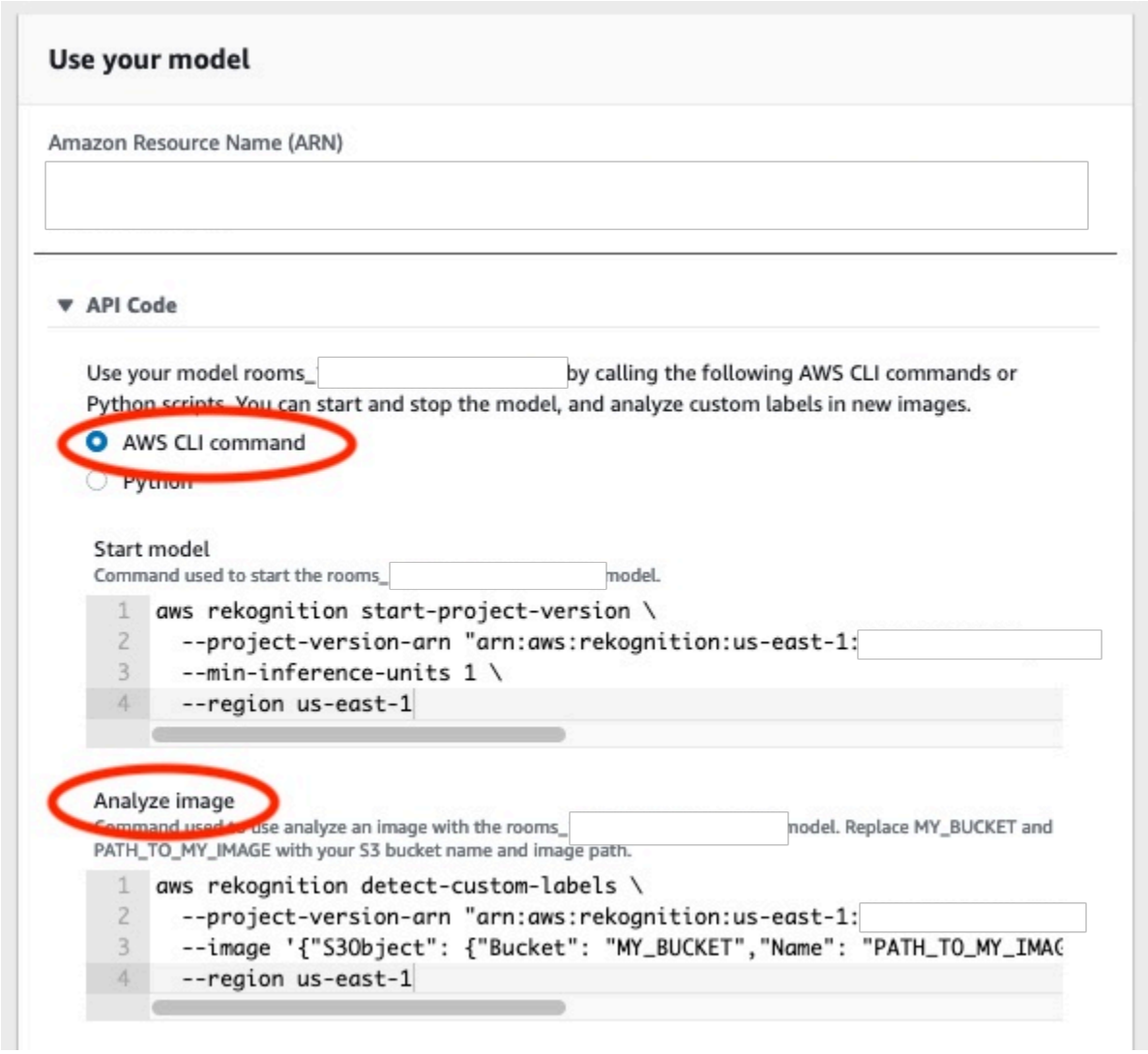
자세한 내용은 훈련된 모델을 사용한 이미지 분석 섹션을 참조하세요.

이미지를 분석하려면(콘솔)

1. 아직 하지 않았다면 AWS CLI 항목을 설정하세요. 지침은 the section called “4단계: 및 SDK 설정 AWS CLIAWS” 섹션을 참조하세요.
2. 모델 사용 탭을 선택한 다음 API 코드를 선택합니다.



3. AWS CLI 명령을 선택합니다.
4. 이미지 분석 항목에서 `detect-custom-labels` 항목을 호출하는 AWS CLI 명령을 복사합니다.



5. Amazon S3 버킷에 이미지를 업로드합니다. 이에 관한 지침은 Amazon Simple Storage Service 사용 설명서에서 Amazon S3에 객체 업로드를 참조하세요. Rooms 프로젝트의 이미지를 사용하는 경우 1단계: 이미지 수집에서 별도의 폴더로 옮긴 이미지 중 하나를 사용하세요.

6. 명령 프롬프트에서 이전 단계에서 복사한 AWS CLI 명령을 입력합니다. 다음 예제와 같아야 합니다.

   `--project-version-arn`의 값은 모델의 Amazon 리소스 이름(ARN)이어야 합니다. `--region`의 값은 모델을 생성한 AWS 리전이어야 합니다.

   `MY_BUCKET` 및 `PATH_TO_MY_IMAGE` 항목을 이전 단계에서 사용한 Amazon S3 버킷과 이미지로 변경합니다.

사용자 지정 레이블 액세스 프로필을 사용하여 자격 증명을 가져오려는 경우 `--profile custom-labels-access` 파라미터를 추가하세요.

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

AWS CLI 명령의 JSON 출력이 다음과 같아야 합니다. Name 항목은 모델이 찾은 이미지 수준 레이블의 이름입니다. Confidence(0-100) 항목은 예측 정확도에 대한 모델의 신뢰도입니다.

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

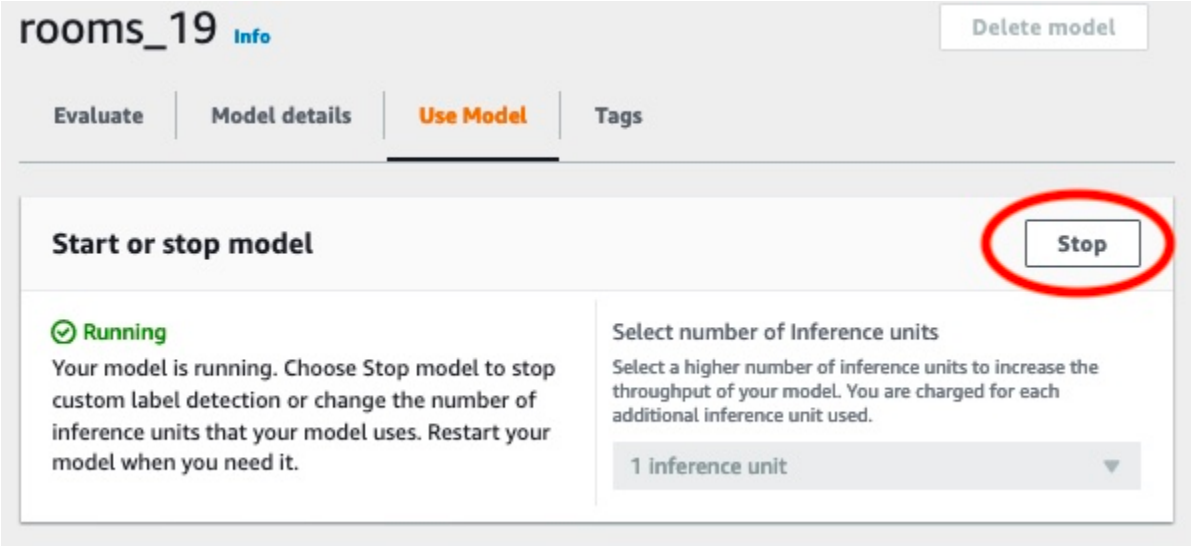
7. 모델을 계속 사용하여 다른 이미지를 분석하세요. 더 이상 사용하지 않을 경우 모델을 중지하세요.

10단계: 모델 중지

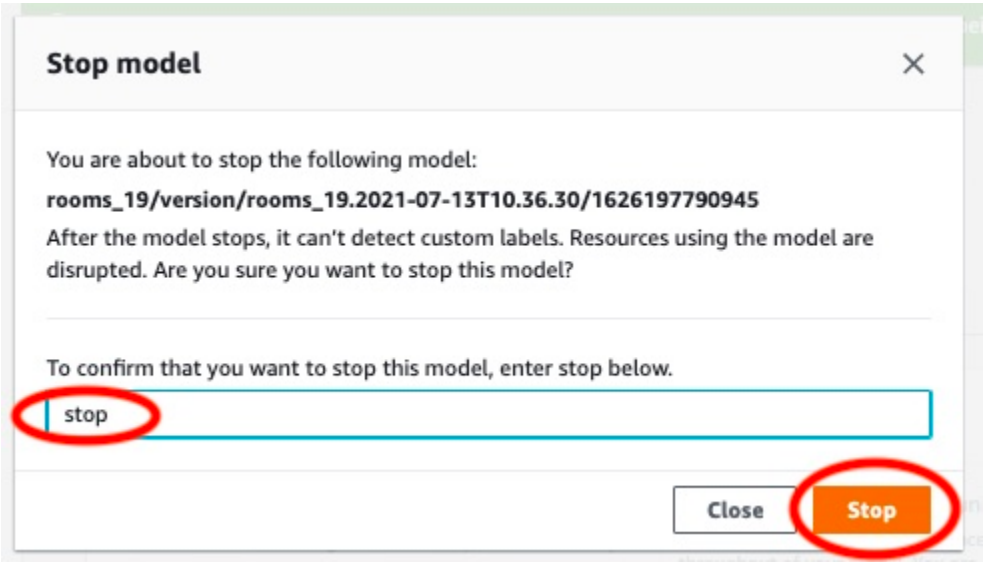
이 단계에서는 모델 실행을 중단합니다. 모델을 실행하는 시간만큼 요금이 부과됩니다. 모델 사용을 마쳤다면 사용을 중지해야 합니다.

모델을 중지하려면

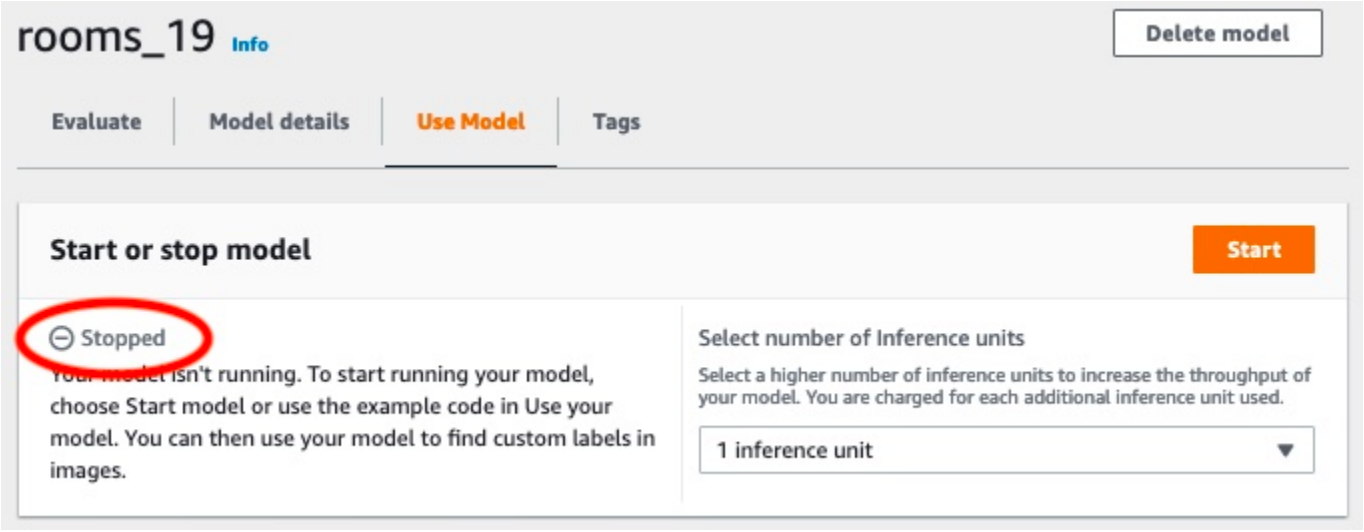
1. 모델 시작 또는 중지 항목에서 중지를 선택합니다.



2. 모델 중지 대화 상자에서 중지를 입력하여 모델 중지를 확인합니다.



3. 중지를 선택하여 모델을 중지합니다. 모델 시작 또는 중지 항목의 상태가 중지됨이면 모델이 중지된 것입니다.


rooms_19 Info
Delete model
Evaluate
Model details
Use Model
Tags
Start or stop model
Start
Stopped
isn't running. To start running your model, choose Start model or use the example code in Use your model. You can then use your model to find custom labels in images.
Select number of Inference units
Select a higher number of inference units to increase the throughput of your model. You are charged for each additional inference unit used.
1 inference unit

Amazon Rekognition Custom Labels 모델 생성

모델은 비즈니스에 고유한 개념, 장면 및 객체를 찾도록 훈련되는 소프트웨어입니다. Amazon Rekognition Custom Labels 콘솔 또는 AWS SDK를 사용하여 모델을 생성할 수 있습니다. Amazon Rekognition Custom Labels 모델을 만들기 전에 Amazon Rekognition Custom Labels의 이해 항목을 먼저 읽어 보는 것이 좋습니다.

이 항목은 프로젝트 생성, 다양한 모델 유형에 대한 훈련 및 테스트 데이터 세트 생성, 모델 훈련에 대한 콘솔 및 SDK 정보를 제공합니다. 이후 항목은 모델을 개선하고 사용하는 방법을 보여줍니다. 콘솔에서 특정 유형의 모델을 생성하고 사용하는 방법을 보여주는 튜토리얼은 튜토리얼: 이미지 분류 항목을 참조하세요.

주제

- 프로젝트 생성
- 훈련 및 테스트 데이터 세트 생성
- Amazon Rekognition Custom Labels 모델 훈련
- 실패한 모델 훈련 디버깅

프로젝트 생성

프로젝트는 모델의 모델 버전, 훈련 데이터 세트, 테스트 데이터 세트를 관리합니다. Amazon Rekognition Custom Labels 콘솔 또는 API를 사용하여 프로젝트를 생성할 수 있습니다. 프로젝트 삭제와 같은 다른 프로젝트 작업에 대해서는 Amazon Rekognition Custom Labels 프로젝트 관리 항목을 참조하세요.

Amazon Rekognition Custom Labels 프로젝트 생성(콘솔)

Amazon Rekognition Custom Labels 콘솔을 사용하여 모델을 생성할 수 있습니다. 새 AWS 리전에서 콘솔을 처음 사용하는 경우 Amazon Rekognition Custom Labels는 AWS 계정에 Amazon S3 버킷(콘솔 버킷)을 생성하도록 요청합니다. 버킷은 프로젝트 파일을 저장하는 데 사용됩니다. 콘솔 버킷을 생성하지 않으면 Amazon Rekognition Custom Labels 콘솔을 사용할 수 없습니다.

Amazon Rekognition Custom Labels 콘솔을 사용하여 모델을 생성할 수 있습니다.

프로젝트 생성 방법(콘솔)

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다.
3. Amazon Rekognition Custom Labels 랜딩 페이지에서 시작하기를 선택합니다.
4. 왼쪽 창에서 프로젝트를 선택합니다.
5. 프로젝트 생성을 선택합니다.
6. 프로젝트 이름에 프로젝트 이름을 입력합니다.
7. 프로젝트를 생성하려면 프로젝트 생성을 선택합니다.
8. 훈련 및 테스트 데이터 세트 생성에 나온 단계에 따라 프로젝트를 위한 훈련 및 테스트 데이터 세트를 만드세요.

Amazon Rekognition Custom Labels 프로젝트 생성(SDK)

CreateProject를 호출하여 Amazon Rekognition Custom Labels 프로젝트를 생성합니다. 응답은 프로젝트를 식별하는 Amazon 리소스 이름(ARN)입니다. 프로젝트를 생성한 후 모델 훈련 및 테스트를 위한 데이터 세트를 생성합니다. 자세한 내용은 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성 섹션을 참조하세요.

프로젝트를 생성하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 코드를 사용하여 프로젝트를 생성합니다.

   AWS CLI

   다음 예제는 프로젝트를 생성하고 해당 ARN을 표시하는 방법을 보여줍니다.

   `project-name`의 값을 생성하려는 프로젝트의 이름으로 변경합니다.

```
aws rekognition create-project --project-name my_project \
 --profile custom-labels-access
```

Python

다음 예제는 프로젝트를 생성하고 해당 ARN을 표시하는 방법을 보여줍니다. 다음 명령줄 인수를 제공하세요.

- project_name: 생성하려는 프로젝트의 이름

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_project(rek_client, project_name):
    """
    Creates an Amazon Rekognition Custom Labels project
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: A name for the new prooject.
    """

    try:
        #Create the project.
        logger.info("Creating project: %s",project_name)

        response=rek_client.create_project(ProjectName=project_name)

        logger.info("project ARN: %s",response['ProjectArn'])

        return response['ProjectArn']


    except ClientError as err:
        logger.exception("Couldn't create project - %s: %s", project_name,
 err.response['Error']['Message'])
        raise

def add_arguments(parser):
```

```
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="A name for the new project."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating project: {args.project_name}")

        # Create the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        project_arn=create_project(rekognition_client,
            args.project_name)

        print(f"Finished creating project: {args.project_name}")
        print(f"ARN: {project_arn}")

    except ClientError as err:
        logger.exception("Problem creating project: %s", err)
        print(f"Problem creating project: {err}")


if __name__ == "__main__":
    main()
```

Java V2

다음 예제는 프로젝트를 생성하고 해당 ARN을 표시하는 방법을 보여줍니다.

다음 명령줄 인수를 제공하세요.

- `project_name`: 생성하려는 프로젝트의 이름

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateProjectRequest;
import software.amazon.awssdk.services.rekognition.model.CreateProjectResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateProject {

    public static final Logger logger =
 Logger.getLogger(CreateProject.class.getName());

    public static String createMyProject(RekognitionClient rekClient, String
 projectName) {

        try {

            logger.log(Level.INFO, "Creating project: {0}", projectName);
            CreateProjectRequest createProjectRequest =
 CreateProjectRequest.builder().projectName(projectName).build();

            CreateProjectResponse response =
 rekClient.createProject(createProjectRequest);

            logger.log(Level.INFO, "Project ARN: {0} ", response.projectArn());
```

```
            return response.projectArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create project: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_name> <bucket> <image>
\n\n" + "Where:\n"
                + "   project_name - A name for the new project\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectName = args[0];
        String projectArn = null;
        ;

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the project
            projectArn = createMyProject(rekClient, projectName);

            System.out.println(String.format("Created project: %s %nProject ARN:
 %s", projectName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
```

```
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. 응답에 표시된 프로젝트 ARN의 이름을 기록해 둡니다. 모델을 생성할 때 필요합니다.
4. 훈련 및 테스트 데이터 세트 생성(SDK) 에 나온 단계에 따라 프로젝트를 위한 훈련 및 테스트 데이터 세트를 만드세요.

훈련 및 테스트 데이터 세트 생성

데이터 세트는 해당 이미지를 설명하는 이미지와 레이블의 집합입니다. 프로젝트에는 훈련 데이터 세트와 테스트 데이터 세트가 필요합니다. Amazon Rekognition Custom Labels는 훈련 데이터 세트를 사용하여 모델을 훈련합니다. Amazon Rekognition Custom Labels는 교육 후 테스트 데이터 세트를 사용하여 훈련된 모델이 올바른 레이블을 얼마나 잘 예측하는지 확인합니다.

Amazon Rekognition 사용자 지정 라벨 콘솔 또는 SDK를 사용하여 데이터세트를 생성할 수 있습니다. AWS 데이터 세트를 생성하기 전에 Amazon Rekognition Custom Labels의 이해 항목을 먼저 읽어보는 것이 좋습니다. 다른 데이터 세트 작업에 대해서는 데이터 세트 관리 항목을 참조하세요.

프로젝트의 훈련 및 테스트 데이터 세트를 만드는 단계는 다음과 같습니다.

프로젝트를 위한 훈련 및 테스트 데이터 세트를 만들려면

1. 훈련 및 테스트 데이터 세트에 레이블을 지정하는 방법을 결정하세요. 자세한 내용은 데이터 세트 목적 설정 항목을 참조하세요.
2. 훈련 및 테스트 데이터 세트에 사용할 이미지를 수집하세요. 자세한 정보는 the section called “이미지 준비”을 참조하세요.
3. 훈련 및 테스트 데이터 세트를 만드세요. 자세한 정보는 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성을 참조하세요. SDK를 사용하는 경우 을 참조하십시오. AWS 훈련 및 테스트 데이터 세트 생성(SDK)
4. 필요한 경우 데이터 세트 이미지에 이미지 수준 레이블 또는 경계 상자를 추가하세요. 자세한 정보는 이미지 레이블 지정을 참조하세요.

데이터 세트를 만든 후 모델을 훈련할 수 있습니다.

주제

- 데이터 세트 목적 설정
- 이미지 준비
- 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성
- 이미지 레이블 지정
- 데이터 세트 디버깅

데이터 세트 목적 설정

프로젝트의 훈련 및 테스트 데이터 세트에 레이블을 지정하는 방법에 따라 생성되는 모델의 유형이 결정됩니다. Amazon Rekognition Custom Labels를 사용하면 다음을 수행하는 모델을 생성할 수 있습니다.

- 객체, 장면 및 개념 찾기
- 객체 위치 찾기
- 브랜드 위치 찾기

객체, 장면 및 개념 찾기

모델은 전체 이미지와 관련된 객체, 장면, 개념을 분류합니다.

이미지 분류와 다중 레이블 분류라는 두 가지 유형의 분류 모델을 만들 수 있습니다. 두 가지 유형의 분류 모델 모두에 대해 모델은 훈련에 사용된 전체 레이블 집합에서 일치하는 레이블을 하나 이상 찾습니다. 훈련 데이터 세트와 테스트 데이터 세트에는 둘 이상의 레이블이 필요합니다.

이미지 분류

모델은 이미지를 미리 정의된 레이블 세트에 속하는 것으로 분류합니다. 예를 들어, 이미지에 거실이 포함되어 있는지 확인하는 모델이 필요할 수 있습니다. 다음 이미지에는 living_space 이미지 수준 레이블이 있을 수 있습니다.

이 유형의 모델에는 훈련 및 테스트 데이터 세트 이미지 각각에 하나의 이미지 수준 레이블을 추가하세요. 예제 프로젝트에 관해서는 이미지 분류 항목을 참조하세요.

다중 레이블 분류

모델은 이미지를 여러 카테고리(예: 꽃 종류, 잎 유무)로 분류합니다. 예를 들어, 다음 이미지에는 mediterranean_spurge 및 no_leaves 이미지 레벨 레이블이 있을 수 있습니다.

이 유형의 모델에는 각 범주의 이미지 수준 레이블을 훈련 및 테스트 데이터 세트 이미지에 할당하세요. 예제 프로젝트에 관해서는 다중 레이블 이미지 분류 항목을 참조하세요.

이미지 수준 레이블 지정

이미지가 Amazon S3 버킷에 저장되어 있는 경우 폴더 이름을 사용하여 이미지 수준 레이블을 자동으로 추가할 수 있습니다. 자세한 정보는 Amazon S3 버킷을 참조하세요. 데이터 세트를 생성한 후 이미지에 이미지 수준 레이블을 추가할 수도 있습니다. 자세한 내용은 the section called “이미지에 이미지 수준 레이블 지정” 섹션을 참조하세요. 필요에 따라 새 레이블을 추가할 수 있습니다. 자세한 정보는 레이블 관리을 참조하세요.

객체 위치 찾기

이미지에서 객체의 위치를 예측하는 모델을 만들려면 훈련 및 테스트 데이터 세트의 이미지에 대한 객체 위치 경계 상자와 레이블을 정의하세요. 경계 상자는 객체를 꼭 둘러싸는 상자입니다. 예를 들어 다음 이미지는 Amazon Echo 및 Amazon Echo Dot 주위의 경계 상자를 보여줍니다. 각 경계 상자에는 지정된 레이블(Amazon Echo 또는 Amazon Echo Dot)이 있습니다.

객체 위치를 찾으려면 데이터 세트에 레이블이 하나 이상 있어야 합니다. 모델 훈련 중에 이미지의 경계 상자 외부 영역을 나타내는 추가 레이블이 자동으로 생성됩니다.

경계 상자 지정

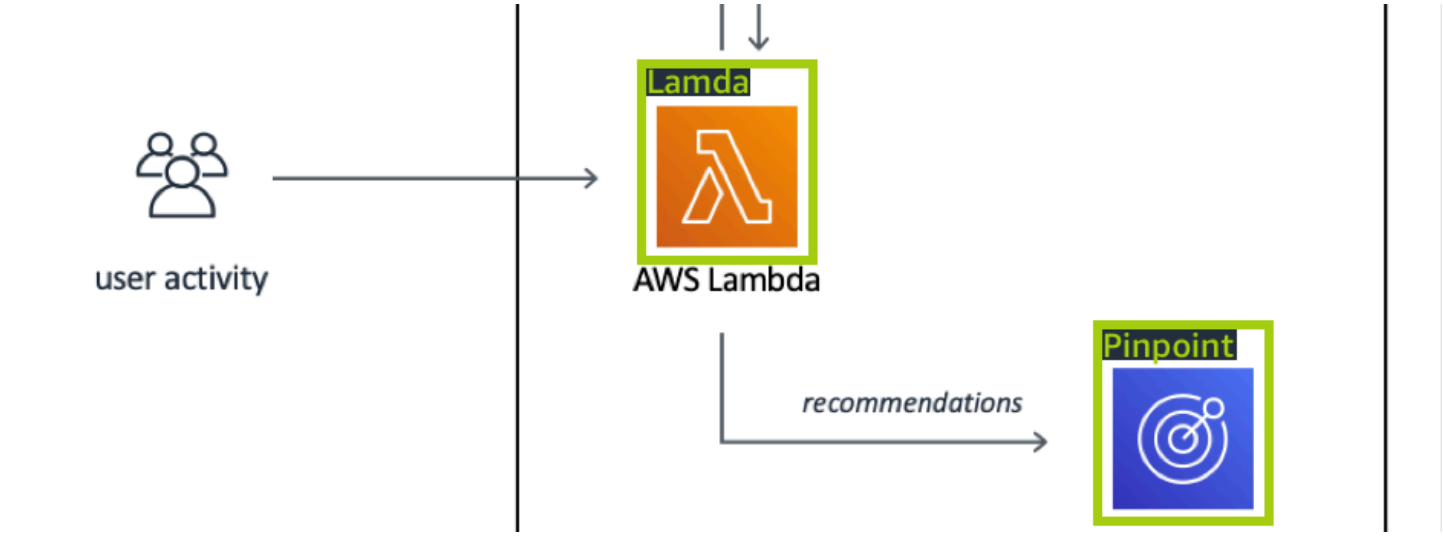
데이터 세트를 만들 때 이미지에 경계 상자 정보를 포함할 수 있습니다. 예를 들어, 경계 상자가 포함된 SageMaker Ground Truth 형식 매니페스트 파일을 가져올 수 있습니다. 데이터 세트를 만든 후 경계 상자를 추가할 수도 있습니다. 자세한 정보는 경계 상자로 객체에 레이블 지정을 참조하세요. 필요에 따라 새 레이블을 추가할 수 있습니다. 자세한 정보는 레이블 관리을 참조하세요.

브랜드 위치 찾기

로고 및 애니메이션 캐릭터 같은 브랜드 위치를 찾으려 한다면 훈련 데이터 세트 이미지에 두 가지 유형의 이미지를 사용할 수 있습니다.

- 로고만 포함된 이미지 각 이미지에는 로고 이름을 나타내는 하나의 이미지 수준 레이블이 필요합니다. 예를 들어, 다음 이미지의 이미지 수준 레이블은 Lambda일 수 있습니다.

- 축구 경기나 건축 도표와 같은 전형적인 위치에 로고가 포함된 이미지 각 훈련 이미지에는 로고의 각 인스턴스를 둘러싸는 경계 상자가 필요합니다. 예를 들어, 다음 이미지는 AWS Lambda 및 Amazon Pinpoint 로고 주변에 레이블이 있는 경계 상자가 있는 아키텍처 다이어그램을 보여줍니다.



훈련 이미지에는 이미지 수준 레이블과 경계 상자를 함께 사용하지 않는 것이 좋습니다.

테스트 이미지에는 찾으려는 브랜드 인스턴스 주위에 경계 상자가 있어야 합니다. 훈련 이미지에 레이블이 지정된 경계 상자가 포함된 경우에만 훈련 데이터 세트를 분할하여 테스트 데이터 세트를 만들 수 있습니다. 훈련 영상에 이미지 수준 레이블만 있는 경우에는 경계 상자 레이블이 지정된 영상이 포함된 테스트 데이터 세트를 만들어야 합니다. 브랜드 위치를 찾도록 모델을 훈련하는 경우 이미지에 레이블을 지정하는 방법에 따라 경계 상자로 객체에 레이블 지정 및 이미지에 이미지 수준 레이블 지정 항목을 수행하세요.

브랜드 감지 예제 프로젝트는 Amazon Rekognition Custom Labels가 레이블이 있는 경계 상자를 사용하여 객체 위치를 찾는 모델을 훈련하는 방법을 보여줍니다.

모델 유형에 대한 레이블 요구 사항

다음 표를 사용하여 이미지에 레이블을 지정하는 방법을 확인하세요.

이미지 수준 레이블과 경계 상자 레이블이 지정된 이미지를 하나의 데이터 세트로 결합할 수 있습니다. 이 경우 Amazon Rekognition Custom Labels는 이미지 수준 모델을 생성할지 아니면 객체 위치 모델을 생성할지 여부를 선택합니다.

예제	훈련 이미지	테스트 이미지
이미지 분류	이미지당 이미지 수준 레이블 1개	이미지당 이미지 수준 레이블 1개

예제	훈련 이미지	테스트 이미지
다중 레이블 분류	이미지당 이미지 수준 레이블 여러 개	이미지당 이미지 수준 레이블 여러 개
브랜드 위치 찾기	이미지 수준 레이블(레이블이 지정된 경계 상자도 사용할 수 있음)	레이블이 지정된 경계 상자
객체 위치 찾기	레이블이 지정된 경계 상자	레이블이 지정된 경계 상자

이미지 준비

훈련 및 테스트 데이터 세트의 이미지에는 모델로 찾고자 하는 객체, 장면 또는 개념이 포함되어 있습니다.

이미지에는 훈련된 모델이 식별해야 할 이미지의 표본이 되는 다양한 배경과 조명이 나타나 있어야 합니다.

이 항목에서는 훈련 및 테스트 데이터 세트의 이미지에 대한 정보를 제공합니다.

이미지 형식

PNG 및 JPEG 형식의 이미지를 사용하여 Amazon Rekognition Custom Labels 모델을 훈련할 수 있습니다. 마찬가지로 `DetectCustomLabels`를 사용하여 사용자 지정 레이블을 감지하려면 PNG 및 JPEG 형식의 이미지가 필요합니다.

이미지 권장 사항 입력

Amazon Rekognition Custom Labels에는 모델을 훈련하고 테스트하기 위한 이미지가 필요합니다. 이미지를 준비하려면 다음을 고려하세요.

- 생성하려는 모델의 특정 도메인을 선택합니다. 예를 들어 경치에 사용할 모델을 선택하고, 기계 부품과 같은 객체에 사용할 모델을 하나 더 선택할 수 있습니다. Amazon Rekognition Custom Labels는 사용자의 이미지가 선택한 도메인에 있는 경우에 가장 잘 작동합니다.
- 최소 10개 이상의 이미지를 사용하여 모델을 훈련하세요.
- 이미지는 PNG 또는 JPEG 형식이어야 합니다.
- 다양한 조명, 배경, 해상도로 객체를 보여주는 이미지를 사용하세요.

- 훈련 및 테스트 이미지는 모델이 감지할 이미지와 비슷해야 합니다.
- 이미지에 지정할 레이블을 결정합니다.
- 이미지 해상도가 충분히 커야 합니다. 자세한 정보는 Amazon Rekognition Custom Labels 지침 및 할당량을 참조하세요.
- 오클루전이 감지하려는 객체를 가리지 않도록 하세요.
- 배경과 충분한 대비를 보이는 이미지를 사용하세요.
- 밝고 선명한 이미지를 사용하세요. 피사체와 카메라 움직임 때문에 흐려진 이미지는 되도록 사용하지 않아야 합니다.
- 객체가 이미지에서 많은 부분을 차지하는 이미지를 사용하세요.
- 테스트 데이터 세트의 이미지는 훈련 데이터 세트에 있는 이미지가 아니어야 합니다. 여기에는 모델이 분석하도록 훈련된 객체, 장면 및 개념이 포함되어야 합니다.

이미지 세트 크기

Amazon Rekognition Custom Labels는 이미지 세트를 사용하여 모델을 훈련합니다. 훈련에는 10개 이상의 이미지를 사용해야 합니다. Amazon Rekognition Custom Labels는 훈련 및 테스트 이미지를 데이터 세트에 저장합니다. 자세한 정보는 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성을 참조하세요.

이미지를 사용하여 훈련 및 테스트 데이터 세트 생성

데이터 세트가 하나인 프로젝트 또는 훈련 데이터 세트와 테스트 데이터 세트가 분리된 프로젝트로 시작할 수 있습니다. 단일 데이터 세트로 시작하는 경우 Amazon Rekognition Custom Labels는 훈련 중에 데이터 세트를 분할하여 프로젝트에 사용할 훈련 데이터 세트(80%)와 테스트 데이터 세트(20%)를 생성합니다. Amazon Rekognition Custom Labels가 훈련 및 테스트에 사용할 이미지를 결정하게 하려면 단일 데이터 세트로 시작하세요. 훈련, 테스트 및 성능 튜닝을 완벽하게 제어하려면 별도의 훈련 및 테스트 데이터 세트로 프로젝트를 시작하는 것이 좋습니다.

다음 위치 중 하나에서 이미지를 가져와서 프로젝트에 대한 훈련 및 테스트 데이터 세트를 만들 수 있습니다.

- Amazon S3 버킷
- 로컬 컴퓨터
- 매니페스트 파일
- 기존 데이터 세트

별도의 훈련 및 테스트 데이터 세트로 프로젝트를 시작하는 경우 데이터 세트마다 다른 소스 위치를 사용할 수 있습니다.

이미지를 가져온 위치에 따라 이미지에 레이블이 지정되지 않을 수 있습니다. 예를 들어 로컬 컴퓨터에서 가져온 이미지에는 레이블이 지정되지 않습니다. Amazon SageMaker Ground Truth 매니페스트 파일에서 가져온 이미지에는 레이블이 지정됩니다. Amazon Rekognition Custom Labels 콘솔을 사용하여 레이블을 추가, 변경 및 할당할 수 있습니다. 자세한 정보는 이미지 레이블 지정을 참조하세요.

이미지 업로드 중에 오류가 발생하거나, 이미지가 누락되었거나, 이미지에 레이블이 누락된 경우 실패한 모델 훈련 디버깅 항목을 읽어보세요.

데이터 세트에 관한 자세한 내용은 데이터 세트 관리 항목을 참조하세요.

훈련 및 테스트 데이터 세트 생성(SDK)

AWS SDK를 사용하여 교육 및 테스트 데이터세트를 생성할 수 있습니다.

훈련 데이터 세트

AWS SDK를 사용하여 다음과 같은 방법으로 훈련 데이터세트를 만들 수 있습니다.

- 제공한 Amazon Sagemaker 형식 매니페스트 파일과 CreateDataset함께 사용하십시오. 자세한 정보는 the section called “매니페스트 파일 생성”을 참조하세요. 예제 코드는 SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기 항목을 참조하세요.
- `CreateDataset`를 사용하여 기존 Amazon Rekognition Custom Labels 데이터 세트를 복사하세요. 예제 코드는 기존 데이터 세트를 사용하여 데이터 세트 생성(SDK) 항목을 참조하세요.
- 를 사용하여 빈 데이터세트를 생성하고 `CreateDataset` 나중에 데이터세트 항목을 추가합니다. UpdateDatasetEntries 빈 데이터 세트를 만들려면 프로젝트에 데이터 세트 추가 항목을 참조하세요. 데이터 세트에 이미지를 추가하려면 더 많은 이미지 추가(SDK) 항목을 참조하세요. 모델을 훈련하려면 먼저 데이터 세트 항목을 추가해야 합니다.

테스트 데이터 세트

AWS SDK를 사용하여 다음과 같은 방법으로 테스트 데이터세트를 만들 수 있습니다.

- 제공한 Amazon Sagemaker 형식 매니페스트 파일과 CreateDataset함께 사용하십시오. 자세한 정보는 the section called “매니페스트 파일 생성”을 참조하세요. 예제 코드는 SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기 항목을 참조하세요.
- `CreateDataset`를 사용하여 기존 Amazon Rekognition Custom Labels 데이터 세트를 복사하세요. 예제 코드는 기존 데이터 세트를 사용하여 데이터 세트 생성(SDK) 항목을 참조하세요.

- `CreateDataset`를 사용하여 빈 데이터 세트를 생성하고 나중에 `UpdateDatasetEntries`로 데이터 세트 항목을 추가합니다. 빈 데이터 세트를 만들려면 프로젝트에 데이터 세트 추가 항목을 참조하세요. 데이터 세트에 이미지를 추가하려면 더 많은 이미지 추가(SDK) 항목을 참조하세요. 모델을 훈련하려면 먼저 데이터 세트 항목을 추가해야 합니다.
- 훈련 데이터 세트를 별도의 훈련 데이터 세트와 테스트 데이터 세트로 분할합니다. 먼저 `CreateDataset`를 사용하여 빈 테스트 데이터 세트를 만듭니다. 그런 다음 를 호출하여 교육 데이터 세트 항목의 20% 를 테스트 데이터 세트로 이동합니다. DistributeDatasetEntries 빈 데이터 세트를 만들려면 프로젝트에 데이터 세트 추가(SDK) 항목을 참조하세요. 훈련 데이터 세트를 분할하려면 훈련 데이터 세트 배포(SDK) 항목을 참조하세요.

Amazon S3 버킷

Amazon S3 버킷에서 이미지를 가져옵니다. 콘솔 버킷 또는 AWS 계정의 다른 Amazon S3 버킷을 사용할 수 있습니다. 콘솔 버킷을 사용하는 경우 필요한 권한은 이미 설정되어 있습니다. 콘솔 버킷을 사용하지 않는 경우 외부 Amazon S3 버킷에 액세스 항목을 참조하세요.

> **Note**
>
> AWS SDK를 사용하여 Amazon S3 버킷의 이미지로부터 직접 데이터세트를 생성할 수는 없습니다. 대신 이미지의 원본 위치를 참조하는 매니페스트 파일을 생성하세요. 자세한 내용은 매니페스트 파일단원을 참조하세요.

데이터 세트를 만드는 동안 이미지가 포함된 폴더의 이름을 기반으로 이미지에 레이블 이름을 할당할 수 있습니다. 폴더는 데이터 세트 생성 시 S3 폴더 위치에 지정하는 Amazon S3 폴더 경로의 하위 폴더여야 합니다. 데이터 세트를 생성하려면 S3 버킷에서 이미지를 가져와서 데이터 세트를 생성합니다. 항목을 참조하세요.

예를 들어 Amazon S3 버킷에 다음과 같은 폴더 구조가 있다고 가정합니다. Amazon S3 폴더 위치를 S3-bucket/alexa-devices로 지정하는 경우, 폴더 echo에 있는 이미지에 echo 레이블이 할당됩니다. 마찬가지로 echo-dot 폴더의 이미지에는 echo-dot라는 레이블이 지정됩니다. 더 깊은 하위 폴더의 이름은 이미지에 레이블을 지정하는 데 사용되지 않습니다. 대신 Amazon S3 폴더 위치의 적절한 하위 폴더가 사용됩니다. 예를 들어 폴더의 이미지에는 echo-dot white-echo-dots레이블이 할당됩니다. S3 폴더 위치 수준의 이미지(alexa-devices)에는 레이블이 지정되어 있지 않습니다.

폴더 구조의 하위 폴더에 더 깊은 S3 폴더 위치를 지정하여 이미지에 레이블을 지정할 수 있습니다. 예를 들어 S3-버킷/Alexa-디바이스/Echo-dot를 지정하면 폴더의 이미지에 레이블이 지정됩니다. white-echo-dotwhite-echo-dot 지정된 s3 폴더 위치 외부의 이미지(예: echo)는 가져올 수 없습니다.

```
S3-bucket
### alexa-devices
    ### echo
    #   ### echo-image-1.png
    #   ### echo-image-2.png
    #   ### .
    #   ### .
    ### echo-dot
        ### white-echo-dot
        #   ### white-echo-dot-image-1.png
        #   ### white-echo-dot-image-2.png
        #
        ### echo-dot-image-1.png
        ### echo-dot-image-2.png
        ### .
        ### .
```

현재 리전에서 콘솔을 처음 열었을 때 Amazon Rekognition에서 생성한 Amazon S3 버킷 (콘솔 버킷)을 사용하는 것이 좋습니다. AWS 사용 중인 Amazon S3 버킷이 콘솔 버킷과 다른 경우(외부), 콘솔은 데이터 세트 생성 중에 적절한 권한을 설정하라는 메시지를 표시합니다. 자세한 정보는 the section called “2단계: 콘솔 권한 설정”을 참조하세요.

S3 버킷에서 이미지를 가져와서 데이터 세트를 생성합니다.

다음 절차는 콘솔 S3 버킷에 저장된 이미지를 사용하여 데이터 세트를 생성하는 방법을 보여줍니다. 이미지는 저장되는 폴더의 이름으로 레이블이 자동 지정됩니다.

이미지를 가져온 후에는 데이터 세트의 갤러리 페이지에서 이미지를 더 추가하고, 레이블을 지정하고, 경계 상자를 추가할 수 있습니다. 자세한 정보는 이미지 레이블 지정을 참조하세요.

이미지를 Amazon Simple Storage Service 버킷에 업로드합니다.

1. 로컬 파일 시스템에 폴더를 생성합니다. alexa-devices 같은 폴더 이름을 사용하세요.
2. 방금 만든 폴더 내에 사용하려는 각 레이블의 이름을 딴 폴더를 생성합니다. echo 및 echo-dot를 예로 들 수 있습니다. 폴더 구조는 다음과 비슷한 모습이어야 합니다.

   ```
   alexa-devices
   ### echo
   #   ### echo-image-1.png
   #   ### echo-image-2.png
   ```

```
#   ### .
#   ### .
### echo-dot
    ### echo-dot-image-1.png
    ### echo-dot-image-2.png
    ### .
    ### .
```

3. 레이블에 상응하는 이미지를 레이블 이름이 같은 폴더에 넣습니다.
4. AWS Management Console 로그인하고 https://console.aws.amazon.com/s3/ 에서 Amazon S3 콘솔을 엽니다.
5. 최초 설정 시 Amazon Rekognition Custom Labels가 생성한 Amazon S3 버킷(콘솔 버킷)에 1단계에서 생성한 폴더를 추가하세요. 자세한 정보는 Amazon Rekognition Custom Labels 프로젝트 관리을 참조하세요.
6. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
7. 사용자 지정 레이블 사용을 선택합니다.
8. Get started를 선택합니다.
9. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
10. 프로젝트 페이지에서 데이터 세트에 추가하려는 프로젝트를 선택합니다. 프로젝트 세부 정보 페이지가 열립니다.
11. 데이터 세트 생성을 선택합니다. 데이터 세트 생성 페이지가 표시됩니다.
12. 시작 구성에서 단일 데이터 세트로 시작 또는 훈련 데이터 세트로 시작을 선택합니다. 더 높은 품질의 모델을 만들려면 별도의 훈련 및 테스트 데이터 세트로 시작하는 것이 좋습니다.

    Single dataset

    a. 훈련 데이터 세트 세부 정보 항목에서 S3 버킷에서 이미지 가져오기를 선택합니다.
    b. 훈련 데이터 세트 세부 정보 항목에서 이미지 소스 구성 항목의 13~15단계를 위한 정보를 입력합니다.

    Separate training and test datasets

    a. 훈련 데이터 세트 세부 정보 항목에서 S3 버킷에서 이미지 가져오기를 선택합니다.
    b. 훈련 데이터 세트 세부 정보 항목에서 이미지 소스 구성 항목의 13~15단계를 위한 정보를 입력합니다.
    c. 테스트 데이터 세트 세부 정보 항목에서 S3 버킷에서 이미지 가져오기를 선택합니다.

d. 테스트 데이터 세트 세부 정보 항목에서 이미지 소스 구성 항목의 13~15단계를 위한 정보를 입력합니다.

13. Amazon S3 버킷에서 이미지 가져오기를 선택합니다.
14. S3 URI에 Amazon S3 버킷 위치 및 폴더 경로를 입력합니다.
15. 폴더를 기반으로 이미지에 레이블 자동 지정을 선택합니다.
16. 데이터 세트 생성을 선택합니다. 프로젝트의 데이터 세트 페이지가 열립니다.
17. 레이블을 추가하거나 변경해야 하면 이미지 레이블 지정 항목을 수행합니다.
18. 모델 훈련(콘솔)에 나온 단계에 따라 모델을 훈련하세요.

로컬 컴퓨터

이미지는 컴퓨터에서 직접 로드됩니다. 한 번에 최대 30개의 이미지를 업로드할 수 있습니다.

업로드하는 이미지에는 연결된 레이블이 없을 것입니다. 자세한 정보는 이미지 레이블 지정을 참조하세요. 업로드할 이미지가 많은 경우 Amazon S3 버킷을 사용하는 것이 좋습니다. 자세한 정보는 Amazon S3 버킷을 참조하세요.

> **Note**
>
> AWS SDK를 사용하여 로컬 이미지가 포함된 데이터세트를 생성할 수 없습니다. 대신 매니페스트 파일을 생성하여 Amazon S3 버킷에 이미지를 업로드합니다. 자세한 정보는 매니페스트 파일을 참조하세요.

로컬 컴퓨터의 이미지를 사용하여 데이터 세트를 만들려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 데이터 세트에 추가하려는 프로젝트를 선택합니다. 프로젝트 세부 정보 페이지가 열립니다.
6. 데이터 세트 생성을 선택합니다. 데이터 세트 생성 페이지가 표시됩니다.
7. 시작 구성에서 단일 데이터 세트로 시작 또는 훈련 데이터 세트로 시작을 선택합니다. 더 높은 품질의 모델을 만들려면 별도의 훈련 및 테스트 데이터 세트로 시작하는 것이 좋습니다.

Single dataset

a. 훈련 데이터 세트 세부 정보 항목에서 컴퓨터에서 이미지 업로드를 선택합니다.

b. 데이터 세트 생성을 선택합니다.

c. 프로젝트의 데이터 세트 페이지에서 이미지 추가를 선택합니다.

d. 사용자의 컴퓨터 파일에서 온 데이트 세트에 업로드할 이미지를 선택합니다. 이미지를 드래그하거나 로컬 컴퓨터에서 업로드할 이미지를 선택할 수 있습니다.

e. 이미지 업로드를 선택합니다.

Separate training and test datasets

a. 훈련 데이터 세트 세부 정보 항목에서 컴퓨터에서 이미지 업로드를 선택합니다.

b. 테스트 데이터 세트 세부정보 섹션에서 컴퓨터에서 이미지 업로드를 선택합니다.

> **Note**
>
> 훈련 데이터 세트와 테스트 데이터 세트에는 서로 다른 이미지 소스가 있을 수 있습니다.

c. 데이터 세트 생성을 선택합니다. 각 데이터 세트에 대한 훈련 탭과 테스트 탭이 있는 사용자 프로젝트의 데이터 세트 페이지가 나타납니다.

d. 작업을 선택한 다음 훈련 데이터 세트에 이미지 추가를 선택합니다.

e. 데이터 세트에 업로드하려는 이미지를 선택합니다. 이미지를 드래그하거나 로컬 컴퓨터에서 업로드할 이미지를 선택할 수 있습니다.

f. 이미지 업로드를 선택합니다.

g. 5e~5g단계를 반복합니다. 5e단계에서는 작업을 선택한 다음 테스트 데이터 세트에 이미지 추가를 선택합니다.

8. 이미지 레이블 지정에 나온 단계에 따라 이미지에 레이블을 지정합니다.

9. 모델 훈련(콘솔)에 나온 단계에 따라 모델을 훈련하세요.

매니페스트 파일

Amazon SageMaker Ground Truth 형식 매니페스트 파일을 사용하여 데이터세트를 생성할 수 있습니다. Amazon SageMaker Ground Truth 작업의 매니페스트 파일을 사용할 수 있습니다. 이미지와 레이

블이 SageMaker Ground Truth 매니페스트 파일 형식이 아닌 경우 형식 매니페스트 파일을 생성하여 레이블이 지정된 이미지를 가져오는 데 사용할 수 있습니다. SageMaker

주제

- SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)
- SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기
- Amazon SageMaker Ground Truth 채용공고
- 매니페스트 파일 생성
- 다른 데이터 세트 형식을 매니페스트 파일로 변환

SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)

다음 절차는 SageMaker Ground Truth 형식 매니페스트 파일을 사용하여 데이터세트를 만드는 방법을 보여줍니다.

1. 다음 중 하나를 수행하여 훈련 데이터 세트의 매니페스트 파일을 생성합니다.

   - 의 지침에 따라 SageMaker GroundTruth Job으로 매니페스트 파일을 생성합니다. Amazon SageMaker Ground Truth 채용공고
   - 매니페스트 파일 생성의 지침에 따라 사용자 고유의 매니페스트 파일을 생성하세요.

   테스트 데이터 세트를 만들려면 1단계를 반복하여 테스트 데이터 세트를 만드세요.
2. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
3. 사용자 지정 레이블 사용을 선택합니다.
4. Get started를 선택합니다.
5. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
6. 프로젝트 페이지에서 데이터 세트에 추가하려는 프로젝트를 선택합니다. 프로젝트 세부 정보 페이지가 열립니다.
7. 데이터 세트 생성을 선택합니다. 데이터 세트 생성 페이지가 표시됩니다.
8. 시작 구성에서 단일 데이터 세트로 시작 또는 훈련 데이터 세트로 시작을 선택합니다. 더 높은 품질의 모델을 만들려면 별도의 훈련 및 테스트 데이터 세트로 시작하는 것이 좋습니다.

Single dataset

a. 교육 데이터세트 세부 정보 섹션에서 SageMaker Ground Truth로 레이블이 지정된 이미지 가져오기를 선택합니다.

b. 매니페스트 파일 위치에 1단계에서 생성한 매니페스트 파일의 위치를 입력하세요.

c. 데이터 세트 생성을 선택합니다. 프로젝트의 데이터 세트 페이지가 열립니다.

Separate training and test datasets

a. 교육 데이터세트 세부 정보 섹션에서 SageMaker Ground Truth로 레이블이 지정된 이미지 가져오기를 선택합니다.

b. .manifest 파일 위치에 1단계에서 생성한 훈련 데이터 세트 매니페스트 파일의 위치를 입력하세요.

c. 테스트 데이터세트 세부정보 섹션에서 SageMaker Ground Truth로 레이블이 지정된 이미지 가져오기를 선택합니다.

> **Note**
>
> 학습 데이터 세트와 테스트 데이터 세트에는 서로 다른 이미지 소스가 있을 수 있습니다.

d. .manifest 파일 위치에 1단계에서 생성한 테스트 데이터 세트 매니페스트 파일의 위치를 입력하세요.

e. 데이터 세트 생성을 선택합니다. 프로젝트의 데이터 세트 페이지가 열립니다.

9. 레이블을 추가하거나 변경해야 하면 이미지 레이블 지정 항목을 수행합니다.

10. 모델 훈련(콘솔)에 나온 단계에 따라 모델을 훈련하세요.

SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기

다음 절차는 API를 사용하여 매니페스트 파일에서 훈련 또는 테스트 데이터세트를 만드는 방법을 보여줍니다. CreateDataset

SageMaker Ground Truth 작업의 출력과 같은 기존 매니페스트 파일을 사용하거나 자체 매니페스트 파일을 생성할 수 있습니다.

1. 아직 설치하지 않았다면 및 SDK를 설치하고 구성하세요. AWS CLI AWS 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 다음 중 하나를 수행하여 훈련 데이터 세트의 매니페스트 파일을 생성합니다.

   - 의 지침에 따라 SageMaker GroundTruth Job으로 매니페스트 파일을 생성합니다. Amazon SageMaker Ground Truth 채용공고
   - 매니페스트 파일 생성의 지침에 따라 사용자 고유의 매니페스트 파일을 생성하세요.

   테스트 데이터 세트를 만들려면 2단계를 반복하여 테스트 데이터 세트를 만드세요.
3. 다음 예제 코드를 사용하여 훈련 및 테스트 데이터 세트를 만드세요.

   AWS CLI

   다음 코드를 사용하여 데이터 세트를 생성하세요. 다음을 바꿉니다.

   - `project_arn`: 테스트 데이터 세트를 추가하려는 프로젝트의 ARN입니다.
   - `type`: 생성하려는 데이터 세트의 유형(훈련 또는 테스트)
   - `bucket`: 데이터 세트의 매니페스트 파일이 들어 있는 버킷
   - `manifest_file`: 매니페스트의 경로 및 파일 이름

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type type \
     --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
    "Name": "manifest_file" } } }' \
     --profile custom-labels-access
   ```

   Python

   다음 값을 사용하여 데이터 세트를 생성하세요. 다음 명령줄 파라미터를 제공하세요.

   - `project_arn`: 테스트 데이터 세트를 추가하려는 프로젝트의 ARN
   - `dataset_type`: 생성하려는 데이터 세트 유형(`train` 또는 `test`)
   - `bucket`: 데이터 세트의 매니페스트 파일이 들어 있는 버킷
   - `manifest_file`: 매니페스트의 경로 및 파일 이름

   ```
   #Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
   ```

```
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import time
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset(rek_client, project_arn, dataset_type, bucket,
 manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param bucket: The S3 bucket that contains the manifest file.
    :param manifest_file: The path and filename of the manifest file.
    """

    try:
        #Create the project
        logger.info("Creating %s dataset for project %s",dataset_type,
 project_arn)

        dataset_type = dataset_type.upper()

        dataset_source = json.loads(
            '{ "GroundTruthManifest": { "S3Object": { "Bucket": "'
            + bucket
            + '", "Name": "'
            + manifest_file
            + '" } } }'
        )

        response = rek_client.create_dataset(
```

```
            ProjectArn=project_arn, DatasetType=dataset_type,
 DatasetSource=dataset_source
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s",dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
                logger.info("Creating dataset: %s ",dataset_arn)
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn


    except ClientError as err:
        logger.exception("Couldn't create dataset: %s",err.response['Error']
['Message'])
        raise
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "bucket", help="The S3 bucket that contains the manifest file."
    )

    parser.add_argument(
        "manifest_file", help="The path and filename of the manifest file."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        #Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        #Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_dataset(rekognition_client,
```

```
            args.project_arn,
            args.dataset_type,
            args.bucket,
            args.manifest_file)

        print(f"Finished creating dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")



if __name__ == "__main__":
    main()
```

Java V2

다음 값을 사용하여 데이터 세트를 생성하세요. 다음 명령줄 파라미터를 제공하세요.

- `project_arn`: 테스트 데이터 세트를 추가하려는 프로젝트의 ARN
- `dataset_type`: 생성하려는 데이터 세트 유형(`train` 또는 `test`)
- `bucket`: 데이터 세트의 매니페스트 파일이 들어 있는 버킷
- `manifest_file`: 매니페스트의 경로 및 파일 이름

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
```

```
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetManifestFiles {

    public static final Logger logger =
 Logger.getLogger(CreateDatasetManifestFiles.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String bucket, String name) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 s3://{2}/{3} ",
                    new Object[] { datasetType, projectArn, bucket, name });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Could not create dataset. Unrecognized
 dataset type: {0}", datasetType);
                throw new Exception("Could not create dataset. Unrecognized
 dataset type: " + datasetType);

            }
```

```
            GroundTruthManifest groundTruthManifest =
GroundTruthManifest.builder()

.s3Object(S3Object.builder().bucket(bucket).name(name).build()).build();

            DatasetSource datasetSource =
DatasetSource.builder().groundTruthManifest(groundTruthManifest).build();

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;
```

```
                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String bucket = null;
        String name = null;
        String projectArn = null;
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
```

```
                + "   bucket - the S3 bucket that contains the manifest file.\n
\n"
                + "   name - the location and name of the manifest file within
 the bucket.\n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        bucket = args[2];
        name = args[3];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


             // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 bucket, name);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

4. 레이블을 추가하거나 변경해야 하면 레이블 관리(SDK) 항목을 참조하세요.
5. 모델 훈련(SDK)에 나온 단계에 따라 모델을 훈련하세요.

Amazon SageMaker Ground Truth 채용공고

Amazon SageMaker Ground Truth를 사용하면 레이블이 지정된 이미지 세트를 생성할 수 있는 기계 학습과 함께 Amazon Mechanical Turk 또는 사내 개인 인력을 활용할 수 있습니다. Amazon Rekognition 사용자 지정 레이블은 지정한 Amazon S3 버킷에서 SageMaker Ground Truth 매니페스트 파일을 가져옵니다.

Amazon Rekognition 사용자 지정 레이블은 다음과 같은 SageMaker Ground Truth 작업을 지원합니다.

- 이미지 분류
- 경계 상자

가져오는 파일은 이미지와 매니페스트 파일입니다. 매니페스트 파일에는 가져온 이미지에 대한 레이블 및 경계 상자 정보가 들어 있습니다.

Amazon Rekognition은 이미지가 저장되는 Amazon S3 버킷에 액세스할 수 있는 권한이 필요합니다. Amazon Rekognition Custom Labels가 자동으로 설정한 콘솔 버킷을 사용하는 경우, 필요한 권한은 이미 설정되어 있습니다. 콘솔 버킷을 사용하지 않는 경우 외부 Amazon S3 버킷에 액세스 항목을 참조하세요.

SageMaker Ground Truth 작업으로 매니페스트 파일 생성 (콘솔)

다음 절차는 SageMaker Ground Truth 작업으로 레이블이 지정된 이미지를 사용하여 데이터세트를 만드는 방법을 보여줍니다. 작업 출력 파일은 Amazon Rekognition Custom Labels 콘솔 버킷에 저장됩니다.

SageMaker Ground Truth 작업 (콘솔) 으로 레이블이 지정된 이미지를 사용하여 데이터세트를 만들려면

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/s3/ 에서 Amazon S3 콘솔을 엽니다.
2. 콘솔 버킷에서 훈련 이미지를 보관할 폴더를 생성합니다.

> **Note**
>
> 콘솔 버킷은 지역에서 Amazon Rekognition 사용자 지정 레이블 콘솔을 처음 열 때 생성됩니다. AWS 자세한 정보는 Amazon Rekognition Custom Labels 프로젝트 관리을 참조하세요.

3. 방금 생성한 폴더에 이미지를 업로드합니다.
4. 콘솔 버킷에서 Ground Truth 작업의 출력을 보관할 폴더를 생성합니다.
5. https://console.aws.amazon.com/sagemaker/ 에서 SageMaker 콘솔을 엽니다.
6. Ground Truth 레이블 지정 작업을 생성하세요. 2단계와 4단계에서 생성한 폴더의 Amazon S3 URL이 필요합니다. 자세한 내용은 데이터 레이블 지정을 위한 Amazon SageMaker Ground Truth 사용을 참조하십시오.
7. 4단계에서 생성한 폴더의 `output.manifest` 파일 위치를 기록해 두세요. 해당 파일은 하위 폴더 *`Ground-Truth-Job-Name`*`/manifests/output`에 있을 것입니다.
8. SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)의 지침에 따라 업로드된 매니페스트 파일로 데이터 세트를 생성하세요. 8단계로 .manifest 파일 위치에 이전 단계에서 기록해 둔 위치의 Amazon S3 URL을 입력합니다. AWS SDK를 사용 중이라면 그렇게 SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기 하십시오.
9. 1~6단계를 반복하여 테스트 데이터세트에 대한 SageMaker Ground Truth 작업을 생성합니다.

매니페스트 파일 생성

SageMaker Ground Truth 형식 매니페스트 파일을 가져와서 테스트 또는 교육 데이터세트를 만들 수 있습니다. 이미지에 Ground Truth 매니페스트 파일이 아닌 형식으로 레이블이 지정된 경우 다음 정보를 사용하여 SageMaker Ground Truth 형식 매니페스트 파일을 만드십시오. SageMaker

매니페스트 파일은 JSON 라인 형식이며, 각 라인은 이미지의 레이블 정보를 나타내는 완전한 JSON 객체입니다. Amazon Rekognition 사용자 지정 레이블은 JSON 라인을 사용하여 다음과 같은 형식의 SageMaker Ground Truth 매니페스트를 지원합니다.

- 분류 작업 출력: 이미지에 이미지 수준 레이블을 추가하는 데 사용합니다. 이미지 수준 레이블은 이미지에 있는 장면, 개념 또는 객체(객체 위치 정보가 필요하지 않은 경우)의 클래스를 정의합니다. 이미지에는 이미지 수준 레이블이 두 개 이상 있을 수 있습니다. 자세한 정보는 매니페스트 파일의 이미지 수준 레이블을 참조하세요.

- 경계 상자 작업 출력: 이미지에 있는 하나 이상의 객체의 클래스와 위치에 레이블을 지정하는 데 사용합니다. 자세한 정보는 매니페스트 파일의 객체 위치 파악을 참조하세요.

이미지 수준 및 위치 파악(경계 상자) JSON 라인을 동일한 매니페스트 파일에 연결할 수 있습니다.

> **Note**
>
> 이 항목의 JSON 라인 예제는 가독성을 위해 형식이 지정되었습니다.

매니페스트 파일을 가져올 때 Amazon Rekognition Custom Labels는 제한, 구문 및 시맨틱에 대한 검증 규칙을 적용합니다. 자세한 정보는 매니페스트 파일의 검증 규칙을 참조하세요.

매니페스트 파일이 참조하는 이미지는 동일한 Amazon S3 버킷에 있어야 합니다. 매니페스트 파일은 이미지를 저장하는 Amazon S3 버킷과 다른 Amazon S3 버킷에 있을 수 있습니다. JSON 라인의 `source-ref` 필드에 이미지 위치를 지정합니다.

Amazon Rekognition은 이미지가 저장되는 Amazon S3 버킷에 액세스할 수 있는 권한이 필요합니다. Amazon Rekognition Custom Labels가 자동으로 설정한 콘솔 버킷을 사용하는 경우, 필요한 권한은 이미 설정되어 있습니다. 콘솔 버킷을 사용하지 않는 경우 외부 Amazon S3 버킷에 액세스 항목을 참조하세요.

주제

- 매니페스트 파일 생성
- 매니페스트 파일의 이미지 수준 레이블
- 매니페스트 파일의 객체 위치 파악
- 매니페스트 파일의 검증 규칙

매니페스트 파일 생성

다음 절차는 훈련 및 테스트 데이터 세트가 포함된 프로젝트를 생성합니다. 데이터 세트는 사용자가 만든 훈련 및 테스트 매니페스트 파일에서 생성됩니다.

SageMaker Ground Truth 형식 매니페스트 파일을 사용하여 데이터세트를 만들려면 (콘솔)

1. 콘솔 버킷에서 매니페스트 파일을 보관할 폴더를 생성합니다.

2. 콘솔 버킷에서 이미지를 보관할 폴더를 생성합니다.
3. 방금 생성한 폴더에 이미지를 업로드합니다.
4. 훈련 데이터세트를 위한 SageMaker Ground Truth 형식 매니페스트 파일을 만드세요. 자세한 내용은 매니페스트 파일의 이미지 수준 레이블 및 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

   > **Important**
   >
   > 각 JSON 라인의 `source-ref` 필드 값은 업로드한 이미지에 매핑되어야 합니다.

5. 테스트 데이터세트를 위한 SageMaker Ground Truth 형식 매니페스트 파일을 만드세요.
6. 방금 생성한 폴더에 매니페스트 파일을 업로드합니다.
7. 매니페스트 파일의 위치를 기록해 둡니다.
8. SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)의 지침에 따라 업로드된 매니페스트 파일로 데이터 세트를 생성하세요. 8단계로 .manifest 파일 위치에 이전 단계에서 기록해 둔 위치의 Amazon S3 URL을 입력합니다. AWS SDK를 사용 중이라면 그렇게 하세요. SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기

매니페스트 파일의 이미지 수준 레이블

이미지 수준 레이블 (현지화 정보가 필요하지 않은 장면, 개념 또는 객체로 레이블이 지정된 이미지) 을 가져오려면 SageMaker Ground Truth Classification Job Output 형식 JSON 라인을 매니페스트 파일에 추가합니다. 매니페스트 파일은 가져오려는 이미지당 하나씩, 하나 이상의 JSON 라인으로 구성됩니다.

> **Tip**
>
> 매니페스트 파일 생성을 단순화하기 위해 CSV 파일에서 매니페스트 파일을 만드는 Python 스크립트가 제공됩니다. 자세한 정보는 CSV 파일로 매니페스트 파일 생성을 참조하세요.

이미지 수준 레이블을 위한 매니페스트 파일 생성

1. 빈 텍스트 파일을 생성합니다.
2. 가져올 각 이미지에 JSON 라인을 추가합니다. 각 JSON 라인은 다음과 비슷한 모습이어야 합니다.

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-
metadata":{"confidence":0.95,"job-name":"labeling-job/
testclconsolebucket","class-name":"Echo Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

3. 파일을 저장합니다. `.manifest` 확장을 사용할 수 있지만 필수는 아닙니다.
4. 생성한 매니페스트 파일을 사용하여 데이터 세트를 생성하세요. 자세한 정보는 SageMaker Ground Truth 형식 매니페스트 파일을 사용하여 데이터세트를 만들려면 (콘솔)을 참조하세요.

이미지 레벨 JSON 라인

이 항목은 하나의 이미지에 JSON 라인을 생성하는 방법을 보여줍니다. 다음 이미지를 고려하세요. 다음 이미지의 장면은 Sunrise라고 부르겠습니다.

Sunrise 장면이 포함된 이전 이미지의 JSON 라인은 다음과 같을 수 있습니다.

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2020-03-06T17:46:39.176",
        "type": "groundtruth/image-classification"
    }
}
```

다음 정보를 참고하세요.

source-ref

(필수) 이미지의 Amazon S3 위치입니다. 형식은 "s3://*BUCKET*/*OBJECT_PATH*"입니다. 가져온 데이터 세트의 이미지는 동일한 Amazon S3 버킷에 저장되어야 합니다.

testdataset-classification_Sunrise

(필수) 레이블 속성 필드 이름을 선택합니다. 필드 값(위 예제의 1)은 레이블 속성 식별자입니다. Amazon Rekognition Custom Labels에서는 사용되지 않으며 임의의 정수 값일 수 있습니다. -metadata가 추가된 필드 이름으로 식별되는 상응하는 메타데이터가 있어야 합니다. 예: "testdataset-classification_Sunrise-metadata"

testdataset-classification_Sunrise-metadata

(필수) 레이블 속성에 대한 메타데이터 필드 이름은 -metadata가 추가된 레이블 속성과 동일해야 합니다.

confidence

(필수) Amazon Rekognition Custom Labels에서는 현재 사용되지 않지만 0에서 1 사이의 값을 제공해야 합니다.

job-name

(선택 사항) 이미지를 처리하는 작업에 원하는 이름을 붙이세요.

class-name

(필수) 이미지에 적용되는 장면이나 개념에 원하는 클래스 이름을 붙이세요. 예를 들어 "Sunrise"입니다.

human-annotated

(필수) 사람이 주석을 완성했으면 "yes"를 지정하세요. 그렇지 않을 경우 "no"입니다.

creation-date

(필수) 레이블이 생성된 협정 세계시(UTC) 날짜와 시간

type

(필수) 이미지에 적용해야 하는 처리 유형 이미지 수준 레이블의 경우 값은 "groundtruth/image-classification"입니다.

이미지에 여러 이미지 수준 레이블 추가

이미지에 레이블을 여러 개 추가할 수 있습니다. 예를 들어 다음 JSON은 하나의 이미지에 축구와 공이라는 두 개의 레이블을 추가합니다.

```
{
    "source-ref": "S3 bucket location",
    "sport0":0, # FIRST label
    "sport0-metadata": {
        "class-name": "football",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    },
    "sport1":1, # SECOND label
    "sport1-metadata": {
        "class-name": "ball",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    }
}  # end of annotations for 1 image
```

매니페스트 파일의 객체 위치 파악

SageMakerGround Truth Bounding Box Job Output 형식의 JSON 라인을 매니페스트 파일에 추가하여 객체 위치 정보가 표시된 이미지를 가져올 수 있습니다.

위치 파악 정보는 이미지 상의 객체 위치를 나타냅니다. 위치는 객체를 둘러싸는 경계 상자로 표시됩니다. 경계 상자 구조에는 경계 상자의 왼쪽 위 좌표와 경계 상자의 너비 및 높이가 포함됩니다. 경계 상자 형식의 JSON 라인에는 이미지에 있는 각 객체의 클래스와 이미지에 있는 하나 이상의 객체 위치에 대한 경계 상자가 포함됩니다.

매니페스트 파일은 하나 이상의 JSON 라인으로 구성되며, 각 라인에는 하나의 이미지에 대한 정보가 들어 있습니다.

객체 위치 파악을 위한 매니페스트 파일을 만들려면

1. 빈 텍스트 파일을 생성합니다.

2. 가져올 각 이미지에 JSON 라인을 추가합니다. 각 JSON 라인은 다음과 비슷한 모습이어야 합니다.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id":
 1, "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0,
 "top": 65, "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata":
 {"objects": [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo",
 "1": "Echo Dot"}, "type": "groundtruth/object-detection", "human-annotated":
 "yes", "creation-date": "2013-11-18T02:53:27", "job-name": "my job"}}
```

3. 파일을 저장합니다. `.manifest` 확장을 사용할 수 있지만 필수는 아닙니다.
4. 방금 생성한 파일을 사용하여 데이터 세트를 생성합니다. 자세한 정보는 SageMaker Ground Truth 형식 매니페스트 파일을 사용하여 데이터세트를 만들려면 (콘솔)을 참조하세요.

객체 경계 상자 JSON 라인

이 항목은 하나의 이미지에 JSON 라인을 생성하는 방법을 보여줍니다. 다음 이미지는 Amazon Echo 및 Amazon Echo Dot 디바이스 주변의 경계 상자를 보여줍니다.

다음은 이전 이미지의 경계 상자 JSON 라인입니다.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
```

```
    "height": 101
   }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2013-11-18T02:53:27",
   "job-name": "my job"
  }
}
```

다음 정보를 참고하세요.

source-ref

(필수) 이미지의 Amazon S3 위치입니다. 형식은 "s3://*BUCKET*/*OBJECT_PATH*"입니다. 가져온 데이터 세트의 이미지는 동일한 Amazon S3 버킷에 저장되어야 합니다.

bounding-box

(필수) 레이블 속성 필드 이름을 선택합니다. 이미지에서 감지된 각 객체의 이미지 크기 및 경계 상자를 포함합니다. -metadata가 추가된 필드 이름으로 식별되는 상응하는 메타데이터가 있어야 합니다. 예: "bounding-box-metadata"

image_size

(필수) 이미지 크기(픽셀 단위)를 포함하는 단일 요소 배열

- height: (필수) 이미지의 높이(픽셀 단위)

- width: (필수) 이미지의 깊이(픽셀 단위)
- depth: (필수) 이미지의 채널 수 RGB 이미지의 경우 값은 3입니다. Amazon Rekognition Custom Labels에서는 현재 사용하지 않지만 값이 필요합니다.

주석

(필수) 이미지에서 감지된 각 객체에 대한 경계 상자 정보의 배열

- class_id: (필수) class-map의 레이블에 매핑됩니다. 위 예제에서 class_id가 1인 객체는 이미지의 Echo Dot입니다.
- top: (필수) 이미지 상단에서 경계 상자 상단까지의 거리(픽셀 단위)
- left: (필수) 이미지 왼쪽에서 경계 상자 왼쪽까지의 거리(픽셀 단위)
- width: (필수) 경계 상자의 너비(픽셀 단위)
- height: (필수) 경계 상자의 높이(픽셀 단위)

bounding-box-metadata

(필수) 레이블 속성에 대한 메타데이터 필드 이름은 -metadata가 추가된 레이블 속성과 동일해야 합니다. 이미지에서 감지된 각 객체에 대한 경계 상자 정보의 배열

Objects

(필수) 이미지에 있는 객체의 배열입니다. 인덱스를 기준으로 주석 배열에 매핑합니다. Amazon Rekognition Custom Labels는 신뢰도 속성을 사용하지 않습니다.

class-map

(필수) 이미지에서 감지된 객체에 적용되는 클래스 맵

type

(필수) 분류 작업 유형입니다. `"groundtruth/object-detection"`은 작업을 객체 감지로 식별합니다.

creation-date

(필수) 레이블이 생성된 협정 세계시(UTC) 날짜와 시간

human-annotated

(필수) 사람이 주석을 완성했으면 `"yes"`를 지정하세요. 그렇지 않을 경우 `"no"`입니다.

job-name

(선택 사항) 이미지를 처리하는 작업의 이름입니다.

매니페스트 파일의 검증 규칙

매니페스트 파일을 가져올 때 Amazon Rekognition Custom Labels는 제한, 구문 및 시맨틱에 대한 검증 규칙을 적용합니다. SageMaker Ground Truth 스키마는 구문 검증을 적용합니다. 자세한 내용은 출력을 참조하세요. 다음은 제한 및 시맨틱에 대한 검증 규칙입니다.

> **Note**
>
> - 20% 무효 규칙은 모든 검증 규칙에 누적되어 적용됩니다. 잘못된 JSON 15%, 잘못된 이미지 15%와 같은 조합으로 인해 가져오기가 20% 제한을 초과하는 경우 가져오기가 실패합니다.
> - 각 데이터 세트 객체는 매니페스트의 한 라인입니다. 비어 있거나 유효하지 않은 라인도 데이터 세트 객체로 간주됩니다.
> - 겹치는 부분은 (테스트와 훈련 간의 공통 레이블)/(훈련 레이블)입니다.

주제

- Limits
- 시맨틱

Limits

검증	제한	발생한 오류
매니페스트 파일 크기	최대 1GB	Error
매니페스트 파일의 최대 라인 수	매니페스트에는 최대 250,000개의 데이터 세트 객체가 한 라인으로 표시됩니다.	Error
레이블당 유효한 데이터 세트 객체 총 수의 하한	>= 1	Error

검증	제한	발생한 오류
레이블의 하한	>=2	Error
레이블의 상한	<= 250	Error
이미지당 최소 경계 상자	0	None
이미지당 최대 경계 상자	50	None

시맨틱

검증	제한	발생한 오류
빈 매니페스트		Error
누락되거나 액세스할 수 없는 source-ref 객체	객체 수 20% 미만	경고
누락되거나 액세스할 수 없는 source-ref 객체	객체 수 > 20%	Error
훈련 데이터 세트에 테스트 레이블이 없습니다.	레이블이 50% 이상 겹쳐야 합니다.	Error
데이터 세트의 동일한 레이블에 대한 레이블과 객체 예제의 혼합 데이터 세트 객체의 동일한 클래스에 대한 분류 및 탐지		오류나 경고 없음
테스트와 트레이닝 간에 중첩되는 자산	테스트 데이터 세트와 훈련 데이터 세트 간에 중첩되는 부분이 없어야 합니다.	
데이터 세트의 이미지는 동일한 버킷에서 가져온 것이어야 합니다.	객체가 다른 버킷에 있는 경우 오류가 발생합니다.	Error

다른 데이터 세트 형식을 매니페스트 파일로 변환

다음 정보를 사용하여 다양한 소스 데이터세트 형식에서 Amazon SageMaker 형식 매니페스트 파일을 생성할 수 있습니다. 매니페스트 파일을 생성한 후 이를 사용하여 데이터 세트에 생성합니다. 자세한 정보는 매니페스트 파일을 참조하세요.

주제

- COCO 데이터 세트 변환
- 다중 레이블 SageMaker Ground Truth 매니페스트 파일 변환
- CSV 파일로 매니페스트 파일 생성

COCO 데이터 세트 변환

COCO는 대규모 객체 감지, 세분화, 캡션 데이터 세트를 지정하는 데 사용되는 형식입니다. 이 Python 예제는 COCO 객체 감지 형식 데이터 세트를 Amazon Rekognition Custom Labels 경계 상자 형식 매니페스트 파일로 변환하는 방법을 보여줍니다. 이 항목에는 직접 코드를 작성하는 데 사용할 수 있는 정보도 포함되어 있습니다.

COCO 형식 JSON 파일은 전체 데이터 세트에 대한 정보를 제공하는 5개 항목으로 구성되어 있습니다. 자세한 정보는 COCO 형식을 참조하세요.

- `info`: 데이터 세트에 대한 일반 정보
- `licenses` : 데이터 세트의 이미지에 대한 라이선스 정보
- `images`: 데이터 세트의 이미지 목록
- `annotations`: 데이터 세트의 모든 이미지에 있는 주석 목록(테두리 상자 포함)
- `categories`: 레이블 카테고리 목록

Amazon Rekognition Custom Labels 매니페스트 파일을 생성하려면 `images`, `annotations` 및 `categories` 목록의 정보가 필요합니다.

Amazon Rekognition Custom Labels 매니페스트 파일은 JSON 라인 형식이며, 각 줄에는 이미지에 있는 하나 이상의 객체에 대한 경계 상자와 레이블 정보가 있습니다. 자세한 정보는 매니페스트 파일의 객체 위치 파악을 참조하세요.

COCO 객체를 사용자 지정 레이블 JSON 라인에 매핑

COCO 형식 데이터 세트를 변환하려면 객체 위치 파악을 위해 COCO 데이터 세트를 Amazon Rekognition Custom Labels 매니페스트 파일에 매핑하세요. 자세한 정보는 매니페스트 파일의 객체 위

치 파악을 참조하세요. 각 이미지에 대한 JSON 라인을 구축하려면 매니페스트 파일에 COCO 데이터 세트 `image`, `annotation`, `category` 객체 필드 ID를 매핑해야 합니다.

다음은 COCO 매니페스트 파일의 예제입니다. 자세한 정보는 COCO 형식을 참조하세요.

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

다음 다이어그램은 데이터 세트의 COCO 데이터세트 목록이 Amazon Rekognition Custom Labels JSON 라인에 매핑되는 이미지를 보여줍니다. 일치하는 색상은 단일 이미지에 대한 정보를 나타냅니다.

단일 JSON 라인에 대한 COCO 객체를 가져오려면

1. 이미지 목록의 각 이미지에 대해 주석 필드 `image_id`의 값이 이미지 `id` 필드와 일치하는 주석 목록에서 주석을 가져옵니다.
2. 1단계에서 일치하는 각 주석에 대해 `categories` 목록을 읽고 `category` 필드 `id` 값이 `annotation` 객체 `category_id` 필드와 일치하는 `category`를 각각 가져옵니다.

3. 일치하는 image, annotation, category 객체를 사용하여 이미지의 JSON 라인을 생성합니다. 필드를 매핑하려면 COCO 객체 필드를 사용자 지정 레이블 JSON 라인 객체 필드에 매핑하기 항목을 참조하세요.
4. images 목록의 각 image 객체에 대해 JSON 라인을 생성할 때까지 1~3단계를 반복합니다.

예제 코드는 COCO 데이터 세트 변환 항목을 참조하세요.

COCO 객체 필드를 사용자 지정 레이블 JSON 라인 객체 필드에 매핑하기

Amazon Rekognition Custom Labels JSON 라인의 COCO 객체를 식별한 후에는 COCO 객체 필드를 상응하는 Amazon Rekognition Custom Labels JSON 라인 객체 필드에 매핑해야 합니다. 다음 예제 Amazon Rekognition Custom Labels JSON 라인은 하나의 이미지(id=000000245915)를 위의 COCO JSON 예제에 매핑합니다. 다음 정보를 참고하세요.

- source-ref는 Amazon S3 버킷의 이미지 위치입니다. Amazon S3 버킷에 COCO 이미지가 저장되지 않은 경우, 이미지를 Amazon S3 버킷으로 이동해야 합니다.
- annotations 목록에는 이미지의 각 객체에 대해 annotation 객체가 포함되어 있습니다. annotation 객체에는 경계 상자 정보(top, left, width, height) 및 레이블 식별자(class_id)가 포함됩니다.
- 레이블 식별자(class_id)는 메타데이터의 class-map 목록에 매핑됩니다. 그것은 이미지에 사용된 레이블을 나열합니다.

```
{
 "source-ref": "s3://custom-labels-bucket/images/000000245915.jpg",
 "bounding-box": {
  "image_size": {
   "width": 640,
   "height": 480,
   "depth": 3
  },
  "annotations": [{
   "class_id": 0,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 1,
   "top": 65,
```

```
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
}
```

다음 정보를 사용하여 Amazon Rekognition Custom Labels 매니페스트 파일 필드를 COCO 데이터 세트 JSON 필드에 매핑할 수 있습니다.

source-ref

이미지 위치의 S3 형식 URL입니다. 이미지는 S3 버킷에 저장되어야 합니다. 자세한 정보는 source-ref을 참조하세요. `coco_url` COCO 필드가 S3 버킷 위치를 가리키는 경우 `coco_url`의 값을 `source-ref`의 값으로 사용할 수 있습니다. 또는 `file_name`(COCO) 필드에 `source-ref`를 매핑하고, 변환 코드에서 이미지가 저장되는 위치에 필요한 S3 경로를 추가할 수 있습니다.

bounding-box

사용자가 선택한 레이블 속성 이름 자세한 정보는 *bounding-box*을 참조하세요.

image_size

이미지 크기(픽셀 단위) 이미지 목록의 `image` 객체에 매핑됩니다.

- `height` -> `image.height`
- `width` -> `image.width`
- `depth` -> Amazon Rekognition Custom Labels에는 사용되지 않지만 값을 입력해야 합니다.

주석

annotation 객체의 목록. 이미지의 각 객체마다 annotation이 하나씩 있습니다.

주석

이미지에 있는 객체의 한 인스턴스에 대한 경계 상자 정보가 들어 있습니다.

- class_id -> 사용자 지정 레이블의 class-map 목록에 매핑되는 숫자 ID
- top -> bbox[1]
- left -> bbox[0]
- width -> bbox[2]
- height -> bbox[3]

***bounding-box*-metadata**

레이블 속성의 메타데이터 레이블 및 레이블 식별자를 포함합니다. 자세한 정보는 *bounding-box*-metadata을 참조하세요.

Objects

이미지에 있는 객체의 배열입니다. 인덱스를 기준으로 annotations 목록에 매핑됩니다.

객체

- confidence->Amazon Rekognition Custom Labels에는 사용되지 않지만 값(1)이 필요합니다.

class-map

이미지에서 감지된 객체에 적용되는 레이블(클래스)의 맵입니다. 카테고리 목록에 있는 카테고리 개체에 매핑됩니다.

- id -> category.id
- id value -> category.name

type

groundtruth/object-detection이어야 합니다.

human-annotated

yes 또는 no을 지정합니다. 자세한 정보는 *bounding-box*-metadata을 참조하세요.

creation-date -> image.date_captured

이미지가 생성된 날짜 및 시간입니다. COCO 이미지 목록에 있는 이미지의 image.date_captured 필드에 매핑됩니다. Amazon Rekognition Custom Labels의 creation-date의 형식은 Y-M-DTH:MS:S일 것으로 예상합니다.

job-name

사용자가 선택한 직무 이름

COCO 형식

COCO 데이터 세트는 전체 데이터 세트에 대한 정보를 제공하는 다섯 개의 항목으로 구성됩니다. COCO 객체 감지 데이터 세트의 형식은 COCO 데이터 형식에 문서화되어 있습니다.

- 정보: 데이터 세트에 대한 일반 정보입니다.
- 라이선스: 데이터 세트의 이미지에 대한 라이선스 정보입니다.
- 이미지: 데이터 세트에 있는 이미지 목록
- 주석: 데이터 세트의 모든 이미지에 있는 주석(경계 상자 포함)의 목록
- 카테고리: 레이블 카테고리 목록

사용자 지정 레이블 매니페스트를 만들려면 COCO 매니페스트 파일의 images, annotations, categories 목록을 사용하세요. 다른 항목(info, licences)은 필수가 아닙니다. 다음은 COCO 매니페스트 파일의 예제입니다.

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
```

```
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

이미지 목록

COCO 데이터 세트에서 참조하는 이미지는 이미지 배열에 나열됩니다. 각 이미지 객체에는 이미지 파일 이름과 같은 이미지에 대한 정보가 들어 있습니다. 다음 예제 이미지 객체에서 다음 정보와 Amazon Rekognition Custom Labels 매니페스트 파일을 생성하는 데 필요한 필드를 기록해 둡니다.

- `id`: (필수) 이미지의 고유 식별자 `id` 필드는 주석 배열(경계 상자 정보가 저장되는 위치)의 `id` 필드에 매핑됩니다.
- `license`: (필수 아님) 라이선스 어레이에 매핑됩니다.
- `coco_url`: (선택 사항) 이미지의 위치
- `flickr_url`: (필수 아님) Flickr에서의 이미지 위치
- `width`: (필수) 이미지의 너비
- `height`: (필수) 이미지의 높이
- `file_name`: (필수) 이미지 파일 이름 이 예제에서 `file_name`과 `id`는 일치하지만 COCO 데이터 세트의 요구 사항은 아닙니다.
- `date_captured`: (필수) 이미지를 캡처한 날짜 및 시간

```
{
    "id": 245915,
    "license": 4,
    "coco_url": "http://images.cocodataset.org/val2017/nnnnnnnnnnnn.jpg",
    "flickr_url": "http://farm1.staticflickr.com/88/nnnnnnnnnnnnnnnnnn.jpg",
    "width": 640,
    "height": 480,
    "file_name": "000000245915.jpg",
    "date_captured": "2013-11-18 02:53:27"
}
```

주석(경계 상자) 목록

모든 이미지에 있는 모든 객체의 경계 상자 정보는 주석 목록에 저장됩니다. 단일 주석 개체에는 단일 개체에 대한 경계 상자 정보와 이미지의 개체 레이블이 포함됩니다. 이미지에 있는 객체의 각 인스턴스에는 주석 개체가 있습니다.

다음 예제에서 다음 정보와 Amazon Rekognition Custom Labels 매니페스트 파일을 생성하는 데 필요한 필드를 기록해 둡니다.

- `id`: (필수 아님) 주석의 식별자
- `image_id`: (필수) 이미지 배열의 `id` 이미지에 대응합니다.
- `category_id`: (필수) 경계 상자 내의 객체를 식별하는 레이블의 식별자입니다. 카테고리 배열의 `id` 필드에 매핑됩니다.
- `iscrowd`: (필수 아님) 이미지에 많은 객체가 포함되어 있는지 여부를 지정합니다.
- `segmentation`: (필수 아님) 이미지 상의 객체에 대한 세그멘트화 정보입니다. Amazon Rekognition Custom Labels는 세그멘트화를 지원하지 않습니다.
- `area`: (필수 아님) 주석의 영역
- `bbox`: (필수) 이미지에 있는 객체 주위의 경계 상자 좌표(픽셀 단위)를 포함합니다.

```
{
    "id": 1409619,
    "category_id": 1,
    "iscrowd": 0,
    "segmentation": [
        [86.0, 238.8,.........382.74, 241.17]
    ],
    "image_id": 245915,
```

```
    "area": 3556.2197000000015,
    "bbox": [86, 65, 220, 334]
}
```

카테고리 목록

레이블 정보는 카테고리 배열에 저장됩니다. 다음 예제 카테고리 객체에서 다음 정보와 Amazon Rekognition Custom Labels 매니페스트 파일을 생성하는 데 필요한 필드를 기록해 둡니다.

- `supercategory`: (필수 아님) 레이블의 상위 카테고리
- `id`: (필수) 레이블 식별자 `id` 필드는 `annotation` 객체의 `category_id` 필드에 매핑됩니다. 다음 예제에서 에코 도트의 식별자는 2입니다.
- `name`: (필수) 레이블 이름

```
{"supercategory": "speaker","id": 2,"name": "echo dot"}
```

COCO 데이터 세트 변환

다음 Python 예제를 사용하여 COCO 형식 데이터 세트의 경계 상자 정보를 Amazon Rekognition Custom Labels 매니페스트 파일로 변환합니다. 해당 코드는 생성된 매니페스트 파일을 Amazon S3 버킷에 업로드합니다. 해당 코드는 이미지를 업로드하는 데 사용할 수 있는 AWS CLI 명령도 제공합니다.

COCO 데이터 세트를 변환하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하세요.
   a. `AmazonS3FullAccess` 권한이 있는지 확인합니다. 자세한 정보는 SDK 권한 설정을 참조하세요.
   b. 및 SDK를 설치 AWS CLI 및 구성합니다. AWS 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 다음 Python 코드를 사용하여 COCO 데이터 세트를 변환합니다. 다음 값을 설정하세요.
   - `s3_bucket`: 이미지 및 Amazon Rekognition Custom Labels 매니페스트 파일을 저장할 S3 버킷의 이름입니다.
   - `s3_key_path_images`: S3 버킷(`s3_bucket`) 내에서 이미지를 배치하려는 위치의 경로
   - `s3_key_path_manifest_file`: S3 버킷(`s3_bucket`) 내에서 사용자 지정 레이블 매니페스트 파일을 배치할 경로

- local_path: 예제에서 입력 COCO 데이터 세트를 열고 새 사용자 지정 레이블 매니페스트 파일도 저장하는 로컬 경로
- local_images_path: 훈련에 사용할 이미지의 로컬 경로
- coco_manifest: 입력 COCO 데이터 세트 파일 이름
- cl_manifest_file: 예제에서 만든 매니페스트 파일의 이름 local_path에서 지정한 위치에 파일이 저장됩니다. 일반적으로 파일에는 .manifest 확장자가 있지만 필수는 아닙니다.
- job_name: 사용자 정의 레이블 작업의 이름

```
import json
import os
import random
import shutil
import datetime
import botocore
import boto3
import PIL.Image as Image
import io

#S3 location for images
s3_bucket = 'bucket'
s3_key_path_manifest_file = 'path to custom labels manifest file/'
s3_key_path_images = 'path to images/'
s3_path='s3://' + s3_bucket  + '/' + s3_key_path_images
s3 = boto3.resource('s3')

#Local file information
local_path='path to input COCO dataset and output Custom Labels manifest/'
local_images_path='path to COCO images/'
coco_manifest = 'COCO dataset JSON file name'
coco_json_file = local_path + coco_manifest
job_name='Custom Labels job name'
cl_manifest_file = 'custom_labels.manifest'

label_attribute ='bounding-box'

open(local_path + cl_manifest_file, 'w').close()

# class representing a Custom Label JSON line for an image
class cl_json_line:
    def __init__(self,job, img):
```

```
        #Get image info. Annotations are dealt with seperately
        sizes=[]
        image_size={}
        image_size["width"] = img["width"]
        image_size["depth"] = 3
        image_size["height"] = img["height"]
        sizes.append(image_size)

        bounding_box={}
        bounding_box["annotations"] = []
        bounding_box["image_size"] = sizes

        self.__dict__["source-ref"] = s3_path + img['file_name']
        self.__dict__[job] = bounding_box

        #get metadata
        metadata = {}
        metadata['job-name'] = job_name
        metadata['class-map'] = {}
        metadata['human-annotated']='yes'
        metadata['objects'] = []
        date_time_obj = datetime.datetime.strptime(img['date_captured'], '%Y-%m-%d
 %H:%M:%S')
        metadata['creation-date']= date_time_obj.strftime('%Y-%m-%dT%H:%M:%S')
        metadata['type']='groundtruth/object-detection'

        self.__dict__[job + '-metadata'] = metadata


print("Getting image, annotations, and categories from COCO file...")

with open(coco_json_file) as f:

    #Get custom label compatible info
    js = json.load(f)
    images = js['images']
    categories = js['categories']
    annotations = js['annotations']

    print('Images: ' + str(len(images)))
    print('annotations: ' + str(len(annotations)))
    print('categories: ' + str(len (categories)))
```

```
print("Creating CL JSON lines...")

images_dict = {image['id']: cl_json_line(label_attribute, image) for image in
 images}

print('Parsing annotations...')
for annotation in annotations:

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}

    # get bounding box information
    cl_bounding_box={}
    cl_bounding_box['left'] = annotation['bbox'][0]
    cl_bounding_box['top'] = annotation['bbox'][1]

    cl_bounding_box['width'] = annotation['bbox'][2]
    cl_bounding_box['height'] = annotation['bbox'][3]
    cl_bounding_box['class_id'] = annotation['category_id']

    getattr(image, label_attribute)['annotations'].append(cl_bounding_box)


    for category in categories:
         if annotation['category_id'] == category['id']:
            getattr(image, label_attribute + '-metadata')['class-map']
[category['id']]=category['name']


    cl_object={}
    cl_object['confidence'] = int(1)  #not currently used by Custom Labels
    getattr(image, label_attribute + '-metadata')['objects'].append(cl_object)

print('Done parsing annotations')

# Create manifest file.
print('Writing Custom Labels manifest...')

for im in images_dict.values():

    with open(local_path+cl_manifest_file, 'a+') as outfile:
```

```
            json.dump(im.__dict__,outfile)
            outfile.write('\n')
            outfile.close()

# Upload manifest file to S3 bucket.
print ('Uploading Custom Labels manifest file to S3 bucket')
print('Uploading'  + local_path + cl_manifest_file + ' to ' +
 s3_key_path_manifest_file)
print(s3_bucket)
s3 = boto3.resource('s3')
s3.Bucket(s3_bucket).upload_file(local_path + cl_manifest_file,
 s3_key_path_manifest_file + cl_manifest_file)

# Print S3 URL to manifest file,
print ('S3 URL Path to manifest file. ')
print('\033[1m s3://' + s3_bucket + '/' + s3_key_path_manifest_file +
 cl_manifest_file + '\033[0m')

# Display aws s3 sync command.
print ('\nAWS CLI s3 sync command to upload your images to S3 bucket. ')
print ('\033[1m aws s3 sync ' + local_images_path + ' ' + s3_path + '\033[0m')
```

3. 코드를 실행합니다.
4. 프로그램 출력에서 `s3 sync` 명령을 기록해 둡니다. 이 정보는 다음 단계에서 필요합니다.
5. 명령 프롬프트에서 `s3 sync` 명령을 실행합니다. 이미지는 S3 버킷에 업로드됩니다. 업로드 중에 명령이 실패하면 로컬 이미지가 S3 버킷과 동기화될 때까지 명령을 다시 실행하세요.
6. 프로그램 출력에서 매니페스트 파일의 S3 URL 경로를 기록해 둡니다. 이 정보는 다음 단계에서 필요합니다.
7. SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)의 지침에 따라 업로드된 매니페스트 파일로 데이터 세트를 생성하세요. 8단계로 .manifest 파일 위치에 이전 단계에서 기록해 둔 Amazon S3 URL을 입력합니다. AWS SDK를 사용하고 있다면 그렇게 하세요. SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기

다중 레이블 SageMaker Ground Truth 매니페스트 파일 변환

이 주제에서는 다중 레이블 Amazon SageMaker Ground Truth 매니페스트 파일을 Amazon Rekognition 사용자 지정 레이블 형식 매니페스트 파일로 변환하는 방법을 보여줍니다.

SageMaker 다중 레이블 작업에 대한 Ground Truth 매니페스트 파일은 Amazon Rekognition 사용자 지정 레이블 형식 매니페스트 파일과 형식이 다릅니다. 다중 레이블 분류란 어떤 이미지가 일련의 클래스로 분류되지만 동시에 여러 클래스에 속할 수 있는 경우를 말합니다. 이 경우 이미지에 축구공, 공과 같은 여러 레이블(다중 레이블)이 있을 수 있습니다.

다중 레이블 SageMaker Ground Truth 작업에 대한 자세한 내용은 이미지 분류 (다중 레이블) 를 참조하십시오. 다중 레이블 형식 Amazon Rekognition Custom Labels 매니페스트 파일에 대한 자세한 내용은 the section called “이미지에 여러 이미지 수준 레이블 추가” 항목을 참조하세요.

SageMaker Ground Truth 작업을 위한 매니페스트 파일 가져오기

다음 절차는 Amazon SageMaker Ground Truth 작업의 출력 매니페스트 파일 (`output.manifest`) 을 가져오는 방법을 보여줍니다. `output.manifest`를 다음 절차의 입력으로 사용합니다.

SageMaker Ground Truth 작업 매니페스트 파일을 다운로드하려면

1. https://console.aws.amazon.com/sagemaker/ 링크를 엽니다.
2. 탐색 창에서 Ground Truth를 선택한 다음 레이블 지정 작업을 선택합니다.
3. 사용할 매니페스트 파일이 들어 있는 레이블 지정 작업을 선택합니다.
4. 세부 정보 페이지의 출력 데이터 세트 위치 아래에 있는 링크를 선택합니다. Amazon S3 콘솔이 데이터 세트 위치에서 열립니다.
5. `Manifests`, `output`, 다음 `output.manifest`을 선택합니다.
6. 매니페스트 파일을 다운로드하려면 객체 작업을 선택하고 다운로드를 선택합니다.

다중 SageMaker 레이블 매니페스트 파일 변환

다음 절차는 기존 다중 레이블 형식 매니페스트 파일에서 다중 레이블 형식의 Amazon Rekognition 사용자 지정 레이블 매니페스트 파일을 생성합니다. SageMaker GroundTruth

> **Note**
>
> 코드를 실행하려면 Python 버전 3 이상이 필요합니다.

다중 레이블 매니페스트 파일을 변환하려면 SageMaker

1. 다음 Python 코드를 실행합니다. SageMaker Ground Truth 작업을 위한 매니페스트 파일 가져오기에서 생성한 매니페스트 파일의 이름을 명령줄 인수로 제공합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0
"""
Purpose
Shows how to create and Amazon Rekognition Custom Labels format
manifest file from an Amazon SageMaker Ground Truth Image
Classification (Multi-label) format manifest file.
"""
import json
import logging
import argparse
import os.path

logger = logging.getLogger(__name__)

def create_manifest_file(ground_truth_manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels format manifest file from
    an Amazon SageMaker Ground Truth Image Classification (Multi-label) format
    manifest file.
    :param: ground_truth_manifest_file: The name of the Ground Truth manifest file,
    including the relative path.
    :return: The name of the new Custom Labels manifest file.
    """

    logger.info('Creating manifest file from %s', ground_truth_manifest_file)
    new_manifest_file =
 f'custom_labels_{os.path.basename(ground_truth_manifest_file)}'

    # Read the SageMaker Ground Truth manifest file into memory.
    with open(ground_truth_manifest_file) as gt_file:
        lines = gt_file.readlines()

    #Iterate through the lines one at a time to generate the
    #new lines for the Custom Labels manifest file.
    with open(new_manifest_file, 'w') as the_new_file:
        for line in lines:
            #job_name - The of the Amazon Sagemaker Ground Truth job.
            job_name = ''
            # Load in the old json item from the Ground Truth manifest file
            old_json = json.loads(line)

            # Get the job name
```

```
            keys = old_json.keys()
            for key in keys:
                if 'source-ref' not in key and '-metadata' not in key:
                    job_name = key

            new_json = {}
            # Set the location of the image
            new_json['source-ref'] = old_json['source-ref']

            # Temporarily store the list of labels
            labels = old_json[job_name]

            # Iterate through the labels and reformat to Custom Labels format
            for index, label in enumerate(labels):
                new_json[f'{job_name}{index}'] = index
                metadata = {}
                metadata['class-name'] = old_json[f'{job_name}-metadata']['class-
map'][str(label)]
                metadata['confidence'] = old_json[f'{job_name}-metadata']
['confidence-map'][str(label)]
                metadata['type'] = 'groundtruth/image-classification'
                metadata['job-name'] = old_json[f'{job_name}-metadata']['job-name']
                metadata['human-annotated'] = old_json[f'{job_name}-metadata']
['human-annotated']
                metadata['creation-date'] = old_json[f'{job_name}-metadata']
['creation-date']
                # Add the metadata to new json line
                new_json[f'{job_name}{index}-metadata'] = metadata
            # Write the current line to the json file
            the_new_file.write(json.dumps(new_json))
            the_new_file.write('\n')

    logger.info('Created %s', new_manifest_file)
    return  new_manifest_file

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "manifest_file", help="The Amazon SageMaker Ground Truth manifest file"
        "that you want to use."
```

```
    )


def main():
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
    try:
        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()
        # Create the manifest file
        manifest_file = create_manifest_file(args.manifest_file)
        print(f'Manifest file created: {manifest_file}')
    except FileNotFoundError as err:
        logger.exception('File not found: %s', err)
        print(f'File not found: {err}. Check your manifest file.')

if __name__ == "__main__":
    main()
```

2. 스크립트에 표시되는 새 매니페스트 파일의 이름을 기록해 둡니다. 다음 단계에서 해당 항목을 사용합니다.
3. 매니페스트 파일을 저장하는 데 사용할 Amazon S3 버킷에 매니페스트 파일을 업로드합니다.

> **Note**
>
> Amazon Rekognition Custom Labels가 매니페스트 파일 JSON 라인의 `source-ref` 필드에서 참조되는 Amazon S3 버킷에 액세스할 수 있는지 확인하세요. 자세한 정보는 외부 Amazon S3 버킷에 액세스을 참조하세요. Ground Truth 작업이 Amazon Rekognition Custom Labels 콘솔 버킷에 이미지를 저장하는 경우 권한을 추가할 필요가 없습니다.

4. SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)의 지침에 따라 업로드된 매니페스트 파일로 데이터 세트를 생성하세요. 8단계로 .manifest 파일 위치에 매니페스트 파일의 위치로 사용할 Amazon S3 URL을 입력합니다. AWS SDK를 사용하고 있다면 그렇게 하세요. SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기

CSV 파일로 매니페스트 파일 생성

이 예제 Python 스크립트는 Comma Separated Values(CSV) 파일을 사용하여 이미지에 레이블을 지정함으로써 매니페스트 파일 생성을 간소화합니다. 사용자가 CSV 파일을 생성합니다. 매니페스트 파

일은 다중 레이블 이미지 분류 또는 다중 레이블 이미지 분류 용도에 적합합니다. 자세한 정보는 객체, 장면 및 개념 찾기을 참조하세요.

> **Note**
>
> 이 스크립트는 객체 위치 또는 브랜드 위치를 찾는 데 적합한 매니페스트 파일을 생성하지 않습니다.

매니페스트 파일은 모델 학습에 사용되는 이미지를 설명합니다. 이미지 위치와 이미지에 지정된 레이블을 예로 들 수 있습니다. 매니페스트 파일은 하나 이상의 JSON 라인으로 구성됩니다. 각 JSON 라인은 단일 이미지를 설명합니다. 자세한 정보는 the section called “매니페스트 파일의 이미지 수준 레이블”을 참조하세요.

CSV 파일은 텍스트 파일의 여러 행에 대한 표 형식 데이터를 나타냅니다. 행의 필드는 쉼표로 구분합니다. 자세한 내용은 comma separated values를 참조하세요. 이 스크립트에서 CSV 파일의 각 행은 단일 이미지를 나타내며 매니페스트 파일의 JSON 라인에 매핑됩니다. 다중 레이블 이미지 분류를 지원하는 매니페스트 파일의 CSV 파일을 만들려면 각 행에 하나 이상의 이미지 수준 레이블을 추가하세요. 이미지 분류에 적합한 매니페스트 파일을 만들려면 각 행에 단일 이미지 수준 레이블을 추가하세요.

예를 들어, 다음 CSV 파일은 다중 레이블 이미지 분류(꽃) 시작하기 프로젝트의 이미지를 설명합니다.

```
camellia1.jpg,camellia,with_leaves
camellia2.jpg,camellia,with_leaves
camellia3.jpg,camellia,without_leaves
helleborus1.jpg,helleborus,without_leaves,not_fully_grown
helleborus2.jpg,helleborus,with_leaves,fully_grown
helleborus3.jpg,helleborus,with_leaves,fully_grown
jonquil1.jpg,jonquil,with_leaves
jonquil2.jpg,jonquil,with_leaves
jonquil3.jpg,jonquil,with_leaves
jonquil4.jpg,jonquil,without_leaves
mauve_honey_myrtle1.jpg,mauve_honey_myrtle,without_leaves
mauve_honey_myrtle2.jpg,mauve_honey_myrtle,with_leaves
mauve_honey_myrtle3.jpg,mauve_honey_myrtle,with_leaves
mediterranean_spurge1.jpg,mediterranean_spurge,with_leaves
mediterranean_spurge2.jpg,mediterranean_spurge,without_leaves
```

스크립트는 각 행에 대해 JSON 라인을 생성합니다. 예를 들어, 다음은 첫 번째 행(`camellia1.jpg,camellia,with_leaves`)의 JSON 라인입니다.

```
{"source-ref": "s3://bucket/flowers/train/camellia1.jpg","camellia": 1,"camellia-
metadata":{"confidence": 1,"job-name": "labeling-job/camellia","class-name":
 "camellia","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"},"with_leaves": 1,"with_leaves-metadata":
{"confidence": 1,"job-name": "labeling-job/with_leaves","class-name":
 "with_leaves","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"}}
```

예제 CSV에는 이미지에 대한 Amazon S3 경로가 없습니다. CSV 파일에 이미지의 Amazon S3 경로가 포함되어 있지 않은 경우 `--s3_path` 명령줄 인수를 사용하여 이미지에 대한 Amazon S3 경로를 지정하세요.

스크립트는 각 이미지의 첫 번째 항목을 중복 제거된 이미지 CSV 파일에 기록합니다. 중복 제거된 이미지 CSV 파일에는 입력 CSV 파일에 있는 각 이미지의 단일 인스턴스가 포함됩니다. 입력 CSV 파일에서 이미지가 추가로 나타나는 경우 중복 이미지 CSV 파일에 기록됩니다. 스크립트가 중복된 이미지를 발견하면 중복 이미지 CSV 파일을 검토하고 필요에 따라 중복 제거된 이미지 CSV 파일을 업데이트하세요. 중복 제거된 파일을 사용하여 스크립트를 다시 실행합니다. 입력 CSV 파일에 중복이 없는 경우 스크립트는 중복 제거된 이미지 CSV 파일과 중복 이미지 CSV 파일이 비어 있으므로 해당 파일을 삭제합니다.

이 절차에서 사용자는 CSV 파일을 만들고 Python 스크립트를 실행하여 매니페스트 파일을 만듭니다.

CSV 파일에서 매니페스트 파일을 생성하려면

1. 각 행에 다음 필드를 포함하는 CSV 파일을 생성합니다 (이미지당 한 행). CSV 파일에 헤더 행을 추가하지 마세요.

필드 1	필드 2	필드 n
이미지 이름 또는 Amazon S3 이미지 경로 예: `s3://my-bucket/flowers/train/camellia1.jpg` Amazon S3 경로가 있는 이미지와 그렇지 않은 이미지를 혼합할 수는 없습니다.	이미지의 첫 번째 이미지 수준 레이블	쉼표로 구분된 하나 이상의 추가적인 이미지 수준 레이블 다중 레이블 이미지 분류를 지원하는 매니페스트 파일을 생성하려는 경우에만 추가하세요.

예: `camellia1.jpg,camellia,with_leaves` 또는 `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`

2. CSV 파일을 저장합니다.
3. 다음 Python 스크립트를 실행합니다. 다음 인수를 제공하세요.
   - `csv_file`: 1단계에서 생성한 CSV 파일
   - `manifest_file`: 생성할 매니페스트 파일의 이름
   - (선택 사항)`--s3_path` *`s3://path_to_folder/`*: 이미지 파일 이름에 추가할 Amazon S3 경로(필드 1) 필드 1의 이미지에 아직 S3 경로가 포함되어 있지 않은 경우 `--s3_path`를 사용합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0

from datetime import datetime, timezone
import argparse
import logging
import csv
import os
import json

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service documentation.
Shows how to create an image-level (classification) manifest file from a CSV file.
You can specify multiple image level labels per image.
CSV file format is
image,label,label,..
If necessary, use the bucket argument to specify the S3 bucket folder for the
 images.
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-gt-cl-
transform.html
"""

logger = logging.getLogger(__name__)


def check_duplicates(csv_file, deduplicated_file, duplicates_file):
```

```
    """
    Checks for duplicate images in a CSV file. If duplicate images
    are found, deduplicated_file is the deduplicated CSV file - only the first
    occurence of a duplicate is recorded. Other duplicates are recorded in
duplicates_file.
    :param csv_file: The source CSV file.
    :param deduplicated_file: The deduplicated CSV file to create. If no duplicates
are found
    this file is removed.
    :param duplicates_file: The duplicate images CSV file to create. If no
duplicates are found
    this file is removed.
    :return: True if duplicates are found, otherwise false.
    """

    logger.info("Deduplicating %s", csv_file)

    duplicates_found = False

    # Find duplicates.
    with open(csv_file, 'r', newline='', encoding="UTF-8") as f,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(duplicates_file, 'w', encoding="UTF-8") as duplicates:

        reader = csv.reader(f, delimiter=',')
        dedup_writer = csv.writer(dedup)
        duplicates_writer = csv.writer(duplicates)

        entries = set()
        for row in reader:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                duplicates_writer.writerow(row)
                duplicates_found = True

    if duplicates_found:
        logger.info("Duplicates found check %s", duplicates_file)
```

```
    else:
        os.remove(duplicates_file)
        os.remove(deduplicated_file)

    return duplicates_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Reads a CSV file and creates a Custom Labels classification manifest file.
    :param csv_file: The source CSV file.
    :param manifest_file: The name of the manifest file to create.
    :param s3_path: The S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s", csv_file)

    image_count = 0
    label_count = 0

    with open(csv_file, newline='', encoding="UTF-8") as csvfile,\
            open(manifest_file, "w", encoding="UTF-8") as output_file:

        image_classifications = csv.reader(
            csvfile, delimiter=',', quotechar='|')

        # Process each row (image) in CSV file.
        for row in image_classifications:
            source_ref = str(s3_path)+row[0]

            image_count += 1

            # Create JSON for image source ref.
            json_line = {}
            json_line['source-ref'] = source_ref

            # Process each image level label.
            for index in range(1, len(row)):
                image_level_label = row[index]

                # Skip empty columns.
                if image_level_label == '':
                    continue
                label_count += 1
```

```
                # Create the JSON line metadata.
                 json_line[image_level_label] = 1
                 metadata = {}
                 metadata['confidence'] = 1
                 metadata['job-name'] = 'labeling-job/' + image_level_label
                 metadata['class-name'] = image_level_label
                 metadata['human-annotated'] = "yes"
                 metadata['creation-date'] = \
                     datetime.now(timezone.utc).strftime('%Y-%m-%dT%H:%M:%S.%f')
                 metadata['type'] = "groundtruth/image-classification"

                 json_line[f'{image_level_label}-metadata'] = metadata

                 # Write the image JSON Line.
            output_file.write(json.dumps(json_line))
            output_file.write('\n')

    output_file.close()
    logger.info("Finished creating manifest file %s\nImages: %s\nLabels: %s",
                manifest_file, image_count, label_count)

    return image_count, label_count


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "csv_file", help="The CSV file that you want to process."
    )

    parser.add_argument(
        "--s3_path", help="The S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the S3 path.",
 required=False
    )


def main():
```

```
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        s3_path = args.s3_path
        if s3_path is None:
            s3_path = ''

        # Create file names.
        csv_file = args.csv_file
        file_name = os.path.splitext(csv_file)[0]
        manifest_file = f'{file_name}.manifest'
        duplicates_file = f'{file_name}-duplicates.csv'
        deduplicated_file = f'{file_name}-deduplicated.csv'

        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
                  f"and then update {deduplicated_file}. ")
            print(f"{deduplicated_file} contains the first occurence of a
duplicate. "
                  "Update as necessary with the correct label information.")
            print(f"Re-run the script with {deduplicated_file}")
        else:
            print("No duplicates found. Creating manifest file.")

            image_count, label_count = create_manifest_file(csv_file,
                                                            manifest_file,
                                                            s3_path)

            print(f"Finished creating manifest file: {manifest_file} \n"
                  f"Images: {image_count}\nLabels: {label_count}")

    except FileNotFoundError as err:
        logger.exception("File not found: %s", err)
        print(f"File not found: {err}. Check your input CSV file.")
```

```
if __name__ == "__main__":
    main()
```

4. 테스트 데이터 세트를 사용하려는 경우 1~3단계를 반복하여 테스트 데이터 세트의 매니페스트 파일을 생성하세요.
5. 필요한 경우 CSV 파일의 열 1에서 지정한(또는 `--s3_path` 명령줄에서 지정한) Amazon S3 버킷 경로에 이미지를 복사합니다. 다음 AWS S3 명령을 사용할 수 있습니다.

```
aws s3 cp --recursive your-local-folder s3://your-target-S3-location
```

6. 매니페스트 파일을 저장하는 데 사용할 Amazon S3 버킷에 매니페스트 파일을 업로드합니다.

> **Note**
>
> Amazon Rekognition Custom Labels가 매니페스트 파일 JSON 라인의 `source-ref` 필드에서 참조되는 Amazon S3 버킷에 액세스할 수 있는지 확인하세요. 자세한 정보는 외부 Amazon S3 버킷에 액세스을 참조하세요. Ground Truth 작업이 Amazon Rekognition Custom Labels 콘솔 버킷에 이미지를 저장하는 경우 권한을 추가할 필요가 없습니다.

7. SageMaker Ground Truth 매니페스트 파일을 사용하여 데이터세트 만들기 (콘솔)의 지침에 따라 업로드된 매니페스트 파일로 데이터 세트를 생성하세요. 8단계로 .manifest 파일 위치에 매니페스트 파일의 위치로 사용할 Amazon S3 URL을 입력합니다. AWS SDK를 사용하고 있다면 그렇게 하세요 SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기.

기존 데이터 세트

이전에 데이터 세트를 만든 경우 해당 콘텐츠를 새 데이터 세트에 복사할 수 있습니다. AWS SDK로 기존 데이터세트에서 데이터세트를 만들려면 을 참조하십시오. 기존 데이터 세트를 사용하여 데이터 세트 생성(SDK)

기존 Amazon Rekognition Custom Labels 데이터 세트를 사용하여 데이터 세트를 생성하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.

5. 프로젝트 페이지에서 데이터 세트에 추가하려는 프로젝트를 선택합니다. 프로젝트 세부 정보 페이지가 열립니다.
6. 데이터 세트 생성을 선택합니다. 데이터 세트 생성 페이지가 표시됩니다.
7. 시작 구성에서 단일 데이터 세트로 시작 또는 훈련 데이터 세트로 시작을 선택합니다. 더 높은 품질의 모델을 만들려면 별도의 훈련 및 테스트 데이터 세트로 시작하는 것이 좋습니다.

   Single dataset

   a. 훈련 데이터 세트 세부 정보 섹션에서 기존 Amazon Rekognition Custom Labels 데이터 세트 복사를 선택합니다.
   b. 훈련 데이터 세트 세부 정보 섹션의 데이터 세트 편집 상자에서 복사하려는 데이터 세트의 이름을 입력하거나 선택합니다.
   c. 데이터 세트 생성을 선택합니다. 프로젝트의 데이터 세트 페이지가 열립니다.

   Separate training and test datasets

   a. 훈련 데이터 세트 세부 정보 항목에서 기존 Amazon Rekognition Custom Labels 데이터 세트 복사를 선택합니다.
   b. 훈련 데이터 세트 세부 정보 항목의 데이터 세트 편집 상자에서 복사하려는 데이터 세트의 이름을 입력하거나 선택합니다.
   c. 테스트 데이터 세트 세부 정보 항목에서 기존 Amazon Rekognition Custom Labels 데이터 세트 복사를 선택합니다.
   d. 테스트 데이터 세트 세부 정보 항목의 데이터 세트 편집 상자에서 복사하려는 데이터 세트의 이름을 입력하거나 선택합니다.

   > **Note**
   >
   > 훈련 데이터 세트와 테스트 데이터 세트에는 서로 다른 이미지 소스가 있을 수 있습니다.

   e. 데이터 세트 생성을 선택합니다. 프로젝트의 데이터 세트 페이지가 열립니다.
8. 레이블을 추가하거나 변경해야 하면 이미지 레이블 지정 항목을 수행합니다.
9. 모델 훈련(콘솔)에 나온 단계에 따라 모델을 훈련하세요.

이미지 레이블 지정

레이블은 이미지에서 객체, 장면, 개념 또는 객체 주위의 경계 상자를 식별합니다. 예를 들어 데이터 세트에 개 이미지가 포함된 경우 개 품종에 대한 레이블을 추가할 수 있습니다.

이미지를 데이터 세트로 가져온 후 이미지에 레이블을 추가하거나 레이블이 잘못된 이미지를 수정해야 할 수 있습니다. 예를 들어 로컬 컴퓨터에서 가져온 이미지는 레이블이 지정되지 않습니다. 데이터 세트 갤러리를 사용하여 데이터 세트에 새 레이블을 추가하고 데이터 세트의 이미지에 레이블과 경계 상자를 지정합니다.

데이터 세트의 이미지에 레이블을 지정하는 방식에 따라 Amazon Rekognition Custom Labels가 훈련하는 모델 유형이 결정됩니다. 자세한 정보는 데이터 세트 목적 설정을 참조하세요.

주제

- 레이블 관리
- 이미지에 이미지 수준 레이블 지정
- 경계 상자로 객체에 레이블 지정

레이블 관리

Amazon Rekognition Custom Labels 콘솔을 사용하여 레이블을 관리할 수 있습니다. 레이블 관리를 위한 특정 API는 없습니다. 레이블은 `CreateDataset`로 데이터 세트를 생성하거나 `UpdateDatasetEntries`로 데이터 세트에 이미지를 더 추가할 때 데이터 세트에 추가됩니다.

주제

- 레이블 관리(콘솔)
- 레이블 관리(SDK)

레이블 관리(콘솔)

Amazon Rekognition Custom Labels 콘솔을 사용하여 데이터 세트에 레이블을 추가, 변경 또는 제거할 수 있습니다. 데이터 세트에 레이블을 추가하려면 새로 만든 레이블을 추가하거나 Rekognition의 기존 데이터 세트에서 레이블을 가져올 수 있습니다.

주제

- 새 레이블 추가(콘솔)

- 레이블 변경 및 제거(콘솔)

새 레이블 추가(콘솔)

데이터 세트에 추가하려는 새 레이블을 지정할 수 있습니다.

편집 창을 사용하여 레이블을 추가합니다.

새 레이블을 추가하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 사용할 프로젝트를 선택합니다. 프로젝트의 세부 정보 페이지가 표시됩니다.
6. 훈련 데이터 세트에 레이블을 추가하려면 훈련 탭을 선택합니다. 그렇지 않으면 테스트 탭을 선택하여 테스트 데이터 세트에 레이블을 추가하세요.
7. 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.
8. 데이터 세트 갤러리의 레이블 항목에서 레이블 관리를 선택하여 레이블 관리 대화 상자를 엽니다.
9. 편집 상자에 새 레이블 이름을 입력합니다.
10. 레이블 추가를 선택합니다.
11. 필요한 레이블을 모두 만들 때까지 9단계와 10단계를 반복합니다.
12. 저장을 선택하여 추가한 레이블을 저장합니다.

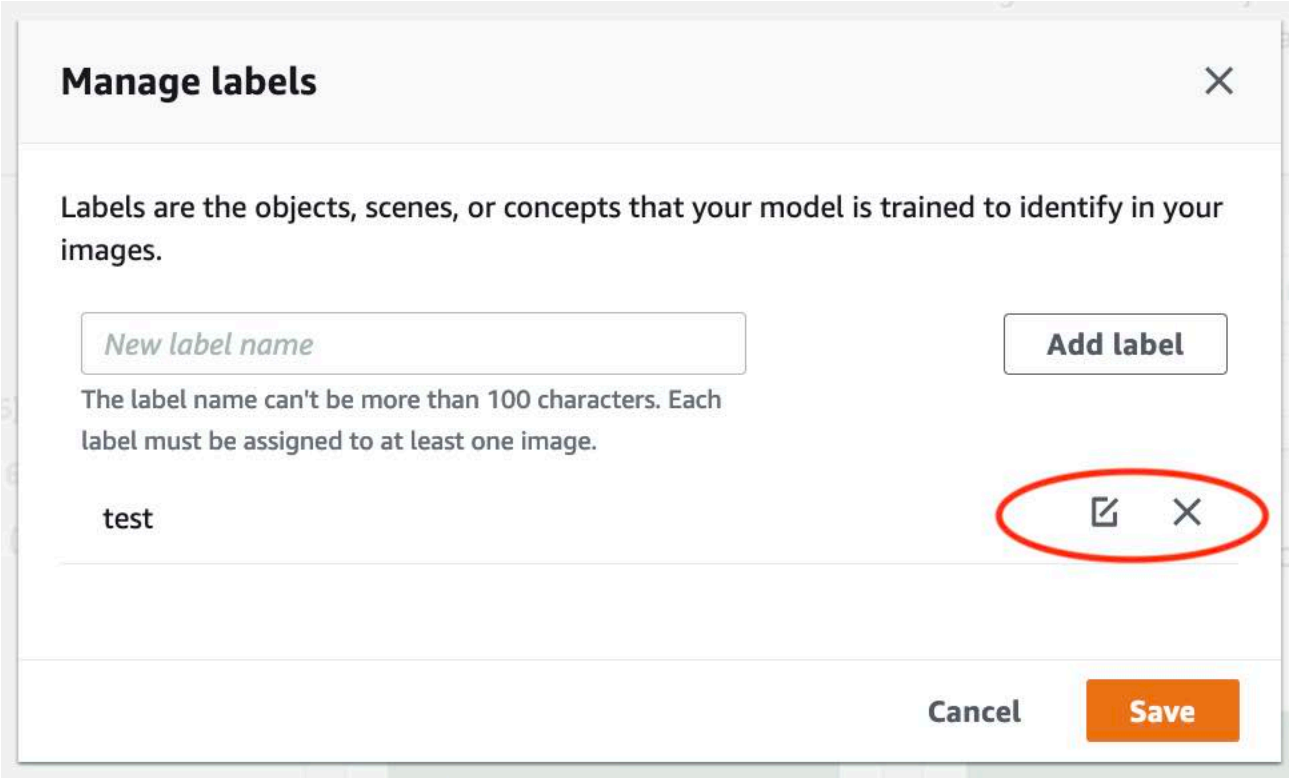
레이블 변경 및 제거(콘솔)

데이터 세트에 레이블을 추가한 후 이름을 바꾸거나 레이블을 제거할 수 있습니다. 이미지에 지정되지 않은 레이블만 제거할 수 있습니다.

기존 레이블의 이름을 바꾸거나 제거하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.

4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 사용할 프로젝트를 선택합니다. 프로젝트의 세부 정보 페이지가 표시됩니다.
6. 훈련 데이터 세트에 레이블을 변경하거나 삭제하려면 훈련 탭을 선택합니다. 그렇지 않으면 테스트 탭을 선택하여 테스트 데이터 세트의 레이블을 변경하거나 삭제할 수 있습니다.
7. 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.
8. 데이터 세트 갤러리의 레이블 항목에서 레이블 관리를 선택하여 레이블 관리 대화 상자를 엽니다.
9. 편집 또는 삭제할 레이블을 선택합니다.



   a. 삭제 아이콘(X)을 선택하면 목록에서 레이블이 제거됩니다.
   b. 레이블을 변경하려면 편집 아이콘(연필과 종이 패드)을 선택하고 편집 상자에 새 레이블 이름을 입력합니다.
10. 저장을 선택하여 변경 사항을 저장합니다.

레이블 관리(SDK)

데이터 세트 레이블을 관리하는 고유한 API는 없습니다. `CreateDataset`로 데이터 세트를 생성하면 매니페스트 파일 또는 복사된 데이터 세트에 있는 레이블이 초기 레이블 세트를 생성합니다. `UpdateDatasetEntries` API를 사용하여 이미지를 더 추가하면 항목에 있는 새 레이블이 데이터 세

트에 추가됩니다. 자세한 정보는 더 많은 이미지 추가(SDK)을 참조하세요. 데이터 세트에서 레이블을 삭제하려면 데이터 세트에서 모든 레이블 주석을 제거해야 합니다.

데이터 세트에서 레이블을 삭제하려면

1. `ListDatasetEntries`를 직접 호출하여 데이터 세트 항목을 가져옵니다. 예제 코드는 데이터 세트 항목 나열(SDK) 항목을 참조하세요.
2. 파일에서 모든 레이블 주석을 제거합니다. 자세한 정보는 매니페스트 파일의 이미지 수준 레이블 및 the section called “매니페스트 파일의 객체 위치 파악” 섹션을 참조하세요.
3. 파일을 사용하여 `UpdateDatasetEntries` API로 데이터 세트를 업데이트하세요. 자세한 정보는 더 많은 이미지 추가(SDK)을 참조하세요.

이미지에 이미지 수준 레이블 지정

이미지 수준 레이블을 사용하여 이미지를 카테고리로 분류하는 모델을 훈련할 수 있습니다. 이미지 수준 레이블은 이미지에 객체, 장면 또는 개념이 포함되어 있음을 나타냅니다. 예를 들어 다음 이미지에는 강이 있습니다. 모델이 이미지를 강을 포함하는 것으로 분류하는 경우 강 이미지 수준 레이블을 추가합니다. 자세한 정보는 데이터 세트 목적 설정을 참조하세요.

이미지 수준 레이블이 포함된 데이터 세트에는 레이블이 두 개 이상 정의되어 있어야 합니다. 각 이미지에는 이미지의 객체, 장면 또는 개념을 식별하는 레이블이 하나 이상 지정되어야 합니다.

이미지에 이미지 수준 레이블을 지정하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 사용할 프로젝트를 선택합니다. 프로젝트의 세부 정보 페이지가 표시됩니다.
6. 왼쪽 탐색 창에서 데이터 세트를 선택합니다.
7. 훈련 데이터 세트에 레이블을 추가하려면 훈련 탭을 선택합니다. 그렇지 않으면 테스트 탭을 선택하여 테스트 데이터 세트에 레이블을 추가하세요.
8. 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.

9. 이미지 갤러리에서 레이블을 추가할 이미지를 하나 이상 선택합니다. 한 번에 하나의 페이지에 있는 이미지만 선택할 수 있습니다. 한 페이지에서 연속된 이미지 범위를 선택하려면:
   a. 범위에 있는 첫 번째 이미지를 선택합니다.
   b. Shift 키를 길게 누릅니다.
   c. 마지막 이미지 범위를 선택합니다. 첫 번째 이미지와 두 번째 이미지 사이의 이미지도 선택됩니다.
   d. Shift 키를 놓습니다.
10. 이미지 수준 레이블 지정을 선택합니다.
11. 선택한 이미지에 이미지 수준 레이블 지정 대화 상자에서 이미지에 할당하려는 레이블을 선택합니다.
12. 지정을 선택하여 이미지에 레이블을 지정합니다.
13. 모든 이미지에 필요한 레이블이 주석으로 달릴 때까지 레이블 지정을 반복합니다.
14. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

이미지 수준 레이블 지정(SDK)

`UpdateDatasetEntries` API를 사용하여 이미지에 할당된 이미지 수준 레이블을 추가하거나 업데이트할 수 있습니다. `UpdateDatasetEntries`는 하나 이상의 JSON 라인을 사용합니다. 각 JSON 라인은 하나의 이미지를 나타냅니다. 이미지 수준 레이블이 있는 이미지의 경우 JSON 라인은 다음과 비슷합니다.

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-metadata":
{"confidence":0.95,"job-name":"labeling-job/testclconsolebucket","class-name":"Echo
 Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

`source-ref` 필드는 이미지 위치를 나타냅니다. JSON 라인에는 이미지에 지정된 이미지 수준 레이블도 포함됩니다. 자세한 정보는 the section called “매니페스트 파일의 이미지 수준 레이블”을 참조하세요.

이미지에 이미지 수준 레이블을 지정하려면

1. `ListDatasetEntries`를 사용하여 기존 이미지의 get JSON 라인을 가져옵니다. `source-ref` 필드에 레이블을 지정하려는 이미지의 위치를 지정합니다. 자세한 정보는 데이터 세트 항목 나열(SDK)을 참조하세요.

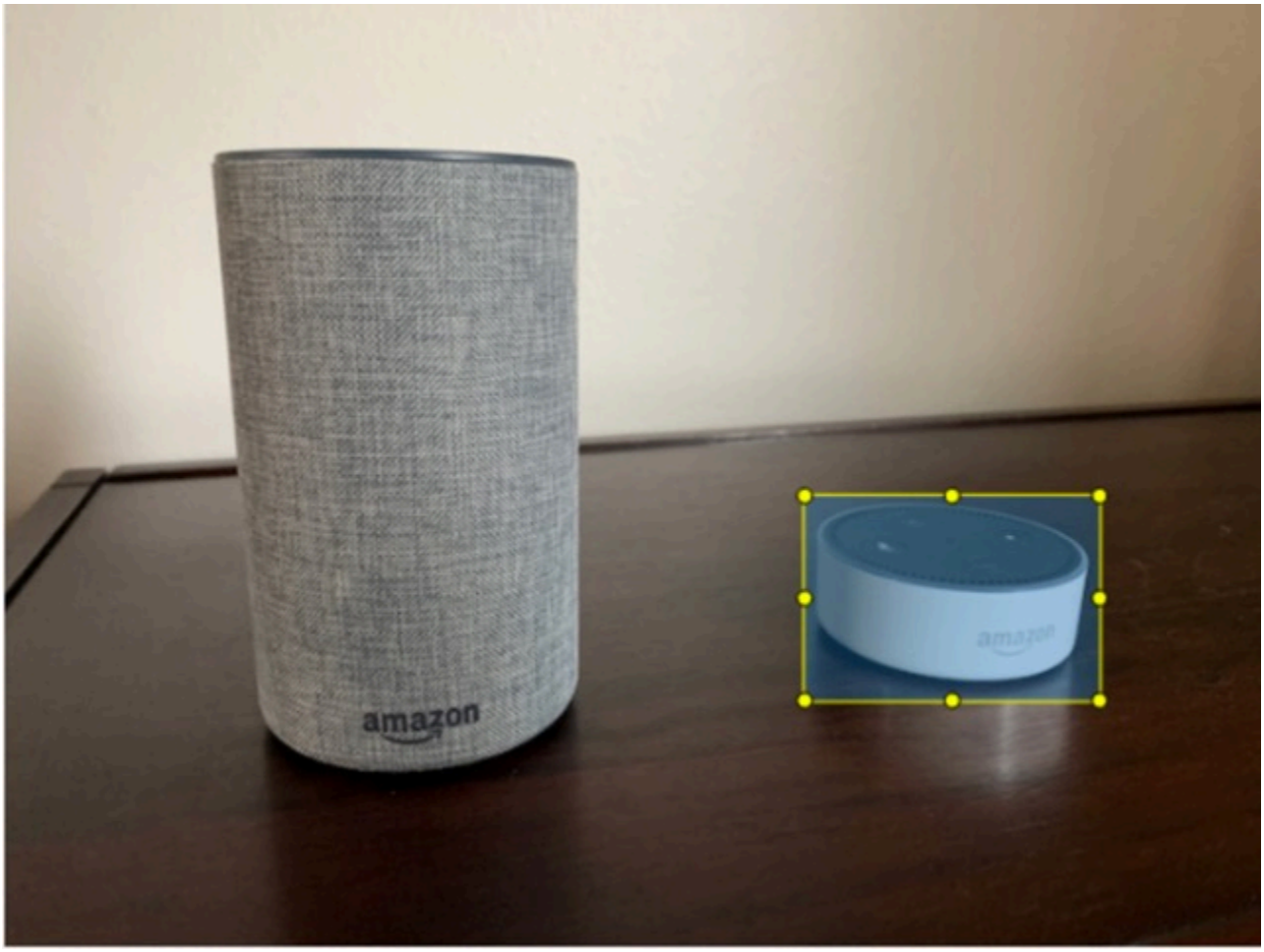
2. 매니페스트 파일의 이미지 수준 레이블의 정보를 사용하여 이전 단계에서 반환된 JSON 라인을 업데이트하세요.
3. 이미지를 업데이트하려면 `UpdateDatasetEntries`를 직접 호출하세요. 자세한 정보는 데이터 세트에 더 많은 이미지 추가을 참조하세요.

경계 상자로 객체에 레이블 지정

모델이 이미지 내 객체의 위치를 감지하도록 하려면 객체가 무엇이고 이미지 내 어디에 있는지 식별해야 합니다. 경계 상자는 이미지에서 객체를 분리하는 상자입니다. 경계 상자를 사용하여 동일한 이미지에서 서로 다른 객체를 감지하도록 모델을 훈련할 수 있습니다. 경계 상자에 레이블을 지정하여 객체를 식별합니다.

> **Note**
>
> 이미지 수준 레이블로 객체, 장면, 개념을 찾도록 모델을 훈련하는 경우에는 이 단계를 수행할 필요가 없습니다.

예를 들어 Amazon Echo Dot 디바이스를 탐지하는 모델을 훈련하려면 이미지의 각 Echo Dot 주위에 경계 상자를 그리고 Echo Dot이라는 레이블을 경계 상자에 지정합니다. 다음 이미지는 Echo Dot 디바이스 주위의 경계 상자를 보여줍니다. 이미지에는 경계 상자가 없는 Amazon Echo도 포함되어 있습니다.



경계 상자가 있는 객체 찾기(콘솔)

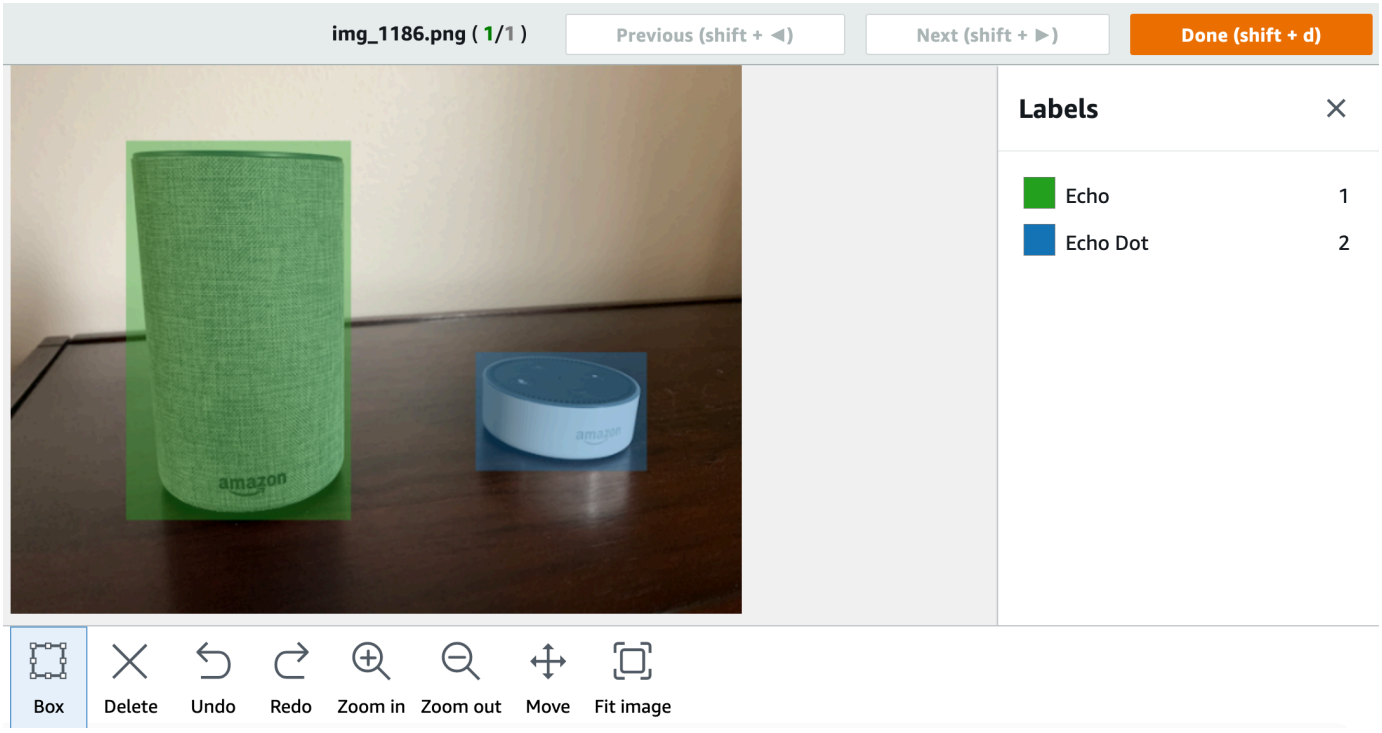
이 절차에서는 콘솔을 사용하여 이미지의 객체 주위에 경계 상자를 그립니다. 경계 상자에 레이블을 지정하여 이미지 내의 객체를 식별할 수도 있습니다.

> **Note**
>
> Safari 브라우저를 사용하여 이미지에 경계 상자를 추가할 수는 없습니다. 지원되는 브라우저에 대한 내용은 Amazon Rekognition Custom Labels 설정 항목을 참조하세요.

경계 상자를 추가하려면 먼저 데이터 세트에 하나 이상의 레이블을 추가해야 합니다. 자세한 정보는 새 레이블 추가(콘솔)을 참조하세요.

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.

3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 사용할 프로젝트를 선택합니다. 프로젝트의 세부 정보 페이지가 표시됩니다.
6. 프로젝트 세부 정보 페이지에서 레이블 이미지를 선택합니다.
7. 훈련 데이터 세트 이미지에 경계 상자를 추가하려면 훈련 탭을 선택하세요. 그렇지 않으면 테스트 탭을 선택하여 테스트 데이터 세트 이미지에 경계 상자를 추가할 수 있습니다.
8. 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.
9. 이미지 갤러리에서 경계 상자를 추가할 이미지를 선택합니다.
10. 경계 상자 그리기를 선택합니다. 경계 상자 편집기가 표시되기 전에 일련의 팁이 표시됩니다.
11. 오른쪽의 레이블 창에서 경계 상자에 지정할 레이블을 선택합니다.
12. 그리기 도구에서 원하는 객체의 왼쪽 상단 영역에 포인터를 놓습니다.
13. 마우스 왼쪽 버튼을 누르고 객체 주위에 상자를 그립니다. 경계 상자를 객체에 최대한 가깝게 그리세요.
14. 마우스 버튼을 놓습니다. 경계 상자가 강조 표시됩니다.
15. 레이블을 지정할 이미지가 더 많으면 다음을 선택합니다. 그렇지 않으면 완료를 선택하여 레이블 지정을 마칩니다.



16. 객체가 포함된 각 이미지에 경계 상자를 만들 때까지 1~7단계를 반복합니다.

17. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.
18. 종료를 선택하여 레이블 지정 모드를 종료합니다.

경계 상자로 객체 찾기(SDK)

UpdateDatasetEntries API를 사용하여 이미지에 대한 객체 위치 정보를 추가하거나 업데이트할 수 있습니다. UpdateDatasetEntries는 하나 이상의 JSON 라인을 사용합니다. 각 JSON 라인은 하나의 이미지를 나타냅니다. 객체 위치 파악의 경우 JSON 라인은 다음과 유사합니다.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id": 1,
 "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0, "top": 65,
 "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata": {"objects":
 [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo", "1": "Echo Dot"},
 "type": "groundtruth/object-detection", "human-annotated": "yes", "creation-date":
 "2013-11-18T02:53:27", "job-name": "my job"}}
```

source-ref 필드는 이미지 위치를 나타냅니다. JSON 라인에는 이미지의 각 객체에 대한 레이블이 지정된 경계 상자도 포함됩니다. 자세한 정보는 the section called “매니페스트 파일의 객체 위치 파악”을 참조하세요.

이미지에 경계 상자를 지정하려면

1. ListDatasetEntries를 사용하여 기존 이미지의 get JSON 라인을 가져옵니다. source-ref 필드에 이미지 수준 레이블을 지정할 이미지의 위치를 지정합니다. 자세한 정보는 데이터 세트 항목 나열(SDK)을 참조하세요.
2. 매니페스트 파일의 객체 위치 파악의 정보를 사용하여 이전 단계에서 반환된 JSON 라인을 업데이트하세요.
3. 이미지를 업데이트하려면 UpdateDatasetEntries를 직접 호출하세요. 자세한 정보는 데이터 세트에 더 많은 이미지 추가을 참조하세요.

데이터 세트 디버깅

데이터 세트 생성 중에는 터미널 오류와 비터미널 오류라는 두 가지 유형의 오류가 발생할 수 있습니다. 터미널 오류로 인해 데이터 세트 생성 또는 업데이트가 중단될 수 있습니다. 비터미널 오류는 발생해도 데이터 세트 생성 또는 업데이트가 중단되지 않습니다.

주제

- 터미널 오류
- 비터미널 오류

터미널 오류

터미널 오류에는 두 가지 유형이 있습니다. 하나는 데이터 세트 생성에 실패하게 하는 파일 오류이고, 다른 하나는 Amazon Rekognition Custom Labels가 데이터 세트에서 제거하는 콘텐츠 오류입니다. 콘텐츠 오류가 너무 많으면 데이터 세트 생성이 실패합니다.

주제

- 터미널 파일 오류
- 터미널 콘텐츠 오류

터미널 파일 오류

다음은 파일 오류입니다. `DescribeDataset`를 직접 호출하고 `Status` 및 `StatusMessage` 필드를 확인하여 파일 오류에 대한 정보를 얻을 수 있습니다. 예제 코드는 데이터 세트 설명(SDK) 항목을 참조하세요.

- ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT
- ERROR_MANIFEST_SIZE_TOO_LARGE.
- ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM
- ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET
- ERROR_TOO_MANY_RECORDS_IN_ERROR
- ERROR_MANIFEST_TOO_MANY_LABELS
- ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

오류 메시지

매니페스트 파일 확장명 또는 콘텐츠가 유효하지 않습니다.

훈련 또는 테스트 매니페스트 파일에 파일 확장자가 없거나 해당 내용이 유효하지 않습니다.

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT 오류를 수정하려면

- 훈련 매니페스트 파일과 테스트 매니페스트 파일 모두에서 다음과 같은 가능한 원인을 확인하세요.
  - 매니페스트 파일에 파일 확장명이 없습니다. 일반적으로 파일 확장자는 `.manifest`입니다.
  - 매니페스트 파일의 Amazon S3 버킷 또는 키를 찾을 수 없습니다.

ERROR_MANIFEST_SIZE_TOO_LARGE

오류 메시지

매니페스트 파일 크기가 지원되는 최대 크기를 초과합니다.

훈련 또는 테스트 매니페스트 파일 크기(바이트)가 너무 큽니다. 자세한 정보는 Amazon Rekognition Custom Labels 지침 및 할당량을 참조하세요. 매니페스트 파일은 최대 JSON 라인 수보다 적으면서도 최대 파일 크기를 초과할 수 있습니다.

Amazon Rekognition Custom Labels 콘솔로는 매니페스트 파일 크기가 지원되는 최대 크기를 초과합니다 오류를 수정할 수 없습니다.

ERROR_MANIFEST_SIZE_TOO_LARGE 오류를 수정하려면

1. 훈련 및 테스트 매니페스트 중에 어떤 것이 최대 파일 크기를 초과하는지 확인하세요.
2. 매니페스트 파일에서 너무 큰 JSON 라인 수를 줄이세요. 자세한 정보는 매니페스트 파일 생성을 참조하세요.

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

오류 메시지

매니페스트 파일에 행이 너무 많습니다.

추가 정보

매니페스트 파일의 JSON 라인 수(이미지 수)가 허용 한도보다 큽니다. 이미지 수준 모델과 객체 위치 모델의 한도는 다릅니다. 자세한 정보는 Amazon Rekognition Custom Labels 지침 및 할당량을 참조하세요.

JSON 라인 수가 `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM` 한도에 도달할 때까지 JSON 라인 오류가 검증됩니다.

Amazon Rekognition Custom Labels로는 ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM 오류를 수정할 수 없습니다.

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM 오류를 수정하려면

- 매니페스트에 있는 JSON 라인 수를 줄입니다. 자세한 정보는 매니페스트 파일 생성을 참조하세요.

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

오류 메시지

S3 버킷 권한이 올바르지 않습니다.

Amazon Rekognition Custom Labels가 훈련 및 테스트 매니페스트 파일이 들어 있는 하나 이상의 버킷에 대한 권한이 없습니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET 오류를 수정하려면

- 훈련 및 테스트 매니페스트가 포함된 버킷의 권한을 확인하세요. 자세한 정보는 2단계: Amazon Rekognition Custom Labels 콘솔 권한 설정을 참조하세요.

ERROR_TOO_MANY_RECORDS_IN_ERROR

오류 메시지

매니페스트 파일에 터미널 오류가 너무 많습니다.

ERROR_TOO_MANY_RECORDS_IN_ERROR 오류를 수정하려면

- 터미널 콘텐츠 오류가 있는 JSON 라인(이미지) 수를 줄이세요. 자세한 정보는 터미널 매니페스트 콘텐츠 오류을 참조하세요.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_MANIFEST_TOO_MANY_LABELS

오류 메시지

매니페스트 파일에 레이블이 너무 많습니다.

추가 정보

매니페스트(데이터 세트)의 고유 레이블 수가 허용된 한도를 초과했습니다. 훈련 데이터 세트를 분할하여 테스트 데이터 세트를 만드는 경우 분할 후 레이블 수가 결정됩니다.

ERROR_MANIFEST_TOO_MANY_LABELS 오류를 수정하려면(콘솔)

- 데이터 세트에서 레이블을 제거합니다. 자세한 정보는 레이블 관리을 참조하세요. 데이터 세트의 이미지와 경계 상자에서 레이블이 자동으로 제거됩니다.

ERROR_MANIFEST_TOO_MANY_LABELS 오류를 수정하려면(JSON 라인)

- 이미지 수준 JSON 라인이 있는 매니페스트: 이미지에 하나의 레이블이 있는 경우 원하는 레이블을 사용하는 이미지의 JSON 라인을 제거하세요. JSON 라인에 여러 레이블이 포함된 경우 원하는 레이블의 JSON 객체만 제거하세요. 자세한 정보는 이미지에 여러 이미지 수준 레이블 추가을 참조하세요.

  객체 위치가 있는 매니페스트 JSON 라인: 제거하려는 레이블의 경계 상자 및 관련 레이블 정보를 제거합니다. 원하는 레이블이 포함된 각 JSON 라인에 대해 이 작업을 수행하세요. `class-map` 배열과 `objects` 및 `annotations` 배열의 해당 객체를 레이블에서 제거해야 합니다. 자세한 정보는 매니페스트 파일의 객체 위치 파악을 참조하세요.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

오류 메시지

매니페스트 파일에 레이블 지정된 이미지가 부족하여 데이터 세트를 배포할 수 없습니다.

Amazon Rekognition Custom Labels가 훈련 데이터 세트를 분할하여 테스트 데이터 세트를 생성할 때 데이터 세트 배포가 발생합니다. `DistributeDatasetEntries` API를 직접 호출하여 데이터 세트를 분할할 수도 있습니다.

ERROR_MANIFEST_TOO_MANY_LABELS 오류를 수정하려면

- 훈련 데이터 세트에 레이블이 지정된 이미지를 더 추가하세요.

터미널 콘텐츠 오류

다음은 터미널 콘텐츠 오류입니다. 데이터 세트를 생성하는 동안 터미널 콘텐츠 오류가 있는 이미지는 데이터 세트에서 제거됩니다. 해당 데이터 세트는 여전히 훈련에 사용할 수 있습니다. 콘텐츠 오류가 너무 많으면 데이터 세트 또는 업데이트가 실패합니다. 데이터 세트 작업과 관련된 터미널 콘텐츠 오류는 콘솔에 표시되지 않으며 `DescribeDataset` 또는 다른 API에서 반환되지 않습니다. 데이터 세트에서 이미지나 주석이 누락된 것을 발견하면 데이터 세트 매니페스트 파일에 다음과 같은 문제가 있는지 확인하세요.

- JSON 라인 길이가 너무 깁니다. 최대 길이는 100,000자입니다.
- JSON 라인에서 `source-ref` 값이 누락되었습니다.
- JSON 라인의 `source-ref` 값 형식이 잘못되었습니다.
- JSON 라인의 내용이 유효하지 않습니다.
- `source-ref` 필드의 값이 두 번 이상 나타납니다. 데이터 세트에서 이미지는 한 번만 참조될 수 있습니다.

`source-ref` 필드에 대한 자세한 내용은 매니페스트 파일 생성 항목을 참조하세요.

비터미널 오류

다음은 데이터 세트 생성 또는 업데이트 중에 발생할 수 있는 비터미널 오류입니다. 이러한 오류는 전체 JSON 라인을 무효화하거나 JSON 라인 내의 주석을 무효화할 수 있습니다. JSON 라인에 오류가 있는 경우 훈련에 사용되지 않습니다. JSON 라인 내의 주석에 오류가 있는 경우에도 JSON 라인은 여전히 학습에 사용되지만 주석이 깨지지는 않습니다. JSON 라인에 관한 자세한 정보는 매니페스트 파일 생성 항목을 참조하세요.

콘솔에서 `ListDatasetEntries` API를 직접 호출하여 비터미널 오류에 액세스할 수 있습니다. 자세한 정보는 데이터 세트 항목 나열(SDK)을 참조하세요.

훈련 중에 다음과 같은 오류도 반환됩니다. 모델을 훈련하기 전에 이러한 오류를 수정하는 것이 좋습니다. 자세한 내용은 비터미널 JSON 라인 검증 오류 섹션을 참조하세요.

- ERROR_NO_LABEL_ATTRIBUTES

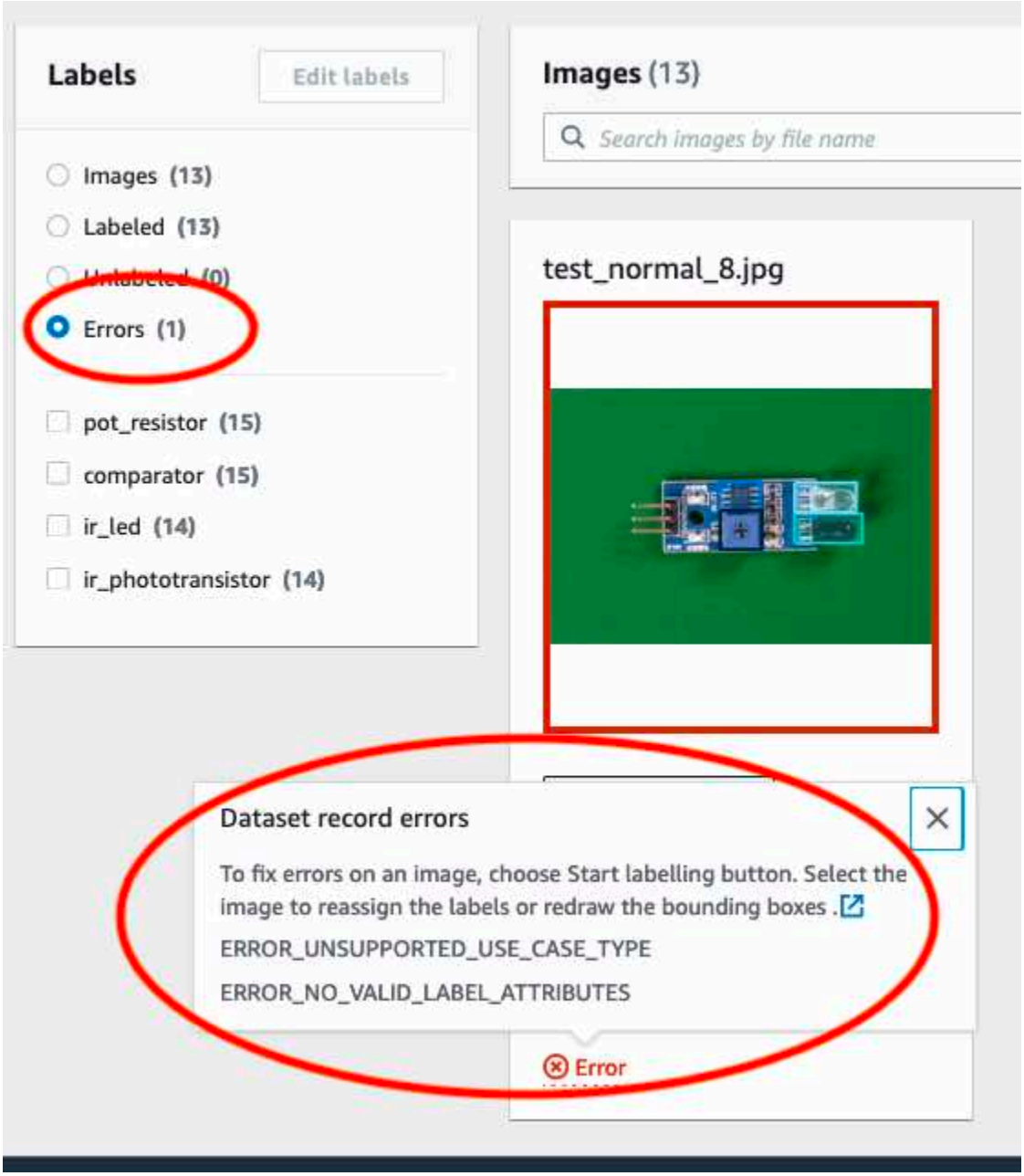
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_NO_VALID_LABEL_ATTRIBUTES
- ERROR_INVALID_BOUNDING_BOX
- ERROR_INVALID_IMAGE_DIMENSION
- ERROR_BOUNDING_BOX_TOO_SMALL
- ERROR_NO_VALID_ANNOTATIONS
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_TOO_MANY_BOUNDING_BOXES
- ERROR_UNSUPTED_USE_CASE_TYPE
- ERROR_INVALID_LABEL_NAME_LENGTH

비터미널 오류에 액세스

콘솔을 사용하여 데이터 세트에서 비터미널 오류가 있는 이미지를 찾을 수 있습니다. `ListDatasetEntries` API를 직접 호출하여 오류 메시지를 받을 수도 있습니다. 자세한 정보는 데이터 세트 항목 나열(SDK)을 참조하세요.

비터미널 오류에 액세스하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 사용할 프로젝트를 선택합니다. 프로젝트의 세부 정보 페이지가 표시됩니다.
6. 훈련 데이터 세트의 비터미널 오류를 보려면 훈련 탭을 선택하세요. 그렇지 않으면 테스트 탭을 선택하여 테스트 데이터 세트의 비터미널 오류를 확인할 수 있습니다.
7. 데이터 세트 갤러리의 레이블 항목에서 오류를 선택합니다. 데이터 세트 갤러리는 오류가 있는 이미지만 표시하도록 필터링됩니다.
8. 이미지 아래에 있는 오류를 선택하면 오류 코드를 확인할 수 있습니다. 비터미널 JSON 라인 검증 오류의 정보를 사용하여 오류를 수정하세요.



Amazon Rekognition Custom Labels 모델 훈련

Amazon Rekognition Custom Labels 콘솔 또는 Amazon Rekognition Custom Labels API를 사용하여 모델을 훈련할 수 있습니다. 모델 훈련이 실패할 경우 실패한 모델 훈련 디버깅의 정보를 사용하여 실패 원인을 찾아내세요.

> **Note**
>
> 모델을 성공적으로 훈련하는 데 걸리는 시간만큼 요금이 부과됩니다. 일반적으로 교육을 완료하는 데 30분에서 24시간이 소요됩니다. 자세한 정보는 훈련 시간 섹션을 참조하세요.

모델을 훈련할 때마다 새 버전의 모델이 생성됩니다. Amazon Rekognition Custom Labels는 프로젝트 이름과 모델 생성 당시의 타임스탬프를 조합하여 모델 이름을 생성합니다.

모델을 훈련하기 위해 Amazon Rekognition Custom Labels는 소스 훈련 및 테스트 이미지의 사본을 만듭니다. 기본적으로 복사된 이미지는 AWS가 소유하고 관리하는 키로 암호화됩니다. 사용자의 AWS KMS key 항목을 사용하는 방법도 있습니다. 자체 KMS 키를 사용하는 경우 KMS 키에 다음 권한이 필요합니다.

- kms:CreateGrant
- kms:DescribeKey

자세한 내용은 AWS Key Management Service 개념을 참조하세요. 소스 이미지는 영향을 받지 않습니다.

KMS 서버 측 암호화(SSE-KMS)를 사용하여 Amazon S3 버킷의 훈련 및 테스트 이미지를 암호화한 후 Amazon Rekognition Custom Labels에서 복사할 수 있습니다. Amazon Rekognition Custom Labels가 이미지에 액세스할 수 있도록 허용하려면 AWS 계정에 KMS 키에 대한 다음 권한이 필요합니다.

- kms:GenerateDataKey
- kms:Decrypt

자세한 내용은 AWS Key Management Service에 저장된 KMS 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.

모델을 훈련한 후 성능을 평가하고 개선할 수 있습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 섹션을 참조하세요.

모델 태그 지정과 같은 다른 모델 작업에 대한 내용은 Amazon Rekognition Custom Labels 모델 관리 항목을 참조하세요.

주제

- 모델 훈련(콘솔)

- 모델 훈련(SDK)

모델 훈련(콘솔)

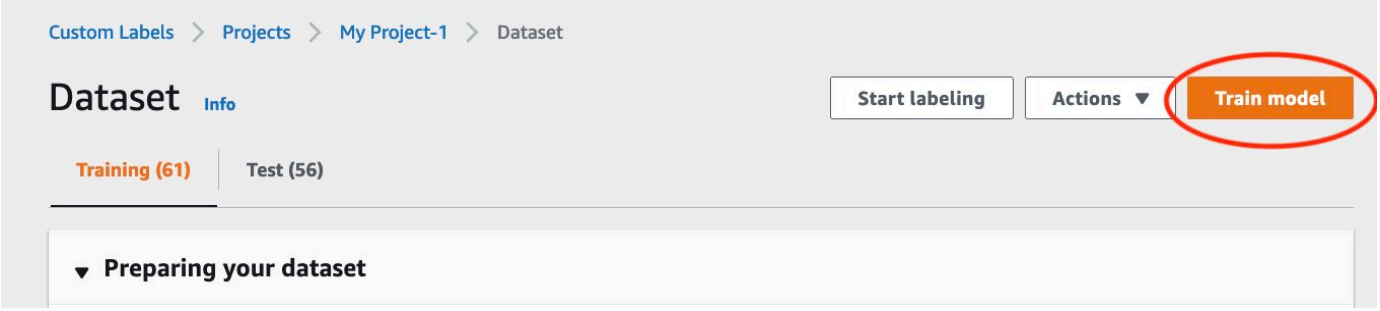
Amazon Rekognition Custom Labels 콘솔을 사용하여 모델을 훈련할 수 있습니다.

훈련을 하려면 훈련 데이터 세트와 테스트 데이터 세트를 갖춘 프로젝트가 필요합니다. 프로젝트에 테스트 데이터 세트가 없는 경우 Amazon Rekognition Custom Labels 콘솔은 훈련 중에 훈련 데이터 세트를 분할하여 프로젝트용 데이터 세트를 생성합니다. 선택한 이미지는 대표적인 샘플이며 훈련 데이터 세트에 사용되지 않습니다. 사용할 수 있는 대체 테스트 데이터 세트가 없는 경우에만 훈련 데이터 세트를 분할하는 것이 좋습니다. 훈련 데이터 세트를 분할하면 훈련에 사용할 수 있는 이미지 수가 줄어듭니다.

> **Note**
>
> 모델을 훈련하는 데 걸리는 시간만큼 요금이 부과됩니다. 자세한 정보는 훈련 시간 섹션을 참조하세요.

모델을 훈련하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
4. 프로젝트 페이지에서 훈련하려는 모델이 포함된 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 Train model을 선택합니다.



6. (선택 사항) 자체 AWS KMS 암호화 키를 사용하려면 다음을 수행하세요.
   a. 이미지 데이터 암호화에서 암호화 설정 사용자 지정(고급)을 선택합니다.

b. encryption.aws_kms_key에서 키의 Amazon 리소스 이름(ARN)을 입력하거나 기존 AWS KMS 키를 선택합니다. 새 키를 생성하려면 AWS IMS 키 생성을 선택합니다.

7. (선택 사항) 모델에 태그를 추가하려면 다음을 수행합니다.

   a. 태그 항목에서 새로운 태그 추가를 선택합니다.

   b. 다음을 입력합니다.

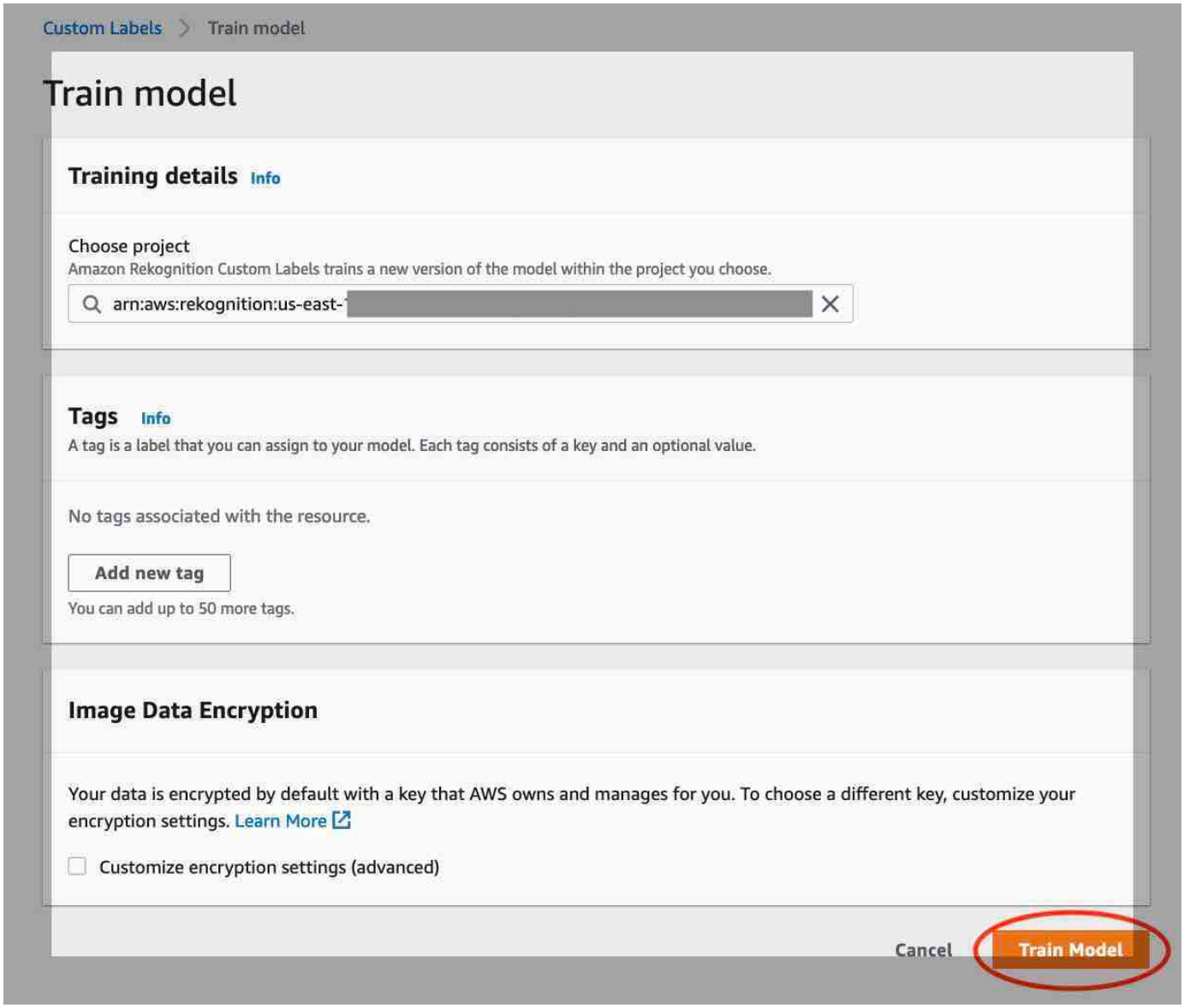
      i. 키에 있는 키의 이름

      ii. 값에 있는 키의 값

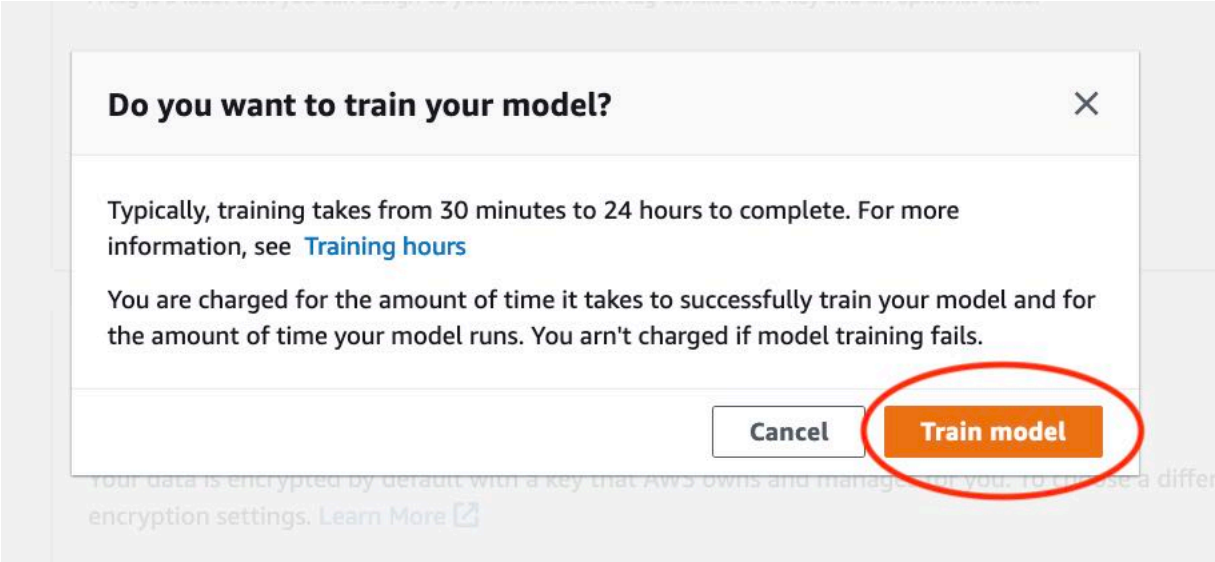
   c. 태그를 더 추가하려면 6a단계와 6b단계를 반복합니다.

   d. (선택 사항) 태그를 제거하려면 제거할 태그 옆의 제거를 선택합니다. 이전에 저장한 태그를 제거하는 경우 변경 내용을 저장하면 해당 태그가 제거됩니다.

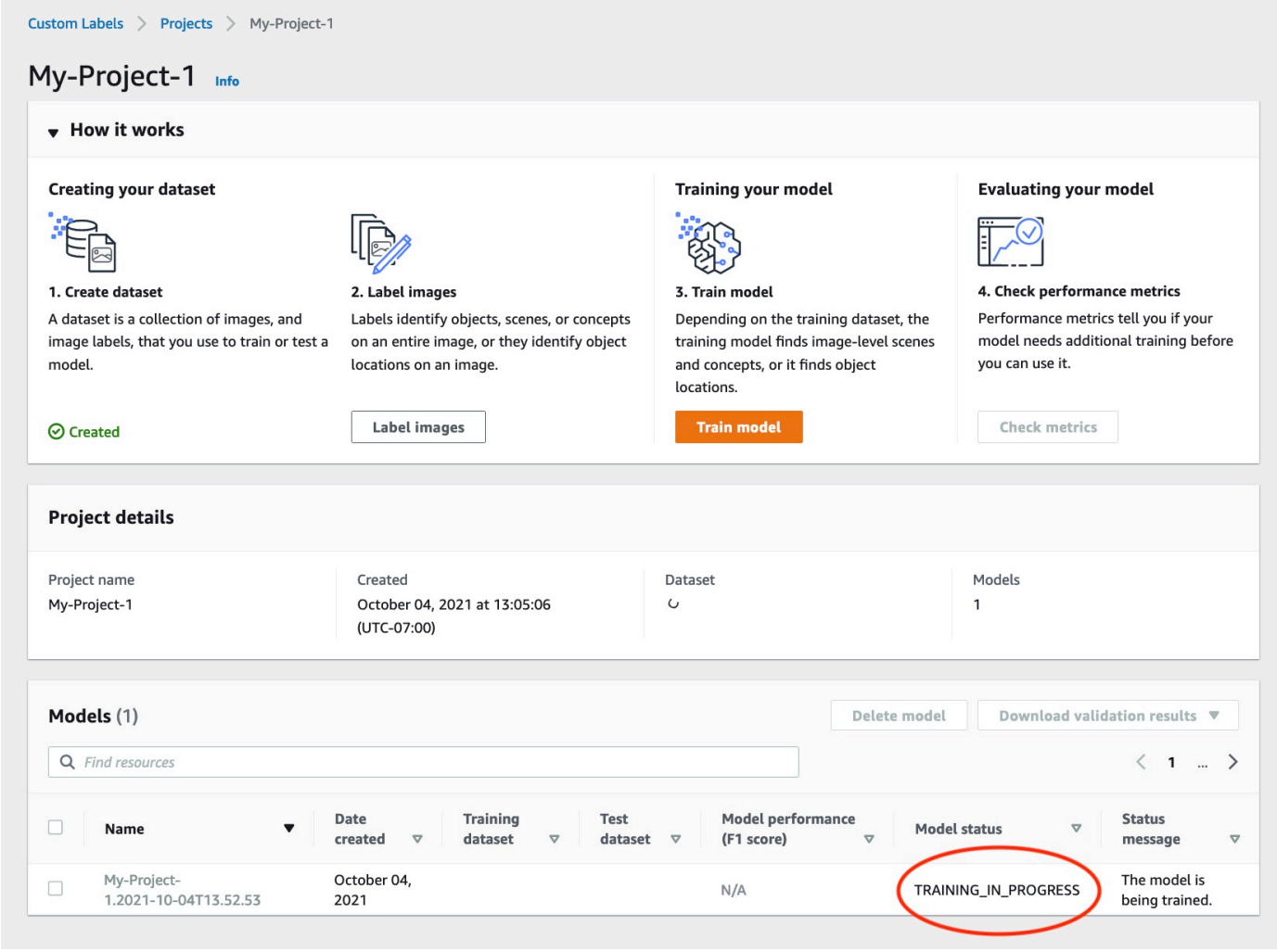
8. 모델 훈련 페이지에서 모델 훈련을 선택합니다. 프로젝트 선택 편집 상자에 프로젝트의 Amazon 리소스 이름(ARN)이 보일 겁니다. 그렇지 않은 경우 프로젝트의 ARN을 입력합니다.



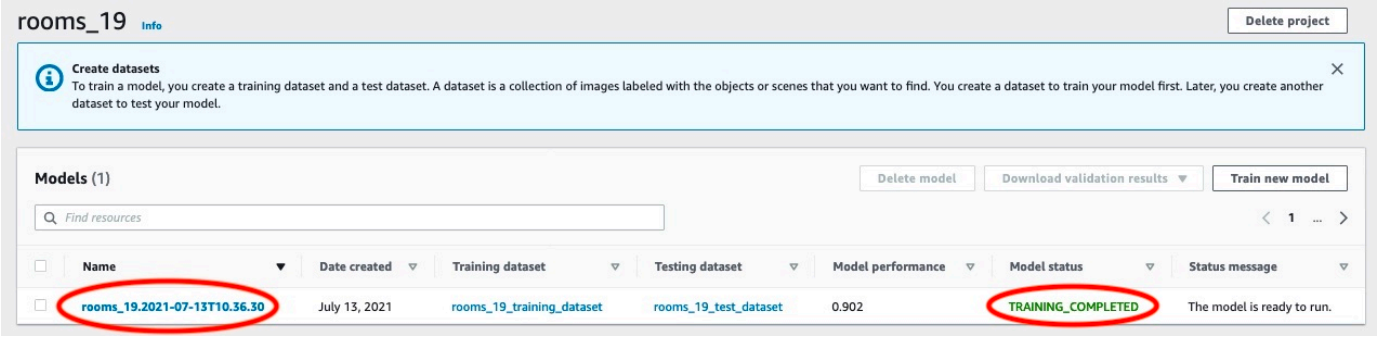
9. 모델을 훈련하고 싶으신가요? 대화 상자에서 모델 훈련을 선택합니다.



10. 프로젝트 페이지의 모델 항목에서 훈련이 진행 중인 `Model Status` 열의 현재 상태를 확인할 수 있습니다. 모델 훈련은 완료하는 데 다소 시간이 걸립니다.



11. 훈련이 완료되면 모델 이름을 선택합니다. 모델 상태가 TRAINING_COMPLETED로 표시되면 훈련이 끝났다는 뜻입니다. 훈련이 실패하면 실패한 모델 훈련 디버깅 항목을 읽어 보세요.



12. 다음 단계: 모델 평가 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 항목을 참조하세요.

모델 훈련(SDK)

CreateProjectVersion을 호출하여 모델을 훈련합니다. 모델을 훈련하려면 다음 정보가 필요합니다.

- 이름: 모델 버전의 고유 이름
- 프로젝트 ARN: 모델을 관리하는 프로젝트의 Amazon 리소스 이름(ARN)
- 훈련 결과 위치: 결과가 배치되는 Amazon S3 위치 콘솔 Amazon S3 버킷과 동일한 위치를 사용하거나 다른 위치를 선택할 수 있습니다. 다른 위치를 선택하는 것이 좋습니다. 이렇게 하면 권한을 설정하고 Amazon Rekognition Custom Labels 콘솔 사용의 훈련 출력과 이름 충돌이 발생할 가능성을 방지할 수 있기 때문입니다.

훈련은 프로젝트와 관련된 훈련 및 테스트 데이터 세트를 사용합니다. 자세한 내용은 데이터 세트 관리 섹션을 참조하세요.

> **Note**
>
> 선택 사항으로 프로젝트 외부에 있는 훈련 및 테스트 데이터 세트 매니페스트 파일을 지정할 수 있습니다. 외부 매니페스트 파일로 모델을 훈련한 후 콘솔을 열면 Amazon Rekognition Custom Labels가 훈련에 사용된 마지막 매니페스트 파일 세트를 사용하여 데이터 세트를 생성합니다. 더 이상 외부 매니페스트 파일을 지정하여 프로젝트의 모델 버전을 훈련할 수 없습니다. 자세한 내용은 CreateProjectVersion을 참조하세요.

CreateProjectVersion의 응답은 후속 요청에서 모델 버전을 식별하는 데 사용하는 ARN입니다. ARN을 사용하여 모델 버전을 보호할 수도 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 프로젝트의 보안 섹션을 참조하세요.

모델 버전 훈련을 완료하는 데 시간이 걸립니다. 이 주제의 Python 및 Java 예제는 훈련이 완료될 때까지 기다리는 데에 waiter를 사용합니다. Waiter는 특정 상태에 대해 폴링되는 유틸리티 메서드입니다. 또는 DescribeProjectVersions 항목을 호출하여 현재 훈련 상태를 확인할 수도 있습니다. Status 필드 값이 TRAINING_COMPLETED면 훈련이 완료된 것입니다. 훈련을 완료한 후 평가 결과를 검토하여 모델의 품질을 평가할 수 있습니다.

모델 훈련(SDK)

다음 예제는 프로젝트와 관련된 훈련 및 테스트 데이터 세트를 사용하여 모델을 훈련하는 방법을 보여줍니다.

모델을 훈련하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 프로젝트를 훈련하려면 다음 예제 코드를 사용하세요.

   AWS CLI

   다음 예제에서는 모델을 생성합니다. 훈련 데이터 세트를 분할하여 테스트 데이터 세트를 생성합니다. 다음을 바꿉니다.

   - my_project_arn 항목을 프로젝트의 Amazon 리소스 이름(ARN)으로 바꿉니다.
   - version_name 항목을 선택한 고유한 버전 이름으로 바꿉니다.
   - output_bucket 항목을 Amazon Rekognition Custom Labels 훈련 결과를 저장하는 Amazon S3 버킷의 이름으로 바꿉니다.
   - output_folder 항목을 훈련 결과가 저장되는 폴더의 이름으로 변경합니다.
   - (선택적 파라미터) --kms-key-id 항목을 AWS Key Management Service 고객 마스터 키의 식별자로 변경합니다.

```
aws rekognition create-project-version \
  --project-arn project_arn \
  --version-name version_name \
```

```
  --output-config '{"S3Bucket":"output_bucket", "S3KeyPrefix":"output_folder"}'
\
  --profile custom-labels-access
```

Python

다음 예제에서는 모델을 생성합니다. 다음 명령줄 인수를 제공하세요.

- `project_arn`: 프로젝트의 Amazon 리소스 이름(ARN)
- `version_name`: 선택한 모델의 고유한 버전 이름
- `output_bucket`: Amazon Rekognition Custom Labels가 훈련 결과를 저장하는 Amazon S3 버킷의 이름
- `output_folder`: 훈련 결과가 저장되는 폴더의 이름

다음 명령줄 파라미터를 제공하여 모델에 태그를 첨부할 수도 있습니다.

- `tag`: 모델에 첨부할 태그의 이름
- `tag_value`: 태그 값

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def train_model(rek_client, project_arn, version_name, output_bucket,
 output_folder, tag_key, tag_key_value):
    """
    Trains an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
```

```
   :param project_arn: The ARN of the project in which you want to train a
model.
   :param version_name: A version for the model.
   :param output_bucket: The S3 bucket that hosts training output.
   :param output_folder: The path for the training output within output_bucket
   :param tag_key: The name of a tag to attach to the model. Pass None to
exclude
   :param tag_key_value: The value of the tag. Pass None to exclude

   """

   try:
       #Train the model

       status=""
       logger.info("training model version %s for project %s",
           version_name, project_arn)


       output_config = json.loads(
           '{"S3Bucket": "'
           + output_bucket
           + '", "S3KeyPrefix": "'
           + output_folder
           + '" }  '
       )

       tags={}

       if tag_key is not None and tag_key_value is not None:
           tags = json.loads(
               '{"' + tag_key + '":"' + tag_key_value + '"}'
           )


       response=rek_client.create_project_version(
           ProjectArn=project_arn,
           VersionName=version_name,
           OutputConfig=output_config,
           Tags=tags
       )

       logger.info("Started training: %s", response['ProjectVersionArn'])
```

```
        # Wait for the project version training to complete.

        project_version_training_completed_waiter =
 rek_client.get_waiter('project_version_training_completed')
        project_version_training_completed_waiter.wait(ProjectArn=project_arn,
        VersionNames=[version_name])


        # Get the completion status.

 describe_response=rek_client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            logger.info("Status: %s", model['Status'])
            logger.info("Message: %s", model['StatusMessage'])
            status=model['Status']


        logger.info("finished training")

        return response['ProjectVersionArn'], status

    except ClientError as err:
        logger.exception("Couldn't create model: %s", err.response['Error']
['Message'] )
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to train a
 model"
    )

    parser.add_argument(
        "version_name", help="A version name of your choosing."
    )

    parser.add_argument(
```

```
        "output_bucket", help="The S3 bucket that receives the training
 results."
    )

    parser.add_argument(
        "output_folder", help="The folder in the S3 bucket where training
 results are stored."
    )

    parser.add_argument(
        "--tag_name",  help="The name of a tag to attach to the model",
 required=False
    )

    parser.add_argument(
        "--tag_value",  help="The value for the tag.", required=False
    )




def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Training model version {args.version_name} for project
 {args.project_arn}")

        # Train the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        model_arn, status=train_model(rekognition_client,
            args.project_arn,
            args.version_name,
            args.output_bucket,
            args.output_folder,
```

```
            args.tag_name,
            args.tag_value)


        print(f"Finished training model: {model_arn}")
        print(f"Status: {status}")


    except ClientError as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")
    except Exception as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

다음 예제는 모델을 훈련합니다. 다음 명령줄 인수를 제공하세요.

- `project_arn`: 프로젝트의 Amazon 리소스 이름(ARN)
- `version_name`: 선택한 모델의 고유한 버전 이름
- `output_bucket`: Amazon Rekognition Custom Labels가 훈련 결과를 저장하는 Amazon S3 버킷의 이름
- `output_folder`: 훈련 결과가 저장되는 폴더의 이름

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TrainModel {

    public static final Logger logger =
 Logger.getLogger(TrainModel.class.getName());

    public static String trainMyModel(RekognitionClient rekClient, String
 projectArn, String versionName,
            String outputBucket, String outputFolder) {

        try {

            OutputConfig outputConfig =
 OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            logger.log(Level.INFO, "Training Model for project {0}",
 projectArn);
            CreateProjectVersionRequest createProjectVersionRequest =
 CreateProjectVersionRequest.builder()

 .projectArn(projectArn).versionName(versionName).outputConfig(outputConfig).build();

            CreateProjectVersionResponse response =
 rekClient.createProjectVersion(createProjectVersionRequest);

            logger.log(Level.INFO, "Model ARN: {0}",
 response.projectVersionArn());
```

```
            logger.log(Level.INFO, "Training model...");

            // wait until training completes

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .versionNames(versionName)
                    .projectArn(projectArn)
                    .build();

            RekognitionWaiter waiter = rekClient.waiter();

            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionTrainingCompleted(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());
            }

            return response.projectVersionArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }
```

```
    public static void main(String args[]) {

        String versionName = null;
        String projectArn = null;
        String projectVersionArn = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
 <output_bucket> <output_folder>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to use.
 \n\n"
                + "   version_name - A version name for the model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
 training output. \n\n"
                + "   output_folder - The folder within the bucket that the
 training output is stored in. \n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        versionName = args[1];
        bucket = args[2];
        location = args[3];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Train model
            projectVersionArn = trainMyModel(rekClient, projectArn, versionName,
 bucket, location);

            System.out.println(String.format("Created model: %s for Project ARN:
 %s", projectVersionArn, projectArn));
```

```
            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. 훈련이 실패하면 실패한 모델 훈련 디버깅 항목을 읽어 보세요.

실패한 모델 훈련 디버깅

모델 훈련 중에 오류가 발생할 수 있습니다. Amazon Rekognition Custom Labels는 콘솔과 DescribeProjectVersions의 응답에서 훈련 오류를 보고합니다.

오류는 터미널(훈련을 계속할 수 없음)이거나 비터미널(훈련 계속 가능)입니다. 훈련 및 테스트 데이터 세트의 내용과 관련된 오류의 경우 검증 결과(매니페스트 요약, 훈련 및 테스트 검증 매니페스트)를 다운로드할 수 있습니다. 검증 결과의 오류 코드를 사용하여 이 항목에서 추가 정보를 찾아보세요. 이 항목에서는 매니페스트 파일 오류(매니페스트 파일 내용이 검증되기 전에 발생하는 터미널 오류)에 대한 정보도 제공합니다.

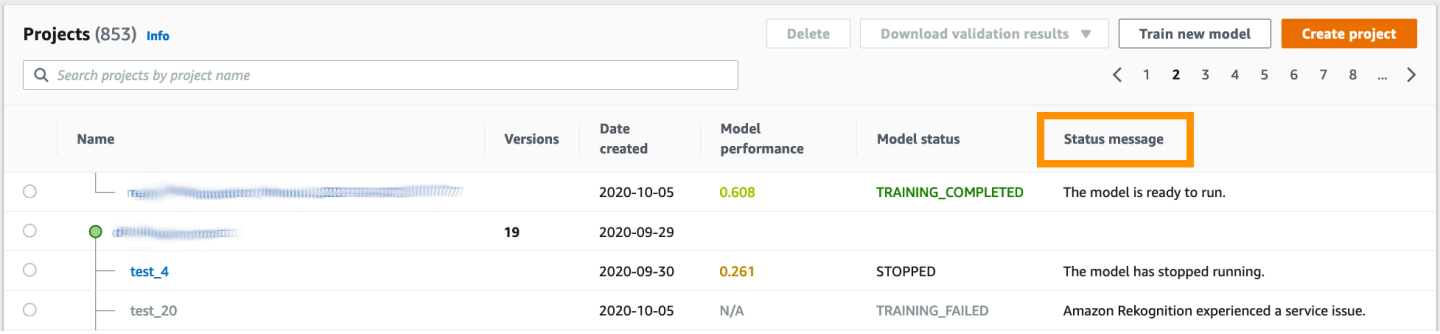
> **Note**
>
> 매니페스트는 데이터 세트의 콘텐츠를 저장하는 데 사용되는 파일입니다.

Amazon Rekognition Custom Labels 콘솔을 사용하여 일부 오류를 수정할 수 있습니다. 다른 오류의 경우 훈련 또는 테스트 매니페스트 파일을 업데이트해야 할 수도 있습니다. IAM 권한과 같은 기타 변경이 필요할 수 있습니다. 자세한 내용은 개별 오류에 대한 설명서를 참조하세요.

터미널 오류

터미널 오류로 인해 모델 훈련이 중단됩니다. 터미널 훈련 오류에는 서비스 오류, 매니페스트 파일 오류, 매니페스트 콘텐츠 오류의 세 가지 범주가 있습니다.

콘솔에서 Amazon Rekognition Custom Labels는 프로젝트 페이지의 상태 메시지 열에 모델의 터미널 오류를 표시합니다.



AWSSDK를 사용하는 경우 DescribeProjectVersions의 응답을 확인하여 터미널 매니페스트 파일 오류 또는 터미널 매니페스트 콘텐츠 오류가 발생했는지 확인할 수 있습니다. 이 경우 `Status` 값은 `TRAINING_FAILED`이고 `StatusMessage` 필드에는 오류가 포함됩니다.

서비스 오류

Amazon Rekognition에서 서비스 문제가 발생하여 훈련을 계속할 수 없을 때 터미널 서비스 오류가 발생합니다. Amazon Rekognition Custom Labels가 의존하는 다른 서비스의 장애를 예로 들 수 있습니다. Amazon Rekognition에서 서비스 문제가 발생함에 따라 Amazon Rekognition Custom Labels가 콘솔에서 서비스 오류를 보고합니다. AWS SDK를 사용하는 경우 CreateProjectVersion 및 DescribeProjectVersions에서 훈련 중에 발생하는 서비스 오류가 `InternalServerError` 예외로 발생합니다.

서비스 오류가 발생하는 경우 모델 훈련을 다시 시도하세요. 훈련이 계속 실패하는 경우, AWS Support에 연락하여 서비스 오류와 함께 보고된 오류 정보를 제공하세요.

터미널 매니페스트 파일 오류

매니페스트 파일 오류는 훈련 및 테스트 데이터 세트에서 파일 수준 또는 여러 파일에서 발생하는 터미널 오류입니다. 매니페스트 파일 오류는 훈련 및 테스트 데이터 세트의 내용이 검증되기 전에 감지됩니다. 매니페스트 파일 오류로 인해 비터미널 검증 오류가 보고되지 않습니다. 예를 들어, 빈 훈련 매니페스트 파일은 매니페스트 파일이 비어 있습니다 오류가 발생합니다. 파일이 비어 있으므로 비터미널 JSON 라인 검증 오류는 보고될 수 없습니다. 매니페스트 요약도 생성되지 않습니다.

모델을 훈련하려면 먼저 매니페스트 파일 오류를 수정해야 합니다.

다음은 매니페스트 파일 오류 목록입니다.

- 매니페스트 파일 확장명 또는 콘텐츠가 유효하지 않습니다.

- 매니페스트 파일이 비어 있습니다.
- 매니페스트 파일 크기가 지원되는 최대 크기를 초과합니다.
- 출력 S3 버킷에 쓸 수 없습니다.
- S3 버킷 권한이 올바르지 않습니다.

터미널 매니페스트 콘텐츠 오류

매니페스트 콘텐츠 오류는 매니페스트 내의 콘텐츠와 관련된 터미널 오류입니다. 예를 들어 매니페스트 파일에 레이블당 레이블이 지정된 이미지가 충분하지 않아 자동 분할을 수행할 수 없습니다라는 오류가 발생하는 경우, 훈련 데이터 세트에 테스트 데이터 세트를 생성할 만큼 레이블이 지정된 이미지가 충분하지 않으므로 훈련을 완료할 수 없습니다.

오류는 콘솔과 `DescribeProjectVersions` 응답 양식에 보고될 뿐 아니라 다른 터미널 매니페스트 콘텐츠 오류와 함께 매니페스트 요약에도 보고됩니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요.

비터미널 JSON 라인 오류는 별도의 훈련 및 테스트 검증 결과 매니페스트에도 보고됩니다. Amazon Rekognition Custom Labels에서 발견된 비터미널 JSON 라인 오류는 훈련을 중단시키는 매니페스트 콘텐츠 오류와 반드시 관련이 있는 것은 아닙니다. 자세한 내용은 훈련 및 테스트 검증 결과 매니페스트의 이해 섹션을 참조하세요.

모델을 훈련하려면 먼저 매니페스트 콘텐츠 오류를 수정해야 합니다.

다음은 매니페스트 콘텐츠 오류에 대한 오류 메시지입니다.

- 매니페스트 파일에 잘못된 행이 너무 많습니다.
- 매니페스트 파일에는 여러 S3 버킷의 이미지가 포함되어 있습니다.
- 이미지 S3 버킷의 소유자 ID가 잘못되었습니다.
- 매니페스트 파일에 레이블당 레이블이 지정된 이미지가 부족하여 자동 분할을 수행할 수 없습니다.
- 매니페스트 파일에 레이블이 너무 적습니다.
- 매니페스트 파일에 레이블이 너무 많습니다.
- 훈련 매니페스트 파일과 테스트 매니페스트 파일 간에 겹치는 레이블이 {}% 미만입니다.
- 매니페스트 파일에 사용 가능한 레이블이 너무 적습니다.
- 훈련 매니페스트 파일과 테스트 매니페스트 파일 간에 겹치는 사용 가능한 레이블이 {}% 미만입니다.

- S3 버킷에서 이미지를 복사하지 못했습니다.

비터미널 JSON 라인 검증 오류

JSON 라인 검증 오류는 Amazon Rekognition Custom Labels가 모델 훈련을 중단할 필요가 없는 비터미널 오류입니다.

JSON 라인 검증 오류는 콘솔에 표시되지 않습니다.

훈련 및 테스트 데이터 세트에서 JSON 라인은 하나의 이미지에 대한 훈련 또는 테스트 정보를 나타냅니다. 잘못된 이미지와 같은 JSON 라인의 검증 오류는 훈련 및 테스트 검증 매니페스트에 보고됩니다. Amazon Rekognition Custom Labels는 매니페스트에 있는 다른 유효한 JSON 라인을 사용하여 훈련을 완료합니다. 자세한 내용은 훈련 및 테스트 검증 결과 매니페스트의 이해 섹션을 참조하세요. 검증 규칙에 대한 자세한 내용은 매니페스트 파일의 검증 규칙 항목을 참조하세요.

> **Note**
>
> JSON 라인 오류가 너무 많으면 훈련이 실패합니다.

향후 오류를 유발하거나 모델 훈련에 영향을 미칠 수 있으므로 비터미널 JSON 라인 오류도 수정하는 것이 좋습니다.

Amazon Rekognition Custom Labels는 다음과 같은 비터미널 JSON 라인 검증 오류를 생성할 수 있습니다.

- source-ref 키가 누락되었습니다.
- source-ref 값의 형식이 잘못되었습니다.
- 레이블 속성을 찾을 수 없습니다.
- 레이블 속성 {}의 형식이 잘못되었습니다.
- 레이블 attributemetadata의 형식이 잘못되었습니다.
- 유효한 레이블 속성을 찾을 수 없습니다.
- 하나 이상의 경계 상자에 신뢰도 값이 누락되었습니다.
- 클래스 맵에서 하나 이상의 클래스 ID가 누락되었습니다.
- JSON 라인의 형식이 잘못되었습니다.

- 이미지가 유효하지 않습니다. S3 경로 및/또는 이미지 속성을 확인하세요.
- 경계 상자에 오프 프레임 값이 있습니다.
- 경계 상자의 높이와 너비가 너무 작습니다.
- 경계 상자가 허용된 최대 값보다 많습니다.
- 유효한 주석을 찾을 수 없습니다.

매니페스트 요약 이해

매니페스트 요약에는 다음 정보가 포함되어 있습니다.

- 검증 중에 발생한 터미널 매니페스트 콘텐츠 오류에 관한 오류 정보
- 훈련 및 테스트 데이터 세트에 있는 비터미널 JSON 라인 검증 오류에 대한 오류 위치 정보
- 훈련 및 테스트 데이터 세트에서 발견된 잘못된 JSON 라인의 총 개수와 같은 오류 통계

터미널 매니페스트 파일 오류 항목이 없는 경우 훈련 중에 매니페스트 요약이 생성됩니다. 매니페스트 요약 파일(manifest_summary.json)의 위치를 가져오려면 검증 결과 가져오기 항목을 참조하세요.

> **Note**
>
> 서비스 오류 및 매니페스트 파일 오류는 매니페스트 요약에 보고되지 않습니다. 자세한 내용은 터미널 오류 섹션을 참조하세요.

특정 매니페스트 콘텐츠 오류에 대한 자세한 내용은 터미널 매니페스트 콘텐츠 오류 항목을 참조하세요.

매니페스트 요약 파일 형식

매니페스트 파일은 `statistics` 및 `errors`의 2개 항목으로 구성되어 있습니다.

통계

`statistics`에는 훈련 및 테스트 데이터 세트의 오류에 대한 정보가 들어 있습니다.

- `training`: 훈련 데이터 세트에서 발견된 통계 및 오류
- `testing`: 테스트 데이터 세트에서 발견된 통계 및 오류

errors 배열의 객체에는 매니페스트 콘텐츠 오류에 대한 오류 코드와 메시지가 포함됩니다.

error_line_indices 배열에는 오류가 있는 훈련 또는 테스트 매니페스트의 각 JSON 행에 대한 줄 번호가 포함됩니다. 자세한 내용은 훈련 오류 수정 섹션을 참조하세요.

오류

훈련 데이터 세트와 테스트 데이터 세트 모두에 관련된 오류가 있습니다. 예를 들어, 훈련 및 테스트 데이터 세트와 겹치는 사용 가능한 레이블이 충분하지 않을 때 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP 오류가 발생합니다.

```
{
    "statistics": {
        "training":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number,   # Total number json lines (images) in the
  training manifest.
                "valid_json_lines": Number,   # Total number of JSON Lines (images)
 that can be used for training.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
 They are not used for training.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
 have no annotations. The aren't used for training and aren't counted as invalid.
                "error_json_line_indices": List[int], # Contains a list of line numbers
 for JSON line errors in the training dataset.
                "errors": [
                    {
                        "code": String, # Error code for a training manifest content
 error.
                        "message": String # Description for a training manifest content
 error.
                    }
                ]
            },
        "testing":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number, # Total number json lines (images) in the
 manifest.
```

```
                "valid_json_lines": Number,  # Total number of JSON Lines (images) that
 can be used for testing.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
 They are not used for testing.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
 have no annotations. They aren't used for testing and aren't counted as invalid.
                "error_json_line_indices": List[int], # contains a list of error record
 line numbers in testing dataset.
                "errors": [
                    {
                        "code": String,   # # Error code for a testing manifest content
 error.
                        "message": String # Description for a testing manifest content
 error.
                    }
                ]
            }
    },
    "errors": [
        {
            "code": String, # # Error code for errors that span the training and
 testing datasets.
            "message": String # Description of the error.
        }
    ]
}
```

예제 매니페스트 요약

다음 예제는 터미널 매니페스트 콘텐츠 오류(ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST)를 보여주는 부분 매니페스트 요약입니다. `error_json_line_indices` 배열에는 해당하는 훈련 또는 테스트 검증 매니페스트에 있는 비터미널 JSON 라인 오류의 줄 번호가 포함됩니다.

```
{
    "errors": [],
    "statistics": {
        "training": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 301,
            "valid_json_lines": 146,
            "invalid_json_lines": 155,
            "ignored_json_lines": 0,
            "errors": [
```

```
                {
                    "code": "ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST",
                    "message": "The manifest file contains too many invalid rows."
                }
            ],
            "error_json_line_indices": [
                15,
                16,
                17,
                22,
                23,
                24,
                 .
                 .
                 .
                 .
                300
            ]
        },
        "testing": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 15,
            "valid_json_lines": 13,
            "invalid_json_lines": 2,
            "ignored_json_lines": 0,
            "errors": [],
            "error_json_line_indices": [
                13,
                15
            ]
        }
    }
}
```

훈련 및 테스트 검증 결과 매니페스트의 이해

Amazon Rekognition Custom Labels는 훈련 중에 비터미널 JSON 라인 오류를 보관하기 위한 검증 결과 매니페스트를 생성합니다. 검증 결과 매니페스트는 오류 정보가 추가된 훈련 및 테스트 데이터 세트의 사본입니다. 훈련이 완료된 후 검증 매니페스트에 액세스할 수 있습니다. 자세한 내용은 검증 결과 가져오기 섹션을 참조하세요. Amazon Rekognition Custom Labels는 오류 위치 및 JSON 라인 오류 수

와 같은 JSON 라인 오류에 대한 개요 정보가 포함된 매니페스트 요약도 생성합니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요.

> **Note**
>
> 검증 결과(훈련 및 테스트 검증 결과 매니페스트와 매니페스트 요약)는 터미널 매니페스트 파일 오류 항목이 없는 경우에만 생성됩니다.

매니페스트에는 데이터 세트의 각 이미지에 대한 JSON 라인이 포함됩니다. 검증 결과 매니페스트 내에서 오류가 발생한 JSON 라인에 JSON 라인 오류 정보가 추가됩니다.

JSON 라인 오류는 하나의 이미지와 관련된 비터미널 오류입니다. 비터미널 검증 오류로 인해 JSON 라인 전체 또는 일부가 무효화될 수 있습니다. 예를 들어 JSON 라인에서 참조된 이미지가 PNG 또는 JPG 형식이 아닌 경우 ERROR_INVALID_IMAGE 오류가 발생하고 전체 JSON 라인이 훈련에서 제외됩니다. 다른 유효한 JSON 라인을 사용하여 훈련이 계속됩니다.

JSON 라인 내에서 오류가 발생하면 JSON 라인을 여전히 훈련에 사용할 수 있다는 의미일 수 있습니다. 예를 들어 레이블과 관련된 네 개의 경계 상자 중 하나의 왼쪽 값이 음수인 경우에도 모델은 다른 유효한 경계 상자를 사용하여 훈련됩니다. 잘못된 경계 상자(ERROR_INVALID_BOUNDING_BOX)에 대한 JSON 라인 오류 정보가 반환됩니다. 이 예제에서는 오류가 발생한 `annotation` 객체에 오류 정보가 추가됩니다.

WARNING_NO_ANNOTATIONS 항목과 같은 경고 오류는 훈련에 사용되지 않으며 매니페스트 요약에서 무시된 JSON 라인(`ignored_json_lines`)으로 간주됩니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요. 또한 무시된 JSON 라인은 훈련 및 테스트의 20% 오류 임계값에 포함되지 않습니다.

특정 비터미널 데이터 검증 오류에 대한 자세한 내용은 비터미널 JSON 라인 검증 오류 항목을 참조하세요.

> **Note**
>
> 데이터 검증 오류가 너무 많으면 훈련이 중단되고 매니페스트 요약에 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST 터미널 오류가 보고됩니다.

JSON Line 오류 수정에 대한 자세한 내용은 훈련 오류 수정 항목을 참조하세요.

JSON 라인 오류 형식

Amazon Rekognition Custom Labels는 비터미널 검증 오류 정보를 이미지 수준 및 객체 위치 파악 형식의 JSON 라인에 추가합니다. 자세한 내용은 the section called “매니페스트 파일 생성” 섹션을 참조하세요.

이미지 수준 오류

다음 예제는 이미지 수준 JSON 라인의 `Error` 배열을 보여줍니다. 두 세트의 오류가 있습니다. 레이블 속성 메타데이터(이 예제에서는 스포츠 메타데이터)와 관련된 오류와 이미지와 관련된 오류가 있습니다. 오류에는 오류 코드(코드), 오류 메시지(메시지)가 포함됩니다. 자세한 내용은 매니페스트 파일의 이미지 수준 레이블 섹션을 참조하세요.

```
{
    "source-ref": String,
    "sport": Number,
    "sport-metadata": {
        "class-name": String,
        "confidence": Float,
        "type": String,
        "job-name": String,
        "human-annotated": String,
        "creation-date": String,
        "errors": [
            {
                "code": String, # error codes for label
                "message": String # Description and additional contextual details of
 the error
            }
        ]
    },
    "errors": [
        {
            "code": String, # error codes for image
            "message": String # Description and additional contextual details of the
 error
        }
    ]
}
```

객체 위치 파악 오류

다음 예제는 객체 위치 파악 JSON 라인의 오류 배열을 보여줍니다. JSON 라인은 다음 JSON 라인 항목의 필드에 대한 `Errors` 배열 정보를 포함합니다. 각 `Error` 객체에는 오류 코드와 오류 메시지가 들어 있습니다.

- 레이블 속성: 레이블 속성 필드의 오류입니다. 예제의 `bounding-box` 항목을 참조하세요.
- 주석: 주석 오류(경계 상자)는 레이블 속성 내 `annotations` 배열에 저장됩니다.
- 레이블 attribute-metadata: 레이블 속성 필드의 오류입니다. 예제의 `bounding-box-metadata` 항목을 참조하세요.
- 이미지: 레이블 속성, 주석 및 레이블 속성 메타데이터 필드와 관련이 없는 오류입니다.

자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

```
{
    "source-ref": String,
    "bounding-box": {
        "image_size": [
            {
                "width": Int,
                "height": Int,
                "depth":Int,
            }
        ],
        "annotations": [
            {
                "class_id": Int,
                "left": Int,
                "top": Int,
                "width": Int,
                "height": Int,
                "errors": [   # annotation field errors
                    {
                        "code": String, # annotation field error code
                        "message": String # Description and additional contextual
 details of the error
                    }
                ]
            }
        ],
        "errors": [ #label attribute field errors
```

```
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
the error
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": Float
            }
        ],
        "class-map": {
            String: String
        },
        "type": String,
        "human-annotated": String,
        "creation-date": String,
        "job-name": String,
        "errors": [  #metadata field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
the error
            }
        ]
    },
   "errors": [  # image errors
        {
            "code": String, # error code
            "message": String # Description and additional contextual details of the
error
        }
    ]
}
```

예제 JSON 라인 오류

다음 객체 위치 파악 JSON 라인(가독성을 위해 형식 지정)에는 ERROR_BOUNDING_BOX_TOO_SMALL 오류가 표시됩니다. 이 예제에서 경계 상자 크기(높이 및 너비)는 1 x 1보다 크지 않습니다.

```
{
    "source-ref": "s3://bucket/Manifests/images/199940-1791.jpg",
    "bounding-box": {
        "image_size": [
            {
                "width": 3000,
                "height": 3000,
                "depth": 3
            }
        ],
        "annotations": [
            {
                "class_id": 1,
                "top": 0,
                "left": 0,
                "width": 1,
                "height": 1,
                "errors": [
                    {
                        "code": "ERROR_BOUNDING_BOX_TOO_SMALL",
                        "message": "The height and width of the bounding box is too
 small."
                    }
                ]
            },
            {
                "class_id": 0,
                "top": 65,
                "left": 86,
                "width": 220,
                "height": 334
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
```

```
        "class-map": {
            "0": "Echo",
            "1": "Echo Dot"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2019-11-20T02:57:28.288286",
        "job-name": "my job"
    }
}
```

검증 결과 가져오기

검증 결과에는 터미널 매니페스트 콘텐츠 오류 및 비터미널 JSON 라인 검증 오류에 대한 오류 정보가 포함됩니다. 세 개의 검증 결과 파일이 있습니다.

- training_manifest_with_validation.json: JSON 라인 오류 정보가 추가된 훈련 데이터 세트 매니페스트 파일의 복사본입니다.
- testing_manifest_with_validation.json: JSON 라인 오류 정보가 추가된 테스트 데이터세트 매니페스트 파일의 복사본입니다.
- manifest_summary.json: 훈련 및 테스트 데이터 세트에서 발견된 매니페스트 콘텐츠 오류 및 JSON 라인 오류에 대한 요약입니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요.

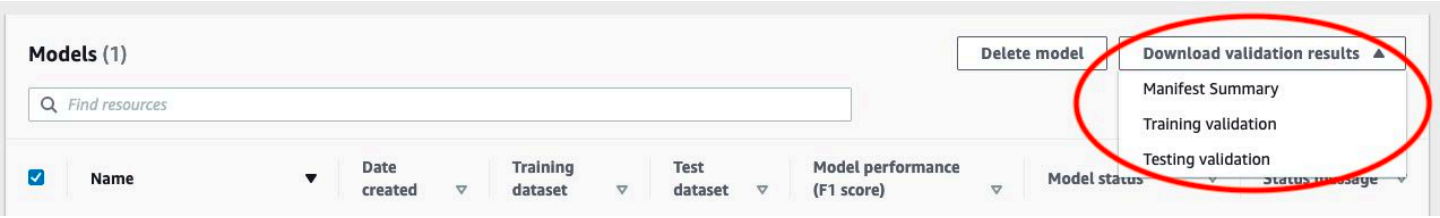
훈련 및 테스트 검증 매니페스트의 내용에 대한 자세한 내용은 실패한 모델 훈련 디버깅 항목을 참조하세요.

> **Note**
>
> - 검증 결과는 훈련 중에 생성된 터미널 매니페스트 파일 오류 항목이 없는 경우에만 생성됩니다.
> - 훈련 및 테스트 매니페스트가 검증된 후 서비스 오류가 발생하는 경우 검증 결과는 생성되지만 DescribeProjectVersions의 응답에는 검증 결과 파일 위치가 포함되지 않습니다.

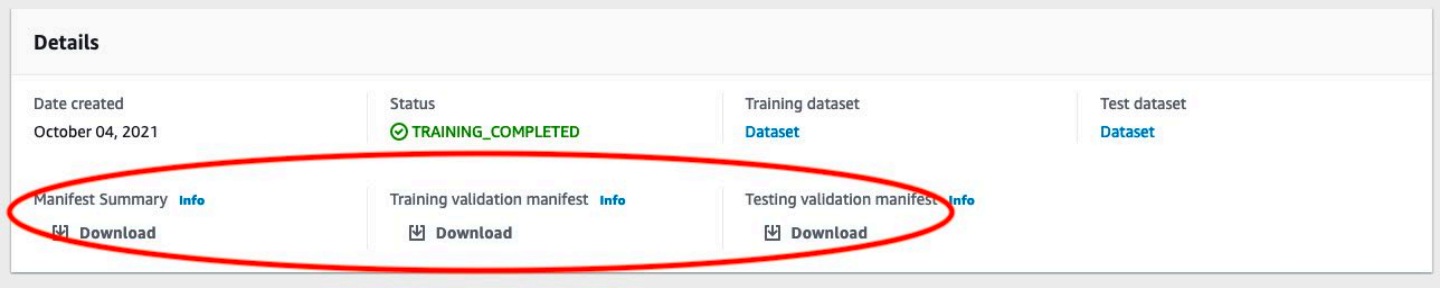
훈련이 완료되거나 실패한 후에는 Amazon Rekognition Custom Labels 콘솔을 사용하여 검증 결과를 다운로드하거나 DescribeProjectVersions API를 호출하여 Amazon S3 버킷 위치를 가져올 수 있습니다.

검증 결과 가져오기(콘솔)

콘솔을 사용하여 모델을 훈련하는 경우 다음 다이어그램과 같이 프로젝트의 모델 목록에서 검증 결과를 다운로드할 수 있습니다.



모델의 세부 정보 페이지에서 검증 결과 다운로드에 액세스할 수도 있습니다.



자세한 내용은 모델 훈련(콘솔) 섹션을 참조하세요.

검증 결과 가져오기(SDK)

모델 훈련이 완료되면 Amazon Rekognition Custom Labels는 훈련 중에 지정된 Amazon S3 버킷에 검증 결과를 저장합니다. 훈련이 완료된 후 DescribeProjectVersions API를 호출하여 S3 버킷 위치를 가져올 수 있습니다. 모델을 훈련하려면 모델 훈련(SDK) 항목을 참조하세요.

훈련 데이터 세트(TrainingDataResult)와 테스트 데이터 세트(TestingDataResult)에 대해 ValidationData 객체가 반환됩니다. 매니페스트 요약이 `ManifestSummary`에 반환됩니다.

Amazon S3 버킷 위치를 확인한 후 검증 결과를 다운로드할 수 있습니다. 자세한 내용은 S3 버킷에서 객체를 다운로드하려면 어떻게 해야 합니까?를 참조하세요. 또한 GetObject 작업을 사용할 수도 있습니다.

검증 데이터를 가져오려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제를 사용하여 검증 결과의 위치를 가져옵니다.

Python

project_arn 항목을 모델이 포함된 프로젝트의 Amazon 리소스 이름(ARN)으로 바꿉니다. 자세한 내용은 Amazon Rekognition Custom Labels 프로젝트 관리 섹션을 참조하세요. version_name 항목을 모델 버전의 이름으로 바꿉니다. 자세한 내용은 모델 훈련(SDK) 섹션을 참조하세요.

```
import boto3
import io
from io import BytesIO
import sys
import json


def describe_model(project_arn, version_name):

    client=boto3.client('rekognition')

    response=client.describe_project_versions(ProjectArn=project_arn,
        VersionNames=[version_name])

    for model in response['ProjectVersionDescriptions']:
        print(json.dumps(model,indent=4,default=str))

def main():

    project_arn='project_arn'
    version_name='version_name'

    describe_model(project_arn, version_name)

if __name__ == "__main__":
    main()
```

3. 프로그램 출력에서 TestingDataResult 및 TrainingDataResult 객체 내의 Validation 필드를 기록해 둡니다. 매니페스트 요약은 ManifestSummary에 있습니다.

훈련 오류 수정

매니페스트 요약을 사용하여 훈련 중에 발생한 터미널 매니페스트 콘텐츠 오류 및 비터미널 JSON 라인 검증 오류 항목을 식별할 수 있습니다. 매니페스트 콘텐츠 오류는 반드시 수정해야 합니다. 비터미

널 JSON 라인 오류도 수정하는 것이 좋습니다. 구체적인 오류와 해결책에 대해서는 비터미널 JSON 라인 검증 오류 및 터미널 매니페스트 콘텐츠 오류 항목을 참조하세요.

훈련에 사용되는 훈련 또는 테스트 데이터 세트를 수정할 수 있습니다. 또는 훈련 및 테스트 검증 매니페스트 파일에서 필요한 부분을 수정하고 이를 사용하여 모델을 훈련할 수 있습니다.

수정한 후에는 업데이트된 매니페스트를 가져와서 모델을 재훈련해야 합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

다음 절차는 매니페스트 요약을 사용하여 터미널 매니페스트 콘텐츠 오류를 수정하는 방법을 보여줍니다. 이 절차는는 훈련 및 테스트 검증 매니페스트에서 JSON 라인 오류를 찾아 수정하는 방법도 보여줍니다.

Amazon Rekognition Custom Labels 훈련 오류를 수정하려면

1. 검증 결과 파일을 다운로드하세요. 파일 이름은 training_manifest_with_validation.json, testing_manifest_with_validation.json, manifest_summary.json입니다. 자세한 내용은 검증 결과 가져오기 섹션을 참조하세요.
2. 매니페스트 요약 파일(manifest_summary.json)을 엽니다.
3. 매니페스트 요약의 모든 오류를 수정합니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요.
4. 매니페스트 요약에서 `training`의 `error_line_indices` 배열을 반복하여 해당 JSON 라인 번호에서 `training_manifest_with_validation.json`의 오류를 수정하세요. 자세한 내용은 the section called “훈련 및 테스트 검증 결과 매니페스트의 이해” 섹션을 참조하세요.
5. `testing`의 `error_line_indices` 배열을 반복하여 해당 JSON 라인 번호에서 `testing_manifest_with_validation.json`의 오류를 수정하세요.
6. 검증 매니페스트 파일을 훈련 및 테스트 데이터 세트로 사용하여 모델을 다시 훈련합니다. 자세한 내용은 the section called “모델 훈련” 섹션을 참조하세요.

AWS SDK를 사용하고 훈련 또는 테스트 검증 데이터 매니페스트 파일의 오류를 수정하기로 선택한 경우, TrainingData 및 TestingData 입력 파라미터의 검증 데이터 매니페스트 파일 위치를 사용하여 CreateProjectVersion을 만드세요. 자세한 내용은 모델 훈련(SDK) 섹션을 참조하세요.

JSON 라인 오류 우선순위

다음 JSON 라인 오류가 먼저 감지됩니다. 이러한 오류가 발생하면 JSON 라인 오류 검증이 중지됩니다. 다른 JSON 라인 오류를 수정하려면 먼저 이러한 오류를 수정해야 합니다.

- MISSING_SOURCE_REF
- ERROR_INVALID_SOURCE_REF_FORMAT
- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_INVALID_JSON_LINE

터미널 매니페스트 파일 오류

이 주제는 터미널 매니페스트 파일 오류에 대해 설명합니다. 매니페스트 파일 오류에는 관련 오류 코드가 없습니다. 터미널 매니페스트 파일 오류가 발생하면 검증 결과 매니페스트가 생성되지 않습니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요. 터미널 매니페스트 오류로 인해 비터미널 JSON 라인 검증 오류 보고가 불가능합니다.

매니페스트 파일 확장명 또는 콘텐츠가 유효하지 않습니다.

훈련 또는 테스트 매니페스트 파일에 파일 확장자가 없거나 해당 내용이 유효하지 않습니다.

매니페스트 파일 확장명 또는 콘텐츠가 유효하지 않습니다 오류를 수정하려면

- 훈련 매니페스트 파일과 테스트 매니페스트 파일 모두에서 다음과 같은 가능한 원인을 확인하세요.
  - 매니페스트 파일에 파일 확장명이 없습니다. 일반적으로 파일 확장자는 `.manifest`입니다.
  - 매니페스트 파일의 Amazon S3 버킷 또는 키를 찾을 수 없습니다.

매니페스트 파일이 비어 있습니다.

훈련에 사용되는 훈련 또는 테스트 매니페스트 파일이 존재하지만 비어 있습니다. 매니페스트 파일에는 훈련 및 테스트에 사용하는 각 이미지에 대한 JSON 라인이 필요합니다.

매니페스트 파일이 비어 있습니다 오류를 수정하려면

1. 훈련 매니페스트와 테스트 매니페스트 중 어느 것이 비어 있는지 확인하세요.

2. 빈 매니페스트 파일에 JSON 라인을 추가합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요. 또는 콘솔을 사용하여 새 데이터 세트를 생성합니다. 자세한 내용은 the section called "이미지가 포함된 데이터 세트 생성" 섹션을 참조하세요.

매니페스트 파일 크기가 지원되는 최대 크기를 초과합니다.

훈련 또는 테스트 매니페스트 파일 크기(바이트)가 너무 큽니다. 자세한 내용은 Amazon Rekognition Custom Labels 지침 및 할당량 섹션을 참조하세요. 매니페스트 파일은 최대 JSON 라인 수보다 적으면서도 최대 파일 크기를 초과할 수 있습니다.

Amazon Rekognition Custom Labels 콘솔로는 매니페스트 파일 크기가 지원되는 최대 크기를 초과합니다 오류를 수정할 수 없습니다.

매니페스트 파일 크기가 지원되는 최대 크기를 초과합니다 오류를 수정하려면

1. 훈련 및 테스트 매니페스트 중에 어떤 것이 최대 파일 크기를 초과하는지 확인하세요.
2. 매니페스트 파일에서 너무 큰 JSON 라인 수를 줄이세요. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

S3 버킷 권한이 올바르지 않습니다.

Amazon Rekognition Custom Labels가 훈련 및 테스트 매니페스트 파일이 들어 있는 하나 이상의 버킷에 대한 권한이 없습니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

S3 버킷 권한이 올바르지 않습니다 오류를 수정하려면

- 훈련 및 테스트 매니페스트가 포함된 버킷의 권한을 확인하세요. 자세한 내용은 2단계: Amazon Rekognition Custom Labels 콘솔 권한 설정 섹션을 참조하세요.

출력 S3 버킷에 쓸 수 없습니다.

서비스가 훈련 출력 파일을 생성할 수 없습니다.

출력 S3 버킷에 쓸 수 없습니다 오류를 수정하려면

- CreateProjectVersion을 생성하기 위한 OutputConfig 입력 파라미터의 Amazon S3 버킷 정보가 정확한지 확인하세요.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

터미널 매니페스트 콘텐츠 오류

이 주제는 매니페스트 요약에 보고된 터미널 매니페스트 콘텐츠 오류 항목을 설명합니다. 매니페스트 요약에는 탐지된 각 오류에 대한 오류 코드와 메시지가 포함됩니다. 자세한 내용은 매니페스트 요약 이해 섹션을 참조하세요. 터미널 매니페스트 콘텐츠 오류는 비터미널 JSON 라인 검증 오류 보고를 중단하지 않습니다.

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

오류 메시지

매니페스트 파일에 잘못된 행이 너무 많습니다.

추가 정보

잘못된 콘텐츠가 포함된 JSON 라인이 너무 많으면 `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` 오류가 발생합니다.

Amazon Rekognition Custom Labels 로는 `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` 오류를 수정할 수 없습니다.

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST를 수정하려면

1. 매니페스트에서 JSON 라인 오류가 있는지 확인하세요. 자세한 내용은 훈련 및 테스트 검증 결과 매니페스트의 이해 섹션을 참조하세요.
2. 오류가 있는 JSON 라인을 수정하세요. 자세한 내용은 비터미널 JSON 라인 검증 오류 섹션을 참조하세요.

ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS

오류 메시지

매니페스트 파일에는 여러 S3 버킷의 이미지가 포함되어 있습니다.

추가 정보

매니페스트는 하나의 버킷에 저장된 이미지만 참조할 수 있습니다. 각 JSON 라인은 이미지 위치의 Amazon S3 위치를 `source-ref`의 값으로 저장합니다. 다음 예제에서 버킷 이름은 my-bucket입니다.

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

`ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS` 오류를 수정하려면

- 모든 이미지가 동일한 Amazon S3 버킷에 있고 모든 JSON 라인의 `source-ref` 값이 이미지가 저장된 버킷을 참조하는지 확인하세요. 또는 선호하는 Amazon S3 버킷을 선택하고 선호하는 버킷을 참조하지 않는 `source-ref` JSON 라인을 제거하세요.

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

오류 메시지

이미지 S3 버킷에 대한 권한이 유효하지 않습니다.

추가 정보

이미지가 포함된 Amazon S3 버킷에 대한 권한이 잘못되었습니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

`ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET` 오류를 수정하려면

- 이미지가 들어 있는 버킷의 권한을 확인하세요. 이미지의 `source-ref` 값에는 버킷 위치가 포함됩니다.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

오류 메시지

이미지 S3 버킷의 소유자 ID가 잘못되었습니다.

추가 정보

훈련 또는 테스트 이미지가 포함된 버킷의 소유자가 훈련 또는 테스트 매니페스트가 포함된 버킷의 소유자와 다릅니다. 다음 명령을 사용하여 버킷의 소유자를 찾을 수 있습니다.

```
aws s3api get-bucket-acl --bucket bucket name
```

OWNER 및 ID 항목이 이미지와 매니페스트 파일을 저장하는 버킷과 일치해야 합니다.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER 오류를 수정하려면

1. 훈련, 테스트, 출력, 이미지 버킷의 원하는 소유자를 선택합니다. 소유자에게는 Amazon Rekognition Custom Labels를 사용할 권한이 있어야 합니다.
2. 원하는 소유자가 현재 소유하지 않은 각 버킷에 대해 원하는 소유자가 소유한 새 Amazon S3 버킷을 생성합니다.
3. 이전 버킷 콘텐츠를 새 버킷에 복사합니다. 자세한 내용은 Amazon S3 버킷 간에 객체를 복사하려면 어떻게 해야 합니까?를 참조하세요.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

오류 메시지

매니페스트 파일에 레이블당 레이블이 지정된 이미지가 부족하여 자동 분할을 수행할 수 없습니다.

추가 정보

모델 훈련 중에 훈련 데이터 세트의 이미지 중 20%를 사용하여 테스트 데이터 세트를 만들 수 있습니다. 이미지가 충분하지 않아 수용 가능한 테스트 데이터 세트를 만들 수 없을 때 ERROR_INFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT이 발생합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT을 수정하려면

- 훈련 데이터 세트에 레이블이 지정된 이미지를 더 추가하세요. Amazon Rekognition Custom Labels 콘솔에서 훈련 데이터세트에 이미지를 추가하거나 훈련 매니페스트에 JSON 라인을 추가하여 이미지를 추가할 수 있습니다. 자세한 내용은 데이터 세트 관리 섹션을 참조하세요.

ERROR_MANIFEST_TOO_FEW_LABELS

오류 메시지

매니페스트 파일에 레이블이 너무 적습니다.

추가 정보

훈련 및 테스트 데이터 세트에는 필요한 최소 개수의 레이블이 있습니다. 최솟값은 데이터 세트가 모델을 훈련/테스트하여 이미지 수준 레이블(분류)을 탐지하는지 또는 모델이 객체 위치를 감지하는지 여부에 따라 달라집니다. 훈련 데이터 세트를 분할하여 테스트 데이터 세트를 만드는 경우, 훈련 데이터 세트가 분할된 후 데이터 세트의 레이블 개수가 결정됩니다. 자세한 내용은 Amazon Rekognition Custom Labels 지침 및 할당량 섹션을 참조하세요.

ERROR_MANIFEST_TOO_FEW_LABELS 오류를 수정하려면(콘솔)

1. 데이터 세트에 새 레이블을 더 추가하세요. 자세한 내용은 레이블 관리 섹션을 참조하세요.
2. 데이터 세트의 이미지에 새 레이블을 추가합니다. 모델이 이미지 수준 레이블을 감지하는 경우 이미지에 이미지 수준 레이블 지정 항목을 참조하세요. 모델이 객체 위치를 감지하는 경우 the section called “경계 상자로 객체에 레이블 지정” 항목을 참조하세요.

ERROR_MANIFEST_TOO_FEW_LABELS 오류를 해결하려면(JSON 라인)

- 새 레이블이 붙은 새 이미지에 JSON 라인을 추가하세요. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요. 모델이 이미지 수준 레이블을 감지하는 경우 `class-name` 필드에 새 레이블 이름을 추가합니다. 예를 들어, 다음 이미지의 레이블은 Sunrise입니다.

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256",
        "type": "groundtruth/image-classification"
    }
}
```

모델이 객체 위치를 감지하는 경우 다음 예제와 같이 class-map에 새 레이블을 추가합니다.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

클래스 맵 테이블을 경계 상자 주석에 매핑해야 합니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

ERROR_MANIFEST_TOO_MANY_LABELS

오류 메시지

매니페스트 파일에 레이블이 너무 많습니다.

추가 정보

매니페스트(데이터 세트)의 고유 레이블 수가 허용된 한도를 초과했습니다. 훈련 데이터 세트를 분할하여 테스트 데이터 세트를 만드는 경우 분할 후 레이블 수가 결정됩니다.

ERROR_MANIFEST_TOO_MANY_LABELS 오류를 수정하려면(콘솔)

- 데이터 세트에서 레이블을 제거합니다. 자세한 내용은 레이블 관리 섹션을 참조하세요. 데이터 세트의 이미지와 경계 상자에서 레이블이 자동으로 제거됩니다.

ERROR_MANIFEST_TOO_MANY_LABELS 오류를 수정하려면(JSON 라인)

- 이미지 수준 JSON 라인이 있는 매니페스트: 이미지에 하나의 레이블이 있는 경우 원하는 레이블을 사용하는 이미지의 JSON 라인을 제거하세요. JSON 라인에 여러 레이블이 포함된 경우 원하는 레이블의 JSON 객체만 제거하세요. 자세한 내용은 이미지에 여러 이미지 수준 레이블 추가 섹션을 참조하세요.

  객체 위치가 있는 매니페스트 JSON 라인: 제거하려는 레이블의 경계 상자 및 관련 레이블 정보를 제거합니다. 원하는 레이블이 포함된 각 JSON 라인에 대해 이 작업을 수행하세요. `class-map` 배열과 `objects` 및 `annotations` 배열의 해당 객체를 레이블에서 제거해야 합니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

ERROR_INSUFFICIENT_LABEL_OVERLAP

오류 메시지

훈련 매니페스트 파일과 테스트 매니페스트 파일 간에 겹치는 레이블이 {}% 미만입니다.

추가 정보

테스트 데이터 세트 레이블 이름과 훈련 데이터 세트 레이블 이름 간의 중첩이 50% 미만입니다.

ERROR_INSUFFICIENT_LABEL_OVERLAP 오류를 수정하려면(콘솔)

- 훈련 데이터 세트에서 레이블을 제거하세요. 또는 테스트 데이터 세트에 더 일반적인 레이블을 추가할 수도 있습니다. 자세한 내용은 레이블 관리 섹션을 참조하세요. 데이터 세트의 이미지와 경계 상자에서 레이블이 자동으로 제거됩니다.

훈련 데이터 세트에서 레이블을 제거하여 ERROR_INSUFFICIENT_LABEL_OVERLAP 오류를 수정하려면(JSON 라인)

- 이미지 수준 JSON 라인이 있는 매니페스트: 이미지에 하나의 레이블이 있는 경우 원하는 레이블을 사용하는 이미지의 JSON 라인을 제거하세요. JSON 라인에 여러 레이블이 포함된 경우 원하는 레이블의 JSON 객체만 제거하세요. 자세한 내용은 이미지에 여러 이미지 수준 레이블 추가 섹션을 참조하세요. 제거하려는 레이블이 들어 있는 매니페스트의 각 JSON 라인에 대해 이 작업을 수행하세요.

  객체 위치가 있는 매니페스트 JSON 라인: 제거하려는 레이블의 경계 상자 및 관련 레이블 정보를 제거합니다. 원하는 레이블이 포함된 각 JSON 라인에 대해 이 작업을 수행하세요. `class-map` 배열과 `objects` 및 `annotations` 배열의 해당 객체를 레이블에서 제거해야 합니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

테스트 데이터 세트에 일반적인 레이블을 추가하여 ERROR_INSUFFICIENT_LABEL_OVERLAP 오류를 수정하려면(JSON 라인)

- 훈련 데이터 세트에 이미 레이블이 지정된 이미지가 포함된 JSON 라인을 테스트 데이터 세트에 추가합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS

오류 메시지

매니페스트 파일에 사용 가능한 레이블이 너무 적습니다.

추가 정보

훈련 매니페스트에는 이미지 수준 레이블 형식과 객체 위치 형식의 JSON 라인이 포함될 수 있습니다. Amazon Rekognition Custom Labels는 훈련 매니페스트에 있는 JSON 라인 유형에 따라 이미지 수준 레이블을 감지하는 모델 또는 객체 위치를 감지하는 모델을 생성합니다. Amazon Rekognition Custom Labels는 선택한 형식이 아닌 JSON 라인에 대해 유효한 JSON 레코드를 필터링합니다. 선택한 모델 유형 매니페스트의 레이블 수가 모델을 훈련하기에 충분하지 않은 경우 ERROR_MANIFEST_TOO_FEW_USABLE_LABELS가 발생합니다.

이미지 수준 레이블을 감지하는 모델을 훈련하려면 최소 1개의 레이블이 필요합니다. 위치를 파악하는 모델을 훈련하려면 최소 2개의 레이블이 필요합니다.

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS 오류를 수정하려면(콘솔)

1. 매니페스트 요약에서 use_case 필드를 확인하세요.
2. use_case의 값과 일치하는 사용 사례(이미지 수준 또는 객체 위치 파악)의 훈련 데이터 세트에 레이블을 더 추가하세요. 자세한 내용은 레이블 관리 섹션을 참조하세요. 데이터 세트의 이미지와 경계 상자에서 레이블이 자동으로 제거됩니다.

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS 오류를 수정하려면(JSON 라인)

1. 매니페스트 요약에서 use_case 필드를 확인하세요.
2. use_case의 값과 일치하는 사용 사례(이미지 수준 또는 객체 위치 파악)의 훈련 데이터 세트에 레이블을 더 추가하세요. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

오류 메시지

훈련 매니페스트 파일과 테스트 매니페스트 파일 간에 겹치는 사용 가능한 레이블이 {}% 미만입니다.

추가 정보

훈련 매니페스트에는 이미지 수준 레이블 형식과 객체 위치 형식의 JSON 라인이 포함될 수 있습니다. Amazon Rekognition Custom Labels는 훈련 매니페스트에 있는 형식에 따라 이미지 수준 레이블을 감지하는 모델 또는 객체 위치를 감지하는 모델을 생성합니다. Amazon Rekognition Custom Labels가 선택한 모델 형식이 아닌 JSON 라인에 대해 유효한 JSON 레코드를 사용

하지 않습니다. 사용되는 테스트 레이블과 훈련 레이블 간에 겹치는 부분이 50% 미만이면 ERROR_INFFICIENT_USABLE_LABEL_OVERLAP이 발생합니다.

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP 오류를 수정하려면(콘솔)

- 훈련 데이터 세트에서 레이블을 제거하세요. 또는 테스트 데이터 세트에 더 일반적인 레이블을 추가할 수도 있습니다. 자세한 내용은 레이블 관리 섹션을 참조하세요. 데이터 세트의 이미지와 경계 상자에서 레이블이 자동으로 제거됩니다.

훈련 데이터세트에서 레이블을 제거하여 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP을 수정하려면(JSON 라인)

- 이미지 수준 레이블을 감지하는 데 사용되는 데이터 세트: 이미지에 하나의 레이블이 있는 경우 원하는 레이블을 사용하는 이미지의 JSON 라인을 삭제하세요. JSON 라인에 여러 레이블이 포함된 경우 원하는 레이블의 JSON 객체만 제거하세요. 자세한 내용은 이미지에 여러 이미지 수준 레이블 추가 섹션을 참조하세요. 제거하려는 레이블이 들어 있는 매니페스트의 각 JSON 라인에 대해 이 작업을 수행하세요.

  객체 위치를 감지하는 데 사용되는 데이터 세트: 제거하려는 레이블의 경계 상자 및 관련 레이블 정보를 제거하세요. 원하는 레이블이 포함된 각 JSON 라인에 대해 이 작업을 수행하세요. `class-map` 배열과 `objects` 및 `annotations` 배열의 해당 객체를 레이블에서 제거해야 합니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

테스트 데이터세트에 공통 레이블을 추가하여 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP을 수정하려면(JSON 라인)

- 훈련 데이터 세트에 이미 레이블이 지정된 이미지가 포함된 JSON 라인을 테스트 데이터 세트에 추가합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_FAILED_IMAGES_S3_COPY

오류 메시지

S3 버킷에서 이미지를 복사하지 못했습니다.

추가 정보

서비스가 데이터 세트에 있는 이미지를 복사하지 못했습니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_FAILED_IMAGES_S3_COPY 오류를 수정하려면

1. 이미지의 권한을 확인하세요.
2. AWS KMS 항목을 사용하는 경우 버킷 정책을 확인하세요. 자세한 내용은 로 암호화된 파일을 복호화하고 있습니다. AWS Key Management Service 섹션을 참조하세요.

매니페스트 파일에 터미널 오류가 너무 많습니다.

터미널 콘텐츠 오류가 있는 JSON 라인이 너무 많습니다.

ERROR_TOO_MANY_RECORDS_IN_ERROR 오류를 수정하려면

- 터미널 콘텐츠 오류가 있는 JSON 라인(이미지) 수를 줄이세요. 자세한 내용은 터미널 매니페스트 콘텐츠 오류 섹션을 참조하세요.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

비터미널 JSON 라인 검증 오류

이 주제에는 Amazon Rekognition Custom Labels가 훈련 중에 보고한 비터미널 JSON 라인 검증 오류가 나열되어 있습니다. 오류는 훈련 및 테스트 검증 매니페스트에 보고됩니다. 자세한 내용은 훈련 및 테스트 검증 결과 매니페스트의 이해 섹션을 참조하세요. 훈련 또는 테스트 매니페스트 파일에서 JSON 라인을 업데이트하여 터미널이 아닌 JSON 라인 오류를 수정할 수 있습니다. 매니페스트에서 JSON 라인을 제거할 수도 있지만 이렇게 하면 모델 품질이 저하될 수 있습니다. 터미널이 아닌 검증 오류가 많으면 매니페스트 파일을 다시 만드는 것이 더 쉬울 수 있습니다. 검증 오류는 일반적으로 수동으로 만든 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요. 검증 오류 수정에 대한 자세한 내용은 훈련 오류 수정 항목을 참조하세요. Amazon Rekognition Custom Labels 콘솔을 사용하여 일부 오류를 수정할 수 있습니다.

ERROR_MISSING_SOURCE_REF

오류 메시지

source-ref 키가 누락되었습니다.

추가 정보

JSON 라인 source-ref 필드는 이미지의 Amazon S3 위치를 제공합니다. 이 오류는 source-ref 키가 누락되었거나 철자가 잘못되었을 때 발생합니다. 이 오류는 일반적으로 수동으로 만든 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_MISSING_SOURCE_REF 오류를 수정하려면

1. source-ref 키가 있고 철자가 올바른지 확인하세요. 완전한 source-ref 키와 값은 "source-ref": "s3://bucket/path/image" 항목과 비슷합니다.
2. JSON 라인의 source-ref 키를 업데이트하세요. 또는 매니페스트 파일에서 JSON 라인을 제거해도 됩니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INVALID_SOURCE_REF_FORMAT

오류 메시지

source-ref 값의 형식이 잘못되었습니다.

추가 정보

source-ref 키가 JSON 라인에 있지만 Amazon S3 경로의 스키마가 올바르지 않습니다. 예를 들어, 경로가 https://.... 대신 S3://....입니다. ERROR_INVALID_SOURCE_REF_FORMAT 오류는 일반적으로 수동으로 생성한 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_INVALID_SOURCE_REF_FORMAT 오류를 수정하려면

1. 스키마가 "source-ref": "s3://bucket/path/image" 항목인지 확인하세요. 예: "source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"
2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 ERROR_INVALID_SOURCE_REF_FORMAT 항목을 수정할 수 없습니다.

ERROR_NO_LABEL_ATTRIBUTES

오류 메시지

레이블 속성을 찾을 수 없습니다.

추가 정보

레이블 속성 또는 레이블 속성 -metadata 키 이름(또는 둘 다)이 잘못되었거나 누락되었습니다. 다음 예제에서는 bounding-box 또는 bounding-box-metadata 키(또는 둘 다)가 누락될 때마다 ERROR_NO_LABEL_ATTRIBUTES 항목이 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
```

```
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
}
```

ERROR_NO_LABEL_ATTRIBUTES 오류는 일반적으로 수동으로 만든 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_NO_LABEL_ATTRIBUTES 오류를 수정하려면

1. 레이블 속성 식별자와 레이블 속성 식별자 -metadata 키가 있고 키 이름의 철자가 올바른지 확인하세요.
2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 ERROR_NO_LABEL_ATTRIBUTES 항목을 수정할 수 없습니다.

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

오류 메시지

레이블 속성 {}의 형식이 잘못되었습니다.

추가 정보

레이블 속성 키의 스키마가 누락되었거나 유효하지 않습니다.
ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 오류는 일반적으로 수동으로 생성한 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 오류를 수정하려면

1. 레이블 속성 키의 JSON 라인 섹션이 올바른지 확인하세요. 다음 예제 객체 위치 예제에서는 image_size 및 annotations 객체가 정확해야 합니다. 레이블 속성 키는 bounding-box 이름이 지정되어 있습니다.

```
"bounding-box": {
```

```
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }, {
  "class_id": 0,
  "top": 65,
  "left": 86,
  "width": 220,
  "height": 334
 }]
},
```

2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

오류 메시지

레이블 속성 메타데이터의 형식이 잘못되었습니다.

추가 정보

레이블 속성 메타데이터 키의 스키마가 누락되었거나 유효하지 않습니다.
ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT 오류는 일반적으로 수동으로 생성한 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 오류를 수정하려면

1. 레이블 속성 메타데이터 키의 JSON 라인 스키마가 다음 예와 비슷한지 확인하세요. 레이블 속성 메타데이터 키는 `bounding-box-metadata` 이름이 지정되어 있습니다.

```
 "bounding-box-metadata": {
```

```
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
 "class-map": {
  "0": "Echo",
  "1": "Echo Dot"
 },
 "type": "groundtruth/object-detection",
 "human-annotated": "yes",
 "creation-date": "2018-10-18T22:18:13.527256",
 "job-name": "my job"
}
```

2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_NO_VALID_LABEL_ATTRIBUTES

오류 메시지

유효한 레이블 속성을 찾을 수 없습니다.

추가 정보

JSON 라인에서 유효한 레이블 속성을 찾을 수 없습니다. Amazon Rekognition Custom Labels는 레이블 속성과 레이블 속성 식별자를 모두 확인합니다. ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 오류는 일반적으로 수동으로 생성한 매니페스트 파일에서 발생합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

JSON 라인이 지원되는 SageMaker 매니페스트 형식이 아닌 경우 Amazon Rekognition Custom Labels는 JSON 라인을 유효하지 않은 것으로 표시하고 `ERROR_NO_VALID_LABEL_ATTRIBUTES` 오류가 보고됩니다. 현재 Amazon Rekognition Custom Labels는 분류 작업 및 경계 상자 형식을 지원합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

`ERROR_NO_VALID_LABEL_ATTRIBUTES` 오류를 수정하려면

1. 레이블 속성 키와 레이블 속성 메타데이터의 JSON이 올바른지 확인하세요.

2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다. 자세한 내용은 the section called "매니페스트 파일 생성" 섹션을 참조하세요.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

오류 메시지

하나 이상의 경계 상자에 신뢰도 값이 누락되었습니다.

추가 정보

하나 이상의 객체 위치 경계 상자에 대한 신뢰도 키가 누락되었습니다. 경계 상자의 신뢰도 키는 다음 예제와 같이 레이블 속성 메타데이터에 있습니다. ERROR_MISSING_BOUNDING_BOX_CONFIDENCE 오류는 일반적으로 수동으로 만든 매니페스트 파일에서 발생합니다. 자세한 내용은 the section called "매니페스트 파일의 객체 위치 파악" 섹션을 참조하세요.

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE 오류를 수정하려면

1. 레이블 속성의 `objects` 배열에 레이블 속성 `annotations` 배열에 있는 객체와 동일한 개수의 신뢰도 키가 포함되어 있는지 확인하세요.
2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_MISSING_CLASS_MAP_ID

오류 메시지

클래스 맵에서 하나 이상의 클래스 ID가 누락되었습니다.

추가 정보

주석(경계 상자) 객체 내의 class_id 항목이 레이블 속성 메타데이터 클래스 맵(class-map)에 일치하는 항목이 없습니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요. ERROR_MISSING_CLASS_MAP_ID 오류는 일반적으로 수동으로 만든 매니페스트 파일에서 발생합니다.

ERROR_MISSING_CLASS_MAP_ID 오류를 수정하려면

1. 다음 예와 같이 각 주석(경계 상자) 객체의 class_id 값에 대해 class-map 배열 안에 상응하는 값이 있는지 확인하세요. annotations 배열과 class_map 배열의 요소 개수는 같아야 합니다.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
```

```
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INVALID_JSON_LINE

오류 메시지

JSON 라인의 형식이 잘못되었습니다.

추가 정보

JSON 라인에서 예상치 못한 문자가 발견되었습니다. JSON 라인은 오류 정보만 포함된 새 JSON 라인으로 대체됩니다. ERROR_INVALID_JSON_LINE 오류는 일반적으로 수동으로 생성한 매니페스트 파일에서 발생합니다. 자세한 내용은 the section called “매니페스트 파일의 객체 위치 파악” 섹션을 참조하세요.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INVALID_JSON_LINE 오류를 수정하려면

1. 매니페스트 파일을 열고 ERROR_INVALID_JSON_LINE 오류가 발생한 JSON 라인으로 이동하세요.
2. JSON 라인에 유효하지 않은 문자가 없고 ; 또는 , 필수 문자가 누락되어 있지 않은지 확인하세요.
3. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

ERROR_INVALID_IMAGE

오류 메시지

이미지가 유효하지 않습니다. S3 경로 및/또는 이미지 속성을 확인하세요.

추가 정보

`source-ref` 항목이 참조한 파일이 유효한 이미지가 아닙니다. 가능한 원인으로는 이미지 종횡비, 이미지 크기, 이미지 형식 등이 있습니다.

자세한 내용은 지침 및 할당량 섹션을 참조하세요.

`ERROR_INVALID_IMAGE` 오류를 수정하려면

1. 다음을 확인하세요.
   - 이미지의 종횡비가 20:1 미만입니다.
   - 이미지 크기가 15MB를 초과합니다.
   - 이미지가 PNG 또는 JPEG 형식입니다.
   - `source-ref`의 이미지에 대한 경로가 올바릅니다.
   - 이미지의 최소 크기는 64픽셀 x 64픽셀입니다.
   - 이미지의 최대 크기는 4096픽셀 x 4096픽셀입니다.
2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_INVALID_IMAGE_DIMENSION

오류 메시지

이미지 크기가 허용된 크기와 일치하지 않습니다.

추가 정보

`source-ref` 항목이 참조하는 이미지가 허용된 이미지 크기를 준수하지 않습니다. 최소 크기는 64픽셀입니다. 최대 크기는 4096픽셀입니다. 경계 상자가 있는 이미지에 대해 `ERROR_INVALID_IMAGE_DIMENSION` 항목이 보고됩니다.

자세한 내용은 지침 및 할당량 섹션을 참조하세요.

`ERROR_INVALID_IMAGE_DIMENSION` 오류를 수정하려면(콘솔)

1. Amazon S3 버킷의 이미지를 Amazon Rekognition Custom Labels가 처리할 수 있는 크기로 업데이트합니다.

2. Amazon Rekognition Custom Labels 콘솔에서 다음을 수행합니다.

   a. 이미지에서 기존 경계 상자를 제거합니다.

   b. 이미지에 경계 상자를 다시 추가합니다.

   c. 변경 내용을 저장합니다.

   자세한 내용은 경계 상자로 객체에 레이블 지정 항목을 참조하세요.

ERROR_INVALID_IMAGE_DIMENSION 오류를 수정하려면(SDK)

1. Amazon S3 버킷의 이미지를 Amazon Rekognition Custom Labels가 처리할 수 있는 크기로 업데이트합니다.
2. ListDataSetEntries를 호출하여 이미지의 기존 JSON 라인을 가져옵니다. `SourceRefContains` 입력 파라미터에 이미지의 Amazon S3 위치 및 파일 이름을 지정합니다.
3. UpdateDataSetEntries를 호출하고 이미지의 JSON 라인을 제공하세요. `source-ref`의 값이 Amazon S3 버킷의 이미지 위치와 일치하는지 확인합니다. 경계 상자 주석을 업데이트하여 업데이트된 이미지에 필요한 경계 상자 크기와 일치하도록 하세요.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
```

```
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2013-11-18T02:53:27",
  "job-name": "my job"
 }
}
```

ERROR_INVALID_BOUNDING_BOX

오류 메시지

경계 상자에 오프 프레임 값이 있습니다.

추가 정보

경계 상자 정보가 이미지 프레임을 벗어나거나 음수 값을 포함하는 이미지를 지정하고 있습니다.

자세한 내용은 지침 및 할당량 섹션을 참조하세요.

ERROR_INVALID_BOUNDING_BOX 오류를 수정하려면

1. annotations 배열의 경계 상자 값을 확인하세요.

```
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
```

```
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }]
 },
```

2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_NO_VALID_ANNOTATIONS

오류 메시지

유효한 주석을 찾을 수 없습니다.

추가 정보

JSON 라인의 주석 객체에 유효한 경계 상자 정보가 포함되어 있지 않습니다.

ERROR_NO_VALID_ANNOTATIONS 오류를 수정하려면

1. 유효한 경계 상자 객체를 포함하도록 annotations 배열을 업데이트하세요. 또한 레이블 속성 메타데이터의 해당 경계 상자 정보(confidence 및 class_map)가 올바른지 확인하세요. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
```

```
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [
  >{
   "confidence": 1          #confidence  object
  },
        {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",    #label
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

2. 매니페스트 파일에서 JSON 라인을 업데이트하거나 제거합니다.

Amazon Rekognition Custom Labels 콘솔로는 이 오류를 수정할 수 없습니다.

ERROR_BOUNDING_BOX_TOO_SMALL

오류 메시지

경계 상자의 높이와 너비가 너무 작습니다.

추가 정보

경계 상자 크기(높이 및 너비)는 1 x 1픽셀보다 커야 합니다.

Amazon Rekognition Custom Labels는 훈련 중에 크기가 1280픽셀을 초과하는 경우 이미지 크기를 조정합니다(소스 이미지는 영향을 받지 않음). 결과 경계 상자의 높이와 너비는 1 x 1픽셀보다 커야 합니

다. 경계 상자 위치는 객체 위치 JSON 라인의 annotations 배열에 저장됩니다. 자세한 정보는 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

오류 정보가 주석 객체에 추가됩니다.

ERROR_BOUNDING_BOX_TOO_SMALL 오류를 해결하려면

- 다음 옵션 중 하나를 선택합니다.
  - 너무 작은 경계 상자의 크기를 늘리세요.
  - 너무 작은 경계 상자를 제거하세요. 경계 상자 제거에 대한 자세한 내용은 ERROR_TOO_MANY_BOUNDING_BOXES 항목을 참조하세요.
  - 매니페스트에서 이미지(JSON 라인)를 제거합니다.

ERROR_TOO_MANY_BOUNDING_BOXES

오류 메시지

경계 상자가 허용된 최대 값보다 많습니다.

추가 정보

경계 상자가 허용된 한도(50개)보다 많습니다. Amazon Rekognition Custom Labels 콘솔에서 초과 경계 상자를 제거하거나 JSON 라인에서 제거할 수 있습니다.

ERROR_TOO_MANY_BOUNDING_BOXES 오류를 수정하려면(콘솔)

1. 어떤 경계 상자를 제거할지 결정하세요.
2. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
3. 사용자 지정 레이블 사용을 선택합니다.
4. 시작하기를 선택합니다.
5. 왼쪽 탐색 창에서 사용하려는 데이터 세트가 포함된 프로젝트를 선택합니다.
6. 데이터 세트 항목에서 사용하려는 데이터 세트를 선택합니다.
7. 데이터 세트 갤러리 페이지에서 레이블 지정 시작을 선택하여 레이블 지정 모드로 전환합니다.
8. 경계 상자를 제거할 이미지를 선택합니다.
9. 경계 상자 그리기를 선택합니다.
10. 그리기 도구에서 삭제할 경계 상자를 선택합니다.
11. 키보드의 Delete 키를 눌러 경계 상자를 삭제합니다.
12. 경계 상자를 충분히 삭제할 때까지 이전 2단계를 반복합니다.
13. 완료를 선택합니다.
14. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.
15. 종료를 선택하여 레이블 지정 모드를 종료합니다.

ERROR_TOO_MANY_BOUNDING_BOXES 오류를 수정하려면(JSON 라인)

1. 매니페스트 파일을 열고 ERROR_TOO_MANY_BOXING_BOXES 오류가 발생한 JSON 라인으로 이동합니다.
2. 제거하려는 각 경계 상자에 대해 다음을 제거합니다.

   - `annotations` 배열에서 필수 `annotation` 객체를 제거합니다.
   - 레이블 속성 메타데이터의 `objects` 배열에서 해당하는 `confidence` 객체를 제거합니다.
   - 다른 경계 상자에서 더 이상 사용하지 않는 경우 `class-map`에서 레이블을 제거하세요.

   다음 예제를 사용하여 제거할 항목을 식별하세요.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
```

```
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [
  >{
   "confidence": 1          #confidence  object
  },
        {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",    #label
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

WARNING_UNANNOTATED_RECORD

경고 메시지

레코드에 주석이 없습니다.

추가 정보

Amazon Rekognition Custom Labels 콘솔을 사용하여 데이터 세트에 추가한 이미지에 레이블이 지정되지 않았습니다. 이미지의 JSON 라인이 훈련에 사용되지 않습니다.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "warnings": [
        {
            "code": "WARNING_UNANNOTATED_RECORD",
            "message": "Record is unannotated."
        }
    ]
}
```

WARNING_UNANNOTATED_RECORD 오류를 수정하려면

- Amazon Rekognition Custom Labels 콘솔을 사용하여 이미지에 레이블을 지정합니다. 지침은 이미지에 이미지 수준 레이블 지정 섹션을 참조하세요.

WARNING_NO_ANNOTATIONS

경고 메시지

주석이 제공되지 않았습니다.

추가 정보

사람이 주석을 달았음(`human-annotated = yes`)에도 불구하고 객체 위치 파악 형식의 JSON 라인에 경계 상자 정보가 없습니다. JSON 라인은 유효하지만 훈련에는 사용되지 않습니다. 자세한 내용은 훈련 및 테스트 검증 결과 매니페스트의 이해 섹션을 참조하세요.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
```

```
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

WARNING_NO_ANNOTATIONS 문제를 해결하려면

- 다음 옵션 중 하나를 선택합니다.

- 경계 상자(annotations) 정보를 JSON 라인에 추가합니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.
- 매니페스트에서 이미지(JSON 라인)를 제거합니다.

WARNING_NO_ATTRIBUTE_ANNOTATIONS

경고 메시지

속성 주석이 제공되지 않았습니다.

추가 정보

사람이 주석을 달았음(human-annotated = yes)에도 불구하고 객체 위치 파악 형식의 JSON 라인에 경계 상자 주석 정보가 없습니다. annotations 배열이 없거나 채워지지 않았습니다. JSON 라인은 유효하지만 훈련에는 사용되지 않습니다. 자세한 내용은 훈련 및 테스트 검증 결과 매니페스트의 이해 섹션을 참조하세요.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
```

```
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

WARNING_NO_ATTRIBUTE_ANNOTATIONS 문제를 해결하려면

- 다음 옵션 중 하나를 선택합니다.
  - 하나 이상의 경계 상자 annotation 객체를 JSON 라인에 추가합니다. 자세한 내용은 매니페스트 파일의 객체 위치 파악 섹션을 참조하세요.
  - 경계 상자 속성을 제거합니다.
  - 매니페스트에서 이미지(JSON 라인)를 제거합니다. JSON 라인에 다른 유효한 경계 상자 속성이 있는 경우 대신 JSON 라인에서 잘못된 경계 상자 속성만 제거할 수 있습니다.

ERROR_UNSUPTED_USE_CASE_TYPE

경고 메시지

추가 정보

type 필드의 값이 groundtruth/image-classification 또는 groundtruth/object-detection 항목이 아닙니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

```
{
    "source-ref": "s3://bucket/test_normal_8.jpg",
    "BB": {
        "annotations": [
            {
                "left": 1768,
```

```
                "top": 1007,
                "width": 448,
                "height": 295,
                "class_id": 0
            },
            {
                "left": 1794,
                "top": 1306,
                "width": 432,
                "height": 411,
                "class_id": 1
            },
            {
                "left": 2568,
                "top": 1346,
                "width": 710,
                "height": 305,
                "class_id": 2
            },
            {
                "left": 2571,
                "top": 1020,
                "width": 644,
                "height": 312,
                "class_id": 3
            }
        ],
        "image_size": [
            {
                "width": 4000,
                "height": 2667,
                "depth": 3
            }
        ]
    },
    "BB-metadata": {
        "job-name": "labeling-job/BB",
        "class-map": {
            "0": "comparator",
            "1": "pot_resistor",
            "2": "ir_phototransistor",
            "3": "ir_led"
        },
        "human-annotated": "yes",
```

```
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "creation-date": "2021-06-22T09:58:34.811Z",
        "type": "groundtruth/wrongtype",
        "cl-errors": [
            {
                "code": "ERROR_UNSUPPORTED_USE_CASE_TYPE",
                "message": "The use case type of the BB-metadata label attribute
 metadata is unsupported. Check the type field."
            }
        ]
    },
    "cl-metadata": {
        "is_labeled": true
    },
    "cl-errors": [
        {
            "code": "ERROR_NO_VALID_LABEL_ATTRIBUTES",
            "message": "No valid label attributes found."
        }
    ]
}
```

ERROR_UNSUPTED_USE_CASE_TYPE 오류를 수정하려면

- 다음 옵션 중 하나를 선택합니다.
  - 생성하려는 모델 유형에 따라 `type` 필드 값을 `groundtruth/image-classification` 또는 `groundtruth/object-detection` 항목으로 변경합니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

- 매니페스트에서 이미지(JSON 라인)를 제거합니다.

ERROR_INVALID_LABEL_NAME_LENGTH

추가 정보

레이블 이름이 너무 깁니다. 최대 길이는 256자입니다.

ERROR_INVALID_LABEL_NAME_LENGTH 오류를 수정하려면

- 다음 옵션 중 하나를 선택합니다.
  - 레이블 이름을 256자 이하로 줄이세요.
  - 매니페스트에서 이미지(JSON 라인)를 제거합니다.

훈련된 Amazon Rekognition Custom Labels 모델 개선

훈련이 완료되면 모델의 성능을 평가합니다. Amazon Rekognition Custom Labels는 사용자를 돕기 위해 각 레이블에 대한 요약 지표 및 평가 지표를 제공합니다. 사용 가능한 지표에 대한 자세한 내용은 모델 평가를 위한 지표 항목을 참조하세요. 지표를 사용하여 모델을 개선하려면 Amazon Rekognition Custom Labels 모델 개선 항목을 참조하세요.

모델의 정확도에 만족하는 경우 사용을 시작할 수 있습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 실행 섹션을 참조하세요.

주제

- 모델 평가를 위한 지표
- 평가 지표 액세스(콘솔)
- Amazon Rekognition Custom Labels 평가 지표에 액세스(SDK)
- Amazon Rekognition Custom Labels 모델 개선

모델 평가를 위한 지표

모델을 훈련한 후 Amazon Rekognition Custom Labels는 모델 테스트에서 지표를 반환하며, 이를 사용하여 모델의 성능을 평가할 수 있습니다. 이 주제는 사용할 수 있는 지표와 훈련된 모델이 제대로 작동하고 있는지 이해하는 방법을 설명합니다.

Amazon Rekognition Custom Labels 콘솔은 다음 지표를 훈련 결과의 요약과 각 레이블의 지표로 제공합니다.

- 정밀도
- 재현율
- F1

제공하는 각 지표는 기계 학습 모델의 성능을 평가하는 데 일반적으로 사용되는 지표입니다. Amazon Rekognition Custom Labels는 전체 테스트 데이터 세트에 대한 테스트 결과에 대한 지표를 각 사용자 지정 레이블에 대한 지표와 함께 반환합니다. 또한 테스트 데이터 세트의 각 이미지에 대해 훈련된 사용자 지정 모델의 성능을 검토할 수 있습니다. 자세한 내용은 평가 지표 액세스(콘솔) 섹션을 참조하세요.

모델 성능 평가

테스트 중에 Amazon Rekognition Custom Labels는 테스트 이미지에 사용자 지정 레이블이 포함되어 있는지 예측합니다. 신뢰도 점수는 모델 예측의 확실성을 정량화하는 값입니다.

사용자 지정 레이블의 신뢰도 점수가 임계값을 초과하는 경우 모델 출력에 이 레이블이 포함됩니다. 예측은 다음과 같은 방법으로 분류할 수 있습니다.

- 참 긍정: Amazon Rekognition Custom Labels 모델이 테스트 이미지에 사용자 지정 레이블이 있는지 정확하게 예측했습니다. 즉, 예측 레이블은 해당 이미지에 대한 “실측 정보” 레이블이기도 합니다. 예를 들어 Amazon Rekognition Custom Labels가 이미지에 축구공이 있을 때 축구공 레이블을 올바르게 반환한 경우입니다.
- 거짓 긍정: Amazon Rekognition Custom Labels 모델이 테스트 이미지에 사용자 지정 레이블이 있는지 잘못 예측했습니다. 즉, 예측 레이블은 이미지에 대한 실측 정보 레이블이 아닙니다. 예를 들어 Amazon Rekognition Custom Labels는 축구공 레이블을 반환했지만 해당 이미지에 대한 실측 정보에는 축구공 레이블이 없는 경우입니다.
- 거짓 부정: Amazon Rekognition Custom Labels 모델이 이미지에 사용자 지정 레이블이 존재할 것으로 예측하지 않았지만 해당 이미지에 대한 “실측 정보”에는 이 레이블이 들어 있었습니다. 예를 들어 Amazon Rekognition Custom Labels가 축구공이 들어 있는 이미지에 대해 '축구공' 사용자 지정 레이블을 반환하지 않은 경우입니다.
- 참 부정: Amazon Rekognition Custom Labels 모델이 테스트 이미지에 사용자 지정 레이블이 없을 것이라고 정확하게 예측했습니다. 예를 들어 Amazon Rekognition Custom Labels가 축구공이 없는 이미지에 대해 '축구공' 사용자 지정 레이블을 반환하지 않은 경우입니다.

콘솔에서는 테스트 데이터 세트의 각 이미지에 대한 참 긍정, 거짓 긍정 및 거짓 부정 값에 액세스할 수 있습니다. 자세한 내용은 평가 지표 액세스(콘솔) 섹션을 참조하세요.

이러한 예측 결과는 각 레이블에 대한 다음 지표를 계산하고 전체 테스트 세트에 대한 집계를 계산하는 데 사용됩니다. 모든 지표가 각 테스트 이미지의 각 경계 상자(예측 또는 실측 정보)에 대해 계산된다는 점을 제외하면 경계 상자 수준에서 모델이 수행한 예측에도 동일한 정의가 적용됩니다.

교차 결합(IoU) 및 객체 감지

교차 결합(IoU)은 두 객체의 경계 상자가 합쳐진 영역에서 겹치는 비율을 측정합니다. 범위는 0(가장 조금 겹침)에서 1(완전히 겹침)까지입니다. 테스트 중에 실측 정보 경계 상자와 예측 경계 상자의 IoU가 0.5 이상이면 예측 경계 상자가 정확한 것으로 판정됩니다.

추정 임계값

Amazon Rekognition Custom Labels는 각 사용자 지정 레이블에 대해 추정 임계값(0~1)을 자동으로 계산합니다. 사용자 지정 레이블에 대해 추정 임계값을 설정할 수 없습니다. 각 레이블의 추정 임계값은 예측을 참 긍정 또는 거짓 긍정으로 계산하는 위의 값입니다. 테스트 데이터 세트를 기반으로 설정됩니다. 추정 임계값은 모델 훈련 중에 테스트 데이터 세트에서 달성한 최고의 F1 점수를 기반으로 계산됩니다.

모델의 훈련 결과에서 레이블에 대한 추정 임계값 값을 가져올 수 있습니다. 자세한 내용은 평가 지표 액세스(콘솔) 섹션을 참조하세요.

추정 임계값의 변경은 일반적으로 모델의 정밀도와 재현율을 개선하는 데 사용됩니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 개선 섹션을 참조하세요. 사용자는 레이블에 대해 모델의 추정 임계값을 설정할 수 없으므로 이미지를 `DetectCustomLabels`로 분석하고 `MinConfidence` 입력 파라미터를 지정하여 동일한 결과를 얻을 수 있습니다. 자세한 내용은 훈련된 모델을 사용한 이미지 분석 섹션을 참조하세요.

정밀도

Amazon Rekognition Custom Labels는 각 레이블에 대한 정밀도 지표와 전체 테스트 데이터 세트에 대한 평균 정밀도 지표를 제공합니다.

정밀도는 특정 레이블의 추정 임계값에서 모든 모델 예측(참 긍정 및 거짓 긍정)에 대한 올바른 예측(참 긍정)의 비율입니다. 임계값이 증가하면 모델의 예측 수가 줄어들 수 있습니다. 그러나 일반적으로 낮은 임계값에 비해 참 긍정 비율과 거짓 긍정 비율이 더 높을 것입니다. 가능한 정밀도 값의 범위는 0~1이며, 값이 높을수록 정밀도가 높습니다.

예를 들어, 모델 영상에 축구공이 있다고 예측한 경우에서 예측이 정확했을 때가 얼마나 많았는지 따져봅시다. 축구공 8개와 바위 5개가 있는 이미지가 있다고 가정해 보겠습니다. 모델이 축구공 9개(정확히 예측된 8개와 거짓 긍정 1개)를 예측한 경우 이 예제의 정밀도는 0.89입니다. 그러나 모델이 이미지에서 축구공 13개를 예측한 경우 결과는 올바른 예측 8개와 틀린 예측 5개가 되므로 정밀도는 낮아집니다.

자세한 내용은 정밀도 및 재현율을 참조하세요.

재현율

Amazon Rekognition Custom Labels는 각 레이블의 평균 재현율 지표와 전체 테스트 데이터 세트에 대한 평균 재현율 지표를 제공합니다.

재현율은 테스트 세트 레이블 중에서 추정 임계값을 초과할 것으로 정확하게 예측된 결과의 비율입니다. 사용자 지정 레이블이 테스트 세트의 이미지에 실제로 존재할 때 모델이 얼마나 자주 사용자 지정 레이블을 정확하게 예측할 수 있는지를 나타내는 척도입니다. 재현율 범위는 0~1입니다. 값이 높을수록 재현율이 높음을 나타냅니다.

예를 들어 이미지에 축구공 8개가 있을 때 그중 몇 개가 정확히 감지되었는지 따져 봅시다. 이미지에 축구공 8개와 바위 5개가 있는 이 예제에서 모델이 축구공 5개를 감지하면 재현율 값은 0.62입니다. 재훈련 후 새 모델이 이미지에 있는 8개를 모두 포함하여 축구공 9개를 감지한 경우 재현율 값은 1.0입니다.

자세한 내용은 정밀도 및 재현율을 참조하세요.

F1

Amazon Rekognition Custom Labels는 F1 점수 지표를 사용하여 각 레이블의 평균 모델 성능과 전체 테스트 데이터 세트의 평균 모델 성능을 측정합니다.

모델 성능은 모든 레이블의 정밀도와 재현율을 모두 고려한 종합 지표입니다(예: F1 점수 또는 평균 정밀도). 모델 성능 점수는 0에서 1 사이의 값입니다. 값이 높을수록 재현율과 정밀도 양면에서 모델의 성능이 좋습니다. 구체적으로, 분류 작업을 위한 모델 성능은 일반적으로 F1 점수로 측정됩니다. 이 점수는 추정 임계값에서의 정밀도 및 재현율 점수의 조화 평균입니다. 예를 들어 정밀도가 0.9이고 재현율이 1.0인 모델의 경우 F1 점수는 0.947입니다.

F1 점수 값이 높으면 모델이 정밀도와 재현율 모두에서 우수한 성능을 보이고 있음을 나타냅니다. 예를 들어 정밀도가 0.30이고 재현율이 1.0인 경우와 같이 모델의 성능이 좋지 않은 경우 F1 점수는 0.46입니다. 마찬가지로 정밀도가 높고(0.95) 재현율이 낮으면(0.20) F1 점수는 0.33입니다. 두 경우 모두 F1 점수가 낮으며 이는 모델에 문제가 있음을 나타냅니다.

자세한 내용은 F1 점수를 참조하세요.

지표 사용

훈련한 특정 모델과 응용 분야에 따라 `MinConfidence` 입력 파라미터를 `DetectCustomLabels`로 사용하여 정밀도와 재현율 사이에서 균형을 맞출 수 있습니다. `MinConfidence` 값이 높을수록 일반적으로 정밀도는 높아지지만(축구공의 예측치가 더 정확함) 재현율은 낮아집니다(실제 축구공을 놓치는 경우가 많음). `MinConfidence` 값이 낮을수록 재현율은 높아지지만(정확하게 예측된 실제 축구공이 더 많음) 정밀도는 낮아집니다(이러한 예측이 틀리는 경우가 많음). 자세한 내용은 훈련된 모델을 사용한 이미지 분석 섹션을 참조하세요.

지표는 필요한 경우 모델 성능을 개선하기 위해 취할 수 있는 조치도 알려줍니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 개선 섹션을 참조하세요.

> **Note**
>
> `DetectCustomLabels`는 지표 범위인 0~1에 해당하는 0에서 100 사이의 예측값을 반환합니다.

평가 지표 액세스(콘솔)

테스트 중에는 테스트 데이터 세트를 기준으로 모델의 성능을 평가합니다. 테스트 데이터 세트의 레이블은 실제 이미지가 나타내는 내용을 나타내므로 '실측 정보'로 간주됩니다. 테스트 중에 모델은 테스트 데이터 세트를 사용하여 예측을 수행합니다. 예측된 레이블을 실측 레이블과 비교하고 콘솔 평가 페이지에서 결과를 확인할 수 있습니다.

Amazon Rekognition Custom Labels 콘솔은 전체 모델의 요약 지표와 개별 레이블에 대한 지표를 보여줍니다. 콘솔에서 사용할 수 있는 지표는 정밀도 재현율, F1 점수, 신뢰도, 신뢰도 임계값입니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 섹션을 참조하세요.

콘솔을 사용하여 개별 지표에 집중할 수 있습니다. 예를 들어 레이블의 정밀도 문제를 조사하려면 훈련 결과를 레이블과 거짓 긍정 결과를 기준으로 필터링할 수 있습니다. 자세한 내용은 모델 평가를 위한 지표 섹션을 참조하세요.

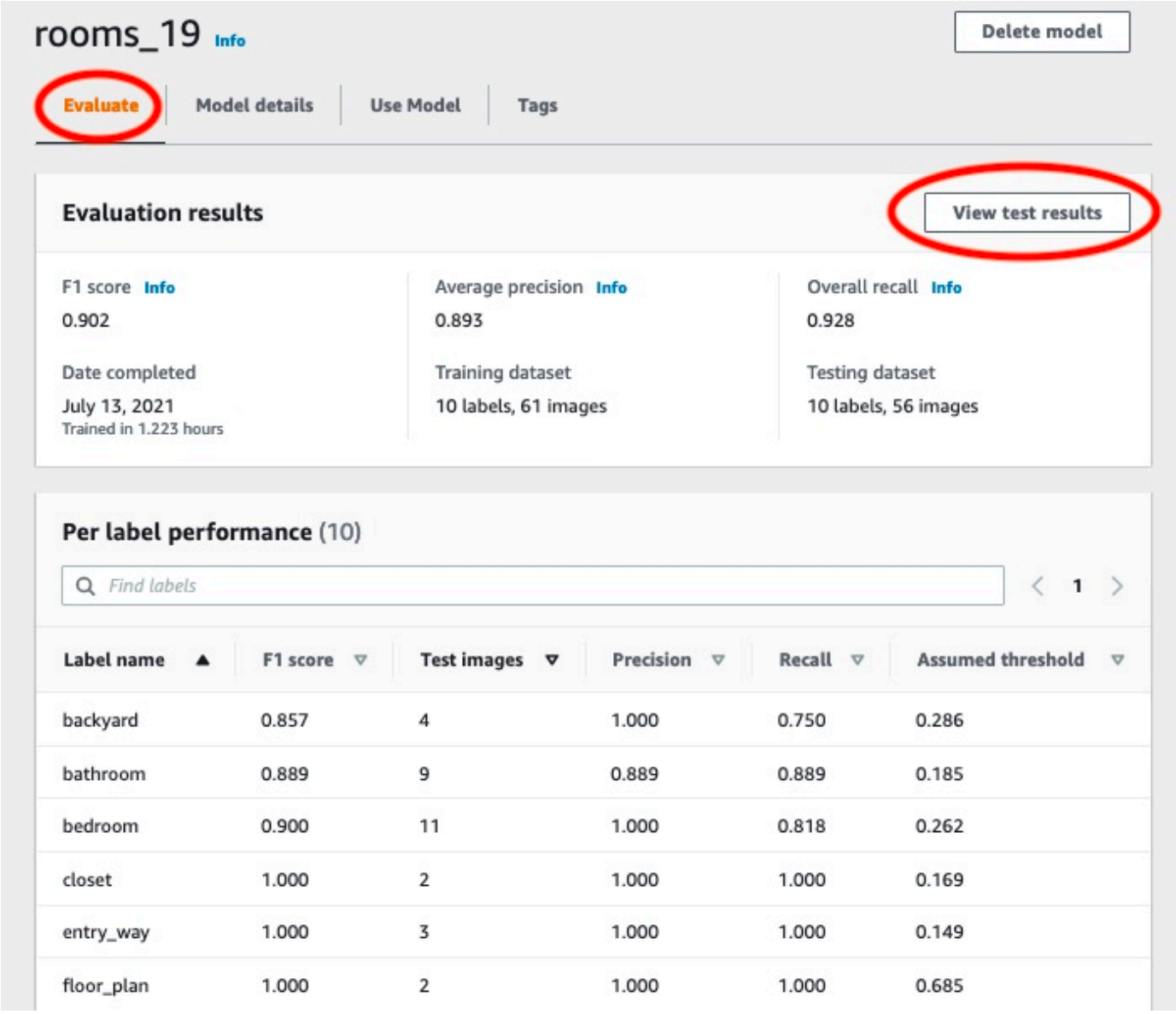
훈련 후에는 훈련 데이터 세트가 읽기 전용이 됩니다. 모델을 개선하기로 결정한 경우 훈련 데이터 세트를 새 데이터세트에 복사할 수 있습니다. 데이터 세트 사본을 사용하여 모델의 새 버전을 훈련할 수 있습니다.

이 단계에서는 콘솔을 사용하여 콘솔의 훈련 결과에 액세스할 수 있습니다.

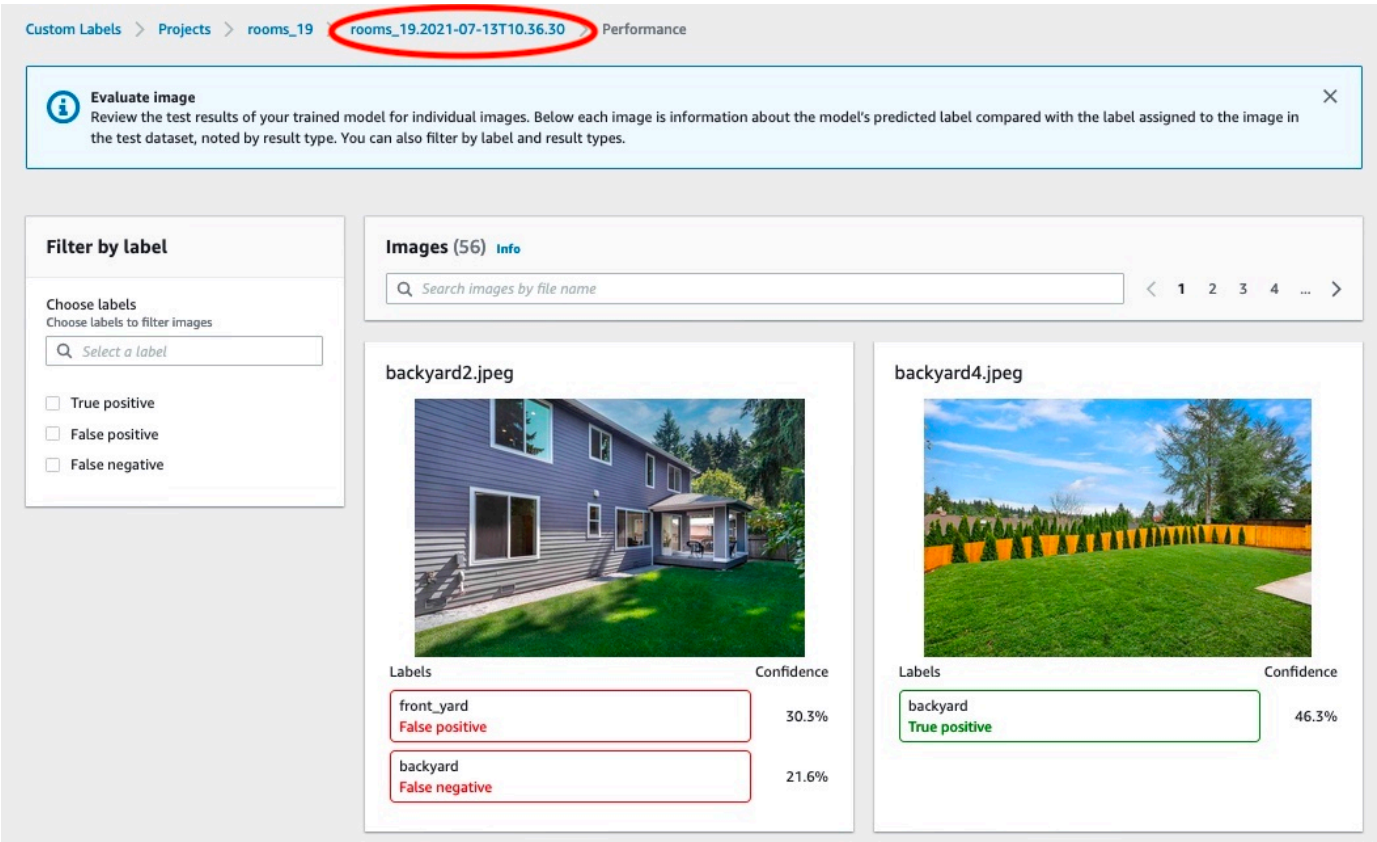
평가 지표에 액세스하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 평가하려는 훈련된 모델이 포함된 프로젝트를 선택합니다.
6. 모델에서 평가하려는 모델을 선택합니다.
7. 평가 탭을 선택하여 평가 결과를 확인합니다. 모델 평가에 대한 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 개선 항목을 참조하세요.

8. 개별 테스트 이미지의 결과를 보려면 테스트 결과 보기를 선택합니다. 자세한 정보는 모델 평가를 위한 지표을 참조하세요. 모델 평가 요약의 다음 스크린샷은 테스트 결과 및 성능 지표와 함께 6개 레이블에 대한 F1 점수, 평균 정밀도 및 전체 재현율을 보여줍니다. 학습된 모델 사용에 대한 세부 정보도 제공됩니다.



9. 테스트 결과를 확인한 후 프로젝트 이름을 선택하여 모델 페이지로 돌아가세요. 테스트 결과 페이지에는 뒷마당 및 앞마당 이미지 카테고리에서 학습된 기계 학습 모델에 대한 예측 레이블 및 신뢰도 점수가 포함된 이미지가 표시됩니다. 두 개의 예제 이미지가 표시됩니다.



10. 지표를 사용하여 모델의 성능을 평가하세요. 자세한 내용은 Amazon Rekognition Custom Labels 모델 개선 섹션을 참조하세요.

Amazon Rekognition Custom Labels 평가 지표에 액세스(SDK)

DescribeProject버전 작업을 통해 콘솔에서 제공하는 것 이상의 메트릭에 액세스할 수 있습니다.

콘솔과 마찬가지로 `DescribeProjectVersions`는 테스트 결과에 대한 요약 정보 및 각 레이블의 테스트 결과로 다음 지표에 대한 액세스를 제공합니다.

- 정밀도
- 재현율
- F1

모든 레이블의 평균 임계값과 개별 레이블의 임계값이 반환됩니다.

`DescribeProjectVersions`는 분류 및 이미지 감지를 위한 다음 지표에 대한 액세스를 제공합니다(이미지 상의 물체 위치).

- 이미지 분류를 위한 오차 행렬 자세한 내용은 모델의 오차 행렬 보기 섹션을 참조하세요.
- 이미지 감지를 위한 Mean Average Precision(mAP)
- 이미지 감지를 위한 Mean Average Recall(mAR)

DescribeProjectVersions는 참 긍정, 거짓 긍정, 거짓 부정 및 참 부정 값에 대한 액세스를 제공합니다. 자세한 내용은 모델 평가를 위한 지표 섹션을 참조하세요.

집계된 F1 점수 지표는 DescribeProjectVersions에서 직접 반환됩니다. Amazon S3 버킷에 저장된 요약 파일 및 평가 매니페스트 스냅샷 파일에서 다른 지표에 액세스할 수 있습니다. 자세한 내용은 요약 파일 및 평가 매니페스트 스냅샷 액세스(SDK) 섹션을 참조하세요.

주제

- 요약 파일
- 평가 매니페스트 스냅샷
- 요약 파일 및 평가 매니페스트 스냅샷 액세스(SDK)
- 모델의 오차 행렬 보기
- 참조: 훈련 결과 요약 파일

요약 파일

요약 파일에는 모델 전체에 대한 평가 결과 정보와 각 레이블의 지표가 포함되어 있습니다. 포함되는 지표는 정밀도, 재현율, F1 점수입니다. 모델의 임계값도 제공됩니다. DescribeProjectVersions에서 반환된 EvaluationResult 객체에서 요약 파일 위치에 액세스할 수 있습니다. 자세한 내용은 참조: 훈련 결과 요약 파일 섹션을 참조하세요.

다음은 예제 요약 파일입니다.

```
{
  "Version": 1,
  "AggregatedEvaluationResults": {
    "ConfusionMatrix": [
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "CAP",
        "Value": 0.9948717948717949
      },
      {
```

```
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "WATCH",
        "Value": 0.008547008547008548
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "CAP",
        "Value": 0.1794871794871795
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "WATCH",
        "Value": 0.7008547008547008
      }
    ],
    "F1Score": 0.9726959470546408,
    "Precision": 0.9719115848331294,
    "Recall": 0.9735042735042735
  },
  "EvaluationDetails": {
    "EvaluationEndTimestamp": "2019-11-21T07:30:23.910943",
    "Labels": [
      "CAP",
      "WATCH"
    ],
    "NumberOfTestingImages": 624,
    "NumberOfTrainingImages": 5216,
    "ProjectVersionArn": "arn:aws:rekognition:us-east-1:nnnnnnnnn:project/my-project/
version/v0/1574317227432"
  },
  "LabelEvaluationResults": [
    {
      "Label": "CAP",
      "Metrics": {
        "F1Score": 0.9794344473007711,
        "Precision": 0.9819587628865979,
        "Recall": 0.9769230769230769,
        "Threshold": 0.9879502058029175
      },
      "NumberOfTestingImages": 390
    },
    {
      "Label": "WATCH",
      "Metrics": {
```

```
        "F1Score": 0.9659574468085106,
        "Precision": 0.961864406779661,
        "Recall": 0.9700854700854701,
        "Threshold": 0.014450683258473873
      },
      "NumberOfTestingImages": 234
    }
  ]
}
```

평가 매니페스트 스냅샷

평가 매니페스트 스냅샷에는 테스트 결과에 대한 세부 정보가 포함됩니다. 스냅샷에는 각 예측에 대한 신뢰도가 포함됩니다. 또한 영상의 실제 분류(참 긍정, 참 부정, 거짓 긍정 또는 거짓 부정)와 비교한 예측의 분류도 포함됩니다.

테스트 및 훈련에 사용할 수 있는 이미지만 포함되므로 파일은 스냅샷입니다. 잘못된 형식의 이미지와 같이 검증할 수 없는 이미지는 매니페스트에 포함되지 않습니다. `DescribeProjectVersions`에서 반환된 `TestingDataResult` 객체에서 테스트 스냅샷 위치에 액세스할 수 있습니다. `DescribeProjectVersions`에서 반환된 `TrainingDataResult` 객체에서 훈련 스냅샷 위치에 액세스할 수 있습니다.

스냅샷은 탐지의 이진 분류 결과와 같은 추가 정보를 제공하기 위해 필드가 추가된 SageMaker Ground Truth 매니페스트 출력 형식입니다. 다음 스니펫은 추가 필드를 보여줍니다.

```
"rekognition-custom-labels-evaluation-details": {
    "version": 1,
    "is-true-positive": true,
    "is-true-negative": false,
    "is-false-positive": false,
    "is-false-negative": false,
    "is-present-in-ground-truth": true
    "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- 버전: 매니페스트 스냅샷에 있는 `rekognition-custom-labels-evaluation-details` 필드 형식의 버전입니다.
- is-true-positive... : 신뢰도 점수가 레이블의 최소 임계값에 비해서 어떤지를 기반으로 한 예측의 바이너리 분류입니다.

- is-present-in-ground-truth: 모델을 통해 예측한 내용이 훈련에 사용된 실측 정보에 포함되면 참이고, 그렇지 않으면 거짓입니다. 이 값은 신뢰 점수가 모델에서 계산한 최소 임계값을 초과하는지 여부를 기반으로 하지 않습니다.
- Ground Truth-labeling-jobs: 매니페스트 라인에 있는 훈련에 사용되는 실측 정보 필드의 목록입니다.

SageMaker Ground Truth 매니페스트 형식에 대한 자세한 내용은 출력을 참조하십시오.

다음은 이미지 분류 및 객체 감지에 대한 지표를 보여주는 테스트 매니페스트 스냅샷의 예입니다.

```
// For image classification
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
    "human-annotated": "no",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  }
}
```

```
// For object detection
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-training-0-metadata": {
    "job-name": "rekognition-custom-labels-training-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "yes",
    "objects": [
      {
        "confidence": 1.0
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  },
  "rekognition-custom-labels-evaluation": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
```

```
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-evaluation-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "no",
    "objects": [
      {
        "confidence": 0.95,
        "rekognition-custom-labels-evaluation-details": {
          "version": 1,
          "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
          "is-true-positive": true,
          "is-true-negative": false,
          "is-false-positive": false,
          "is-false-negative": false,
          "is-present-in-ground-truth": true
        }
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  }
}
```

요약 파일 및 평가 매니페스트 스냅샷 액세스(SDK)

학습 결과를 얻으려면 Versions를 DescribeProject 호출해야 합니다. 예제 코드는 모델 설명(SDK) 항목을 참조하세요.

지표의 위치는 DescribeProjectVersions의 ProjectVersionDescription 응답에 반환됩니다.

- EvaluationResult: 요약 파일의 위치
- TestingDataResult: 테스트에 사용된 평가 매니페스트 스냅샷의 위치

F1 점수와 요약 파일 위치가 EvaluationResult에 반환됩니다. 예:

```
"EvaluationResult": {
                "F1Score": 1.0,
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
            }
```

평가 매니페스트 스냅샷은 사용자가 모델 훈련(SDK)에서 지정한 --output-config 입력 파라미터에서 지정된 위치에 저장됩니다.

> **Note**
>
> 훈련 요금이 청구된 시간(초)으로 BillableTrainingTimeInSeconds에서 반환됩니다.

Amazon Rekognition Custom Labels에서 반환되는 지표에 대한 자세한 내용은 Amazon Rekognition Custom Labels 평가 지표에 액세스(SDK) 지표를 참조하세요.

모델의 오차 행렬 보기

오차 행렬을 사용하면 모델이 모델의 다른 레이블과 혼동하는 레이블을 확인할 수 있습니다. 오차 행렬을 사용하면 모델 개선에 집중할 수 있습니다.

모델 평가 중에 Amazon Rekognition Custom Labels는 테스트 이미지를 사용하여 잘못 식별된(혼동된) 레이블을 식별함으로써 오차 행렬을 생성합니다. Amazon Rekognition Custom Labels는 분류 모델에 오차 행렬만 생성합니다. Amazon Rekognition Custom Labels가 모델 교육 중에 생성하는 요약 파일에서 분류 행렬에 액세스할 수 있습니다. Amazon Rekognition Custom Labels 콘솔에서는 오차 행렬을 볼 수 없습니다.

주제

오차 행렬 사용

다음 표는 Rooms 이미지 분류 예제 프로젝트의 오차 행렬입니다. 열 제목은 테스트 이미지에 할당된 레이블(실측 정보 레이블)입니다. 행 제목은 모델이 테스트 이미지에 대해 예측하는 레이블입니다. 각 셀은 실측 정보 레이블(열)이 되어야 하는 레이블(행)에 대한 예측의 백분율입니다. 예를 들어, 욕실에 대한 예측의 67%가 욕실로 올바르게 레이블 지정되었고, 욕실의 33%가 주방으로 잘못 레이블 지정되었을 수 있습니다. 성능이 높은 모델은 예측 레이블이 실측 정보 레이블과 일치하여 셀 값이 높습니다. 이를 첫 번째 예측 레이블부터 마지막 예측 레이블 및 실측 정보 레이블까지 대각선으로 볼 수 있습니다. 셀 값이 0인 경우 셀의 실측 정보 레이블이어야 하는 셀의 예측 레이블에 대한 예측은 수행되지 않았다는 뜻입니다.

> **Note**
>
> 모델은 비결정적이므로 Rooms 프로젝트를 훈련하여 얻은 오차 행렬 셀 값은 다음 표와 다를 수 있습니다.

오차 행렬은 집중해야 할 영역을 식별합니다. 예를 들어, 오차 행렬은 모델이 옷장을 침실과 혼동한 경우가 50%라는 것을 보여줍니다. 이 상황에서는 훈련 데이터 세트에 옷장과 침실 이미지를 더 추가해야 합니다. 또한 기존 옷장 및 침실 이미지에 레이블이 올바르게 지정되어 있는지도 확인해야 합니다. 이렇게 하면 모델이 두 레이블을 더 잘 구분할 수 있습니다. 데이터 세트에 이미지를 더 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요.

오차 행렬도 유용하지만 다른 지표도 고려하는 것이 중요합니다. 예를 들어, 예측의 100%가 평면도 레이블을 올바르게 찾았는데, 이는 성능이 우수하다는 뜻입니다. 하지만 테스트 데이터 세트에는 평면도 레이블이 있는 이미지가 2개만 있습니다. 또한 거실 레이블이 지정된 11개의 이미지도 있습니다. 이러한 불균형은 훈련 데이터 세트(거실 이미지 13개, 옷장 이미지 2개)에도 있습니다. 더 정확하게 평가하

려면 제대로 대표되지 않은 레이블의 이미지(이 예에서는 평면도)를 더 추가하여 훈련 데이터 세트와 테스트 데이터 세트의 균형을 맞추세요. 레이블당 테스트 이미지 수를 가져오려면 평가 지표 액세스(콘솔) 항목을 참조하세요.

	실측 정보 레이블									
예측 레이블	뒷마당	욕실	침실	옷장	진입로	평면도	앞마당	주방	거실	파티오
뒷마당	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%
욕실	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
침실	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
옷장	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%
진입로	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
평면도	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
앞마당	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
주방	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
거실	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
파티오	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

모델의 오차 행렬 가져오기

다음 코드는 DescribeProjects및 DescribeProject버전 연산을 사용하여 모델의 요약 파일을 가져옵니다. 그런 다음 요약 파일을 사용하여 모델의 오차 행렬을 표시합니다.

모델의 오차 행렬 표시하기(SDK)

1. 아직 설치하지 않았다면 및 AWS SDK를 설치하고 구성하세요. AWS CLI 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 다음 코드를 사용하여 모델의 오차 행렬을 표시할 수 있습니다. 다음 명령줄 인수를 제공하세요.
   - `project_name`: 사용하려는 프로젝트의 이름 Amazon Rekognition Custom Labels 콘솔의 프로젝트 페이지에서 프로젝트 이름을 가져올 수 있습니다.
   - `version_name`: 사용하려는 모델의 버전 Amazon Rekognition Custom Labels 콘솔의 프로젝트 세부 정보 페이지에서 버전 이름을 가져올 수 있습니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose

Shows how to display the confusion matrix for an Amazon Rekognition Custom labels
 image
classification model.
"""


import json
import argparse
import logging
import boto3
import pandas as pd
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_summary_location(rek_client, project_name, version_name):
    """
    Get the summary file location for a model.

    :param rek_client: A Boto3 Rekognition client.
    :param project_arn: The Amazon Resource Name (ARN) of the project that contains
 the model.
```

```
    :param model_arn: The Amazon Resource Name (ARN) of the model.
    :return: The location of the model summary file.
    """

    try:
        logger.info(
            "Getting summary file for model %s in project %s.", version_name,
 project_name)

        summary_location = ""

        # Get the project ARN from the project name.
        response = rek_client.describe_projects(ProjectNames=[project_name])

        assert len(response['ProjectDescriptions']) > 0, \
            f"Project {project_name} not found."

        project_arn = response['ProjectDescriptions'][0]['ProjectArn']

        # Get the summary file location for the model.
        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        assert len(describe_response['ProjectVersionDescriptions']) > 0, \
            f"Model {version_name} not found."

        model=describe_response['ProjectVersionDescriptions'][0]

        evaluation_results=model['EvaluationResult']

        summary_location=(f"s3://{evaluation_results['Summary']['S3Object']
['Bucket']}"
                          f"/{evaluation_results['Summary']['S3Object']
['Name']}")

        return summary_location

    except ClientError as err:
        logger.exception(
            "Couldn't get summary file location: %s", err.response['Error']
['Message'])
        raise
```

```
def show_confusion_matrix(summary):
    """
    Shows the confusion matrix for an Amazon Rekognition Custom Labels
    image classification model.
    :param summary: The summary file JSON object.
    """
    pd.options.display.float_format = '{:.0%}'.format

    # Load the model summary JSON into a DataFrame.

    summary_df = pd.DataFrame(
        summary['AggregatedEvaluationResults']['ConfusionMatrix'])

    # Get the confusion matrix.
    confusion_matrix = summary_df.pivot_table(index='PredictedLabel',
                                              columns='GroundTruthLabel',
                                              fill_value=0.0).astype(float)

    # Display the confusion matrix.
    print(confusion_matrix)


def get_summary(s3_resource, summary):
    """
    Gets the summary file.
    : return: The summary file in bytes.
    """
    try:
        summary_bucket, summary_key = summary.replace(
            "s3://", "").split("/", 1)

        bucket = s3_resource.Bucket(summary_bucket)
        obj = bucket.Object(summary_key)
        body = obj.get()['Body'].read()
        logger.info(
            "Got summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't get summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
        raise
    else:
```

```
        return body


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    : param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to describe."
    )


def main():
    """
    Entry point for script.
    """

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get the command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Showing confusion matrix for: {args.version_name} for project
 {args.project_name}.")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
        s3_resource = session.resource('s3')

        # Get the summary file for the model.
        summary_location = get_model_summary_location(rekognition_client,
 args.project_name,
                                                      args.version_name
```

```
                                                    )
        summary = json.loads(get_summary(s3_resource, summary_location))

        # Check that the confusion matrix is available.
        assert 'ConfusionMatrix' in summary['AggregatedEvaluationResults'], \
            "Confusion matrix not found in summary. Is the model a classification
 model?"

        # Show the confusion matrix.
        show_confusion_matrix(summary)
        print("Done")

    except ClientError as err:
        logger.exception("Problem showing confusion matrix: %s", err)
        print(f"Problem describing model: {err}")

    except AssertionError as err:
        logger.exception(
            "Error: %s.\n", err)
        print(
            f"Error: {err}\n")


if __name__ == "__main__":
    main()
```

참조: 훈련 결과 요약 파일

훈련 결과 요약에는 모델을 평가하는 데 사용할 수 있는 지표가 포함되어 있습니다. 요약 파일은 콘솔 훈련 결과 페이지에 지표를 표시하는 데도 사용됩니다. 요약 파일은 훈련 후 Amazon S3 버킷에 저장됩니다. 요약 파일을 가져오려면 `DescribeProjectVersion`를 직접 호출하세요. 예제 코드는 요약 파일 및 평가 매니페스트 스냅샷 액세스(SDK) 항목을 참조하세요.

요약 파일

다음 JSON은 요약 파일의 형식입니다.

EvaluationDetails (섹션 3)

훈련 작업에 대한 개요 정보입니다. 여기에는 모델이 속한 프로젝트의 ARN(`ProjectVersionArn`)), 훈련 종료 날짜 및 시간, 평가된 모델 버전(`EvaluationEndTimestamp`), 교육 중에 감지된

레이블 목록(Labels) 등이 포함됩니다. 또한 훈련(NumberOfTrainingImages) 및 평가(NumberOfTestingImages)에 사용된 이미지 수도 포함됩니다.

AggregatedEvaluationResults (섹션 1)

AggregatedEvaluationResults를 테스트 데이터 세트와 함께 사용할 경우 훈련된 모델의 전체 성능을 평가하는 데 사용할 수 있습니다. 집계된 지표는 Precision, Recall, 및 F1Score 지표에 포함됩니다. 객체 감지(이미지 상의 객체 위치)의 경우 AverageRecall(mAR) 및 AveragePrecision(mAP) 지표가 반환됩니다. 분류(이미지의 객체 유형)의 경우 오차 행렬 지표가 반환됩니다.

LabelEvaluationResults (섹션 2)

labelEvaluationResults를 개별 레이블의 성능을 평가하는 데 사용할 수 있습니다. 레이블은 각 레이블의 F1 점수를 기준으로 정렬됩니다. 포함된 지표는 Precision, Recall, F1Score, Threshold(분류에 사용됨)입니다.

파일 이름은 다음 형식을 띕니다: EvaluationSummary-ProjectName-VersionName.json

```
{
  "Version": "integer",
  // section-3
  "EvaluationDetails": {
    "ProjectVersionArn": "string",
    "EvaluationEndTimestamp": "string",
    "Labels": "[string]",
    "NumberOfTrainingImages": "int",
    "NumberOfTestingImages": "int"
  },
  // section-1
  "AggregatedEvaluationResults": {
    "Metrics": {
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float",
      // The following 2 fields are only applicable to object detection
      "AveragePrecision": "float",
      "AverageRecall": "float",
      // The following field is only applicable to classification
      "ConfusionMatrix":[
        {
```

```
          "GroundTruthLabel": "string",
          "PredictedLabel": "string",
          "Value": "float"
        },
        ...
      ],
    }
  },
  // section-2
  "LabelEvaluationResults": [
    {
      "Label": "string",
      "NumberOfTestingImages", "int",
      "Metrics": {
        "Threshold": "float",
        "Precision": "float",
        "Recall": "float",
        "F1Score": "float"
      },
    },
    ...
  ]
}
```

Amazon Rekognition Custom Labels 모델 개선

기계 학습 모델의 성능은 사용자 지정 레이블(관심 있는 특정 객체 및 장면)의 복잡성 및 가변성, 제공하는 훈련 데이터 세트의 품질 및 대표성, 모델 훈련에 사용되는 모델 프레임워크 및 기계 학습 방법과 같은 요인에 따라 크게 달라집니다.

Amazon Rekognition Custom Labels를 사용하면 이 프로세스가 더 간단해지며 기계 학습 전문 지식이 필요하지 않습니다. 하지만 좋은 모델을 구축하는 프로세스에는 원하는 성능을 달성하기 위해 데이터를 반복하고 모델을 개선해야 하는 경우가 많습니다. 다음은 모델을 개선하는 방법에 대한 정보입니다.

데이터

일반적으로 품질이 더 좋은 데이터를 대량으로 확보하여 모델 품질을 개선할 수 있습니다. 객체나 장면을 명확하게 보여주고 불필요한 물건이 쌓여 있지 않은 훈련 이미지를 사용하세요. 객체 주위의 경계 상자의 경우, 객체가 다른 물체에 가려지지 않고 완전히 보이는 훈련 이미지를 사용하세요.

훈련 및 테스트 데이터 세트가 추론을 실행할 대상 이미지 유형과 일치하는지 확인하세요. 로고와 같이 훈련 예제가 몇 개 없는 객체의 경우 테스트 이미지의 로고 주위에 경계 상자를 제공해야 합니다. 이 이미지는 객체의 위치를 파악하려는 시나리오를 나타내거나 묘사합니다.

훈련 또는 테스트 데이터 세트에 더 많은 이미지를 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요.

거짓 긍정 감소(정밀도 향상)

- 먼저, 추정 임계값을 높였을 때 정확한 예측을 유지하면서 거짓 긍정이 줄어드는지 확인하세요. 특정 모델의 정밀도와 재현율 간의 대립성 때문에 어느 시점부터는 효과가 줄어듭니다. 레이블에 대해 추정 임계값을 설정할 수는 없지만 `MinConfidence` 입력 파라미터에 대해 높은 값을 `DetectCustomLabels`로 지정하여 동일한 결과를 얻을 수 있습니다. 자세한 내용은 훈련된 모델을 사용한 이미지 분석 섹션을 참조하세요.
- 관심 있는 사용자 지정 레이블(A) 중 하나 이상이 동일한 분류의 객체(관심 있는 레이블은 아님)(B)와 계속 혼동되는 상황이 나타날 수 있습니다. 이를 해결하려면 B를 거짓 긍정이 나온 이미지와 함께 훈련 데이터 세트에 객체 분류 레이블로 추가하세요. 이는 모델이 새로운 훈련 이미지를 통해 A가 아닌 B를 예측하도록 학습하도록 돕는 것입니다. 훈련 데이터 세트에 이미지를 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요.
- 두 개의 사용자 지정 레이블(A와 B)로 인해 모델이 혼란을 겪을 수 있습니다. 레이블 A가 있는 테스트 이미지는 레이블 B가 있는 것으로 예측되며 그 반대의 경우도 마찬가지입니다. 이 경우 먼저 훈련 세트와 테스트 세트에서 레이블이 잘못 지정된 이미지가 없는지 확인하세요. 데이터 세트 갤러리를 사용하여 데이터 세트에 할당된 레이블을 관리하세요. 자세한 내용은 레이블 관리 섹션을 참조하세요. 또한 이러한 유형의 혼란과 관련된 훈련 이미지를 더 추가하면 재훈련된 모델이 A와 B를 더 잘 구별하는 데 도움이 됩니다. 훈련 데이터 세트에 이미지를 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요.

거짓 긍정 감소(기억력 향상)

- 추정 임계값에는 더 낮은 값을 사용하세요. 레이블에 대해 추정 임계값을 설정할 수는 없지만 더 낮은 `MinConfidence` 입력 파라미터를 `DetectCustomLabels`로 지정하여 동일한 결과를 얻을 수 있습니다. 자세한 내용은 훈련된 모델을 사용한 이미지 분석 섹션을 참조하세요.
- 더 나은 예제를 사용하여 객체와 객체가 나타나는 이미지의 다양성을 모델링하세요.
- 레이블을 학습하기 쉬운 두 개의 분류로 나누세요. 예를 들어, 모델이 각각의 고유한 개념을 더 잘 학습하는 데 도움이 되도록 좋은 쿠키와 나쁜 쿠키 대신 좋은 쿠키, 탄 쿠키, 깨진 쿠키를 제공할 수 있습니다.

훈련된 Amazon Rekognition Custom Labels 모델 실행

모델의 성능이 만족스러우면 사용을 시작할 수 있습니다. 콘솔 또는 AWS SDK를 사용하여 모델을 시작하고 중지할 수 있습니다. 콘솔에는 사용할 수 있는 예제 SDK 작업도 포함되어 있습니다.

주제

추론 단위

모델을 시작할 때 모델에서 사용하는 컴퓨팅 리소스(추론 단위라고도 함)의 수를 지정합니다.

> **Important**
>
> 모델 실행 구성 방법에 따라 모델이 실행되는 시간 및 모델이 실행되는 동안 사용하는 추론 단위 수에 대한 요금이 부과됩니다. 예를 들어 두 개의 추론 단위로 모델을 시작하고 8시간 동안 모델을 사용하면 추론 시간 16시간 (실행 시간 8시간* 추론 단위 2개) 에 대한 요금이 부과됩니다. 자세한 정보는 추론 시간을 참조하세요. 명시적으로 모델을 중지하지 않으면 모델로 이미지를 적극적으로 분석하지 않더라도 요금이 부과됩니다.

단일 추론 단위가 지원하는 초당 트랜잭션 수(TPS)는 다음의 영향을 받습니다.

- 이미지 수준 레이블(분류)을 탐지하는 모델은 일반적으로 객체를 감지하고 경계 상자로 객체의 위치를 파악하는 모델(객체 감지)보다 TPS가 높습니다.
- 모델의 복잡성
- 해상도가 높은 이미지는 분석에 더 많은 시간이 필요합니다.
- 이미지에 객체가 많을수록 분석하는 데 더 많은 시간이 필요합니다.
- 작은 이미지는 큰 이미지보다 빠르게 분석됩니다.

- 이미지 바이트로 전달된 이미지는 먼저 Amazon S3 버킷에 이미지를 업로드한 다음 업로드된 이미지를 참조하는 것보다 빠르게 분석됩니다. 이미지 바이트로 전달된 이미지는 4.0MB 미만이어야 합니다. 이미지를 거의 실시간으로 처리하고 이미지 크기가 4.0MB 미만인 경우에는 이미지 바이트를 사용하는 것이 좋습니다. IP 카메라에서 캡처한 이미지를 예로 들 수 있습니다.
- Amazon S3 버킷에 저장된 이미지를 처리하는 것이 이미지를 다운로드하고 이미지 바이트로 변환한 다음 분석을 위해 이미지 바이트를 전달하는 것보다 빠릅니다.
- Amazon S3 버킷에 이미 저장된 이미지를 분석하는 것이 이미지 바이트로 전달된 동일한 이미지를 분석하는 것보다 더 빠를 수 있습니다. 이미지 크기가 더 큰 경우 특히 그렇습니다.

`DetectCustomLabels`에 대한 직접 호출 수가 모델에서 사용하는 추론 단위의 합계가 지원하는 최대 TPS를 초과하는 경우 Amazon Rekognition Custom Labels는 `ProvisionedThroughputExceededException` 예외를 반환합니다.

추론 단위를 사용한 처리량 관리

애플리케이션의 요구에 따라 모델의 처리량을 늘리거나 줄일 수 있습니다. 처리량을 늘리려면 추가 추론 단위를 사용하십시오. 추론 단위가 추가될 때마다 처리 속도가 추론 단위당 1씩 빨라집니다. 필요한 추론 단위 수를 계산하는 방법에 대한 자세한 내용은 Amazon Rekognition Custom Labels 및 Amazon Lookout for Vision 모델의 추론 단위 계산을 참조하세요. 모델의 지원되는 처리량을 변경하려면 다음 두 가지 옵션이 있습니다.

수동으로 추론 유닛을 추가하거나 제거합니다.

모델을 중지한 다음 필요한 수의 추론 단위로 다시 시작합니다. 이 접근 방식의 단점은 모델을 다시 시작하는 동안 요청을 받을 수 없고 급증하는 수요를 처리하는 데 사용할 수 없다는 것입니다. 모델의 처리량이 안정적이고 사용 사례가 10~20분의 가동 중지 시간을 견딜 수 있는 경우 이 접근 방식을 사용하세요. 예를 들어 주간 일정을 사용하여 모델과의 통화를 일괄 처리하려는 경우를 들 수 있습니다.

추론 단위 자동 규모 조정

모델이 수요 급증을 수용해야 하는 경우 Amazon Rekognition Custom Labels는 모델에서 사용하는 추론 단위 수를 자동으로 규모 조정할 수 있습니다. 수요가 증가하면 Amazon Rekognition Custom Labels는 모델에 추론 단위를 추가하고 수요가 감소하면 추론 단위를 제거합니다.

Amazon Rekognition Custom Labels가 자동으로 모델의 추론 단위를 규모 조정하도록 하려면 모델을 시작하고 `MaxInferenceUnits` 파라미터를 사용하여 사용할 수 있는 최대 추론 단위 수를 설정하세요. 최대 추론 단위 수를 설정하면 사용 가능한 추론 단위 수를 제한하여 모델 실행 비용을 관리할 수 있

습니다. 최대 단위 수를 지정하지 않는 경우 Amazon Rekognition Custom Labels는 처음에 사용한 추론 단위 수만 사용하여 모델을 자동으로 규모 조정하지 않습니다. 최대 추론 단위 수에 대한 자세한 내용은 Service Quotas를 참조하세요.

`MinInferenceUnits` 파라미터를 사용하여 최소 추론 단위 수를 지정할 수도 있습니다. 이를 통해 모델의 최소 처리량을 지정할 수 있습니다. 여기서 단일 추론 단위는 1시간의 처리 시간을 나타냅니다.

> **Note**
>
> Amazon Rekognition Custom Labels 콘솔에서는 최대 추론 단위 수를 설정할 수 없습니다. 대신 `StartProjectVersion` 작업에 `MaxInferenceUnits` 입력 파라미터를 지정할 수 있습니다.

Amazon Rekognition 사용자 지정 레이블은 모델의 현재 자동 조정 상태를 확인하는 데 사용할 수 있는 다음과 같은 CloudWatch Amazon Logs 지표를 제공합니다.

지표	설명
`DesiredInferenceUnits`	Amazon Rekognition Custom Labels가 스케일 업 또는 스케일 다운되는 추론 단위의 수입니다.
`InServiceInferenceUnits`	모델이 사용하는 추론 단위의 수.

`DesiredInferenceUnits` = `InServiceInferenceUnits`인 경우 Amazon Rekognition Custom Labels는 현재 추론 단위 수를 규모 조정하지 않습니다.

`DesiredInferenceUnits` > `InServiceInferenceUnits`인 경우 Amazon Rekognition Custom Labels는 `DesiredInferenceUnits`의 값까지 스케일 업됩니다.

`DesiredInferenceUnits` < `InServiceInferenceUnits`인 경우 Amazon Rekognition Custom Labels는 `DesiredInferenceUnits`의 값으로 스케일 다운됩니다.

Amazon Rekognition 사용자 지정 레이블에서 반환되는 지표 및 필터링 차원에 대한 자세한 내용은 Rekognition의 지표를 참조하십시오. CloudWatch

모델에 대해 요청한 최대 추론 단위 수를 확인하려면 `DescribeProjectsVersion`을 직접 호출하고 응답에서 `MaxInferenceUnits` 필드를 확인하세요. 예제 코드는 모델 설명(SDK) 항목을 참조하세요.

가용 영역

Amazon Rekognition Custom Labels는 하나의 AWS 리전 내 여러 가용 영역에 추론 단위를 배포하여 가용성을 높입니다. 자세한 내용은 가용 영역을 참조하세요. 가용 영역 중단 및 추론 단위 장애로부터 프로덕션 모델을 보호하려면 최소 두 개의 추론 단위로 프로덕션 모델을 시작하십시오.

가용 영역이 중단되면 가용 영역의 모든 추론 유닛을 사용할 수 없게 되고 모델 용량이 감소합니다. DetectCustom레이블 호출은 나머지 추론 단위에 재분배됩니다. 이러한 직접 호출은 나머지 추론 단위의 지원되는 초당 트랜잭션 수(TPS)를 초과하지 않으면 성공합니다. AWS가 가용 영역을 복구하면 추론 유닛이 다시 시작되고 전체 용량이 복원됩니다.

단일 추론 단위에 장애가 발생하는 경우 Amazon Rekognition Custom Labels는 동일한 가용 영역에서 새 추론 단위를 자동으로 시작합니다. 새 추론 단위가 시작되기 전까지는 모델 용량이 줄어듭니다.

Amazon Rekognition Custom Labels 모델 시작

콘솔을 사용하거나 버전 작업을 사용하여 Amazon Rekognition 사용자 지정 레이블 모델 실행을 시작할 수 있습니다. StartProject

> **Important**
>
> 모델이 실행되는 시간 수와 모델이 실행되는 동안 사용하는 추론 단위 수에 따라 요금이 부과됩니다. 자세한 정보는 훈련된 Amazon Rekognition Custom Labels 모델 실행을 참조하세요.

모델을 시작하는 데 몇 분 정도 걸릴 수 있습니다. 모델 준비 상태의 현재 상태를 확인하려면 프로젝트의 세부 정보 페이지를 확인하거나 버전을 사용하십시오. DescribeProject

모델이 시작된 후에는 DetectCustom레이블을 사용하여 모델을 사용하여 이미지를 분석합니다. 자세한 정보는 훈련된 모델을 사용한 이미지 분석을 참조하세요. 콘솔은 `DetectCustomLabels`를 직접 호출할 예제 코드도 제공합니다.

주제

- Amazon Rekognition Custom Labels 모델 시작(콘솔)
- Amazon Rekognition Custom Labels 모델 시작(SDK)

Amazon Rekognition Custom Labels 모델 시작(콘솔)

다음 절차를 사용하여 콘솔에서 Amazon Rekognition Custom Labels 모델을 실행하세요. 콘솔에서 직접 모델을 시작하거나 콘솔에서 제공하는 AWS SDK 코드를 사용할 수 있습니다.

모델을 시작하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 리소스 페이지에서 시작하려는 학습된 모델이 포함된 프로젝트를 선택합니다.
6. 모델 항목에서 시작할 모델을 선택합니다.
7. 모델 사용 탭을 선택합니다.
8. 다음 중 하나를 수행하십시오.

   Start model using the console

   모델 시작 또는 중지 항목에서 다음을 수행하세요.

   1. 사용할 추론 단위 수를 선택합니다. 자세한 정보는 훈련된 Amazon Rekognition Custom Labels 모델 실행을 참조하세요.
   2. 시작을 선택합니다.
   3. 모델 시작 대화 상자에서 시작을 선택합니다.

   Start model using the AWS SDK

   모델 사용 항목에서 다음을 수행하세요.

   1. API 코드를 선택합니다.
   2. AWS CLI 또는 Python을 선택합니다.
   3. 시작 모델에서 예제 코드를 복사합니다.
   4. 예제 코드를 사용하여 모델을 시작합니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 시작(SDK)을 참조하세요.
9. 프로젝트 개요 페이지로 돌아가려면 페이지 상단에서 프로젝트 이름을 선택합니다.

10. 모델 항목에서 모델의 상태를 확인합니다. 모델 상태가 실행 중이면 모델을 사용하여 이미지를 분석할 수 있습니다. 자세한 정보는 훈련된 모델을 사용한 이미지 분석을 참조하세요.

Amazon Rekognition Custom Labels 모델 시작(SDK)

StartProject버전 API를 호출하고 ProjectVersionArn 입력 파라미터에 모델의 Amazon 리소스 이름 (ARN) 을 전달하여 모델을 시작합니다. 또한 사용할 추론 단위 수를 지정합니다. 자세한 정보는 훈련된 Amazon Rekognition Custom Labels 모델 실행을 참조하세요.

모델을 시작하는 데 시간이 걸릴 수 있습니다. 이 항목의 Python 및 Java 예제에서는 웨이터를 사용하여 모델이 시작되기를 기다립니다. Waiter는 특정 상태에 대해 폴링되는 유틸리티 메서드입니다. 또는 Versions를 DescribeProject 호출하여 현재 상태를 확인할 수도 있습니다.

모델을 시작하려면(SDK)

1. 아직 설치하지 않았다면 및 AWS SDK를 설치하고 구성하십시오. AWS CLI 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 다음 예제 코드를 사용하여 모델을 시작합니다.

   CLI

   project-version-arn의 값을 시작하려는 모델의 ARN으로 변경합니다. --min-inference-units의 값을 사용하려는 추론 단위 수로 변경합니다. 선택적으로 --max-inference-units를 Amazon Rekognition Custom Labels가 모델을 자동으로 규모 조정하는 데 사용할 수 있는 추론 단위의 최대 개수로 변경할 수 있습니다.

   ```
   aws rekognition start-project-version  --project-version-arn model_arn \
      --min-inference-units minimum number of units \
      --max-inference-units maximum number of units \
      --profile custom-labels-access
   ```

   Python

   다음 명령줄 파라미터를 제공하세요.

   - project_arn: 시작하려는 모델이 포함된 프로젝트의 ARN
   - model_arn: 시작하려는 모델의 ARN
   - min_inference_units: 사용하려는 추론 단위의 개수

- (선택 사항) `--max_inference_units` Amazon Rekognition Custom Labels가 모델을 자동 규모 조정하는 데 사용할 수 있는 최대 추론 단위 수입니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to start running an Amazon Lookout for Vision model.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    :return: The model status
    """

    logger.info("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]

    models = rek_client.describe_project_versions(ProjectArn=project_arn,
                                                  VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:

        logger.info("Status: %s", model['StatusMessage'])
        return model["Status"]

    error_message = f"Model {model_arn} not found."
```

```
    logger.exception(error_message)
    raise Exception(error_message)


def start_model(rek_client, project_arn, model_arn, min_inference_units,
 max_inference_units=None):
    """
    Starts the hosting of an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that contains the
    model that you want to start hosting.
    :param min_inference_units: The number of inference units to use for
 hosting.
    :param max_inference_units: The number of inference units to use for auto-
scaling
    the model. If not supplied, auto-scaling does not happen.
    """

    try:
        # Start the model
        logger.info(f"Starting model: {model_arn}. Please wait....")

        if max_inference_units is None:
            rek_client.start_project_version(ProjectVersionArn=model_arn,

 MinInferenceUnits=int(min_inference_units))
        else:
            rek_client.start_project_version(ProjectVersionArn=model_arn,
                                             MinInferenceUnits=int(
                                                 min_inference_units),

 MaxInferenceUnits=int(max_inference_units))

        # Wait for the model to be in the running state
        version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
        project_version_running_waiter = rek_client.get_waiter(
            'project_version_running')
        project_version_running_waiter.wait(
            ProjectArn=project_arn, VersionNames=[version_name])

        # Get the running status
        return get_model_status(rek_client, project_arn, model_arn)

    except ClientError as err:
```

```
        logger.exception("Client error: Problem starting model: %s", err)
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains that the model
 you want to start."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to start."
    )
    parser.add_argument(
        "min_inference_units", help="The minimum number of inference units to
 use."
    )
    parser.add_argument(
        "--max_inference_units",  help="The maximum number of inference units to
 use for auto-scaling the model.", required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Start the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status = start_model(rekognition_client,
```

```
                                args.project_arn, args.model_arn,
                                args.min_inference_units,
                                args.max_inference_units)

        print(f"Finished starting model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        error_message = f"Client error: Problem starting model: {err}"
        logger.exception(error_message)
        print(error_message)

    except Exception as err:
        error_message = f"Problem starting model:{err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

다음 명령줄 파라미터를 제공하세요.

- `project_arn`: 시작하려는 모델이 포함된 프로젝트의 ARN
- `model_arn`: 시작하려는 모델의 ARN
- `min_inference_units`: 사용하려는 추론 단위의 개수
- (선택 사항)`max_inference_units`: Amazon Rekognition Custom Labels가 모델을 자동 규모 조정하는 데 사용할 수 있는 최대 추론 단위 수입니다. 값을 지정하지 않으면 자동 규모 조정이 발생하지 않습니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionResponse;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class StartModel {

    public static final Logger logger =
 Logger.getLogger(StartModel.class.getName());



    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void startMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn,
            Integer minInferenceUnits, Integer maxInferenceUnits
            ) throws Exception, RekognitionException {

        try {
```

```
            logger.log(Level.INFO, "Starting model: {0}", modelArn);

            StartProjectVersionRequest startProjectVersionRequest = null;

            if (maxInferenceUnits == null) {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                        .projectVersionArn(modelArn)
                        .minInferenceUnits(minInferenceUnits)
                        .build();
            }
            else {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                            .projectVersionArn(modelArn)
                            .minInferenceUnits(minInferenceUnits)
                            .maxInferenceUnits(maxInferenceUnits)
                            .build();

            }

            StartProjectVersionResponse response =
rekClient.startProjectVersion(startProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString() );


            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);


            // wait until model starts

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                        .versionNames(versionName)
                        .projectArn(projectArn)
                        .build();

            RekognitionWaiter waiter = rekClient.waiter();
```

```
            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionRunning(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                if(projectVersionDescription.status() ==
ProjectVersionStatus.RUNNING) {
                    logger.log(Level.INFO, "Model is running" );

                }
                else {
                    String error = "Model training failed: " +
projectVersionDescription.statusAsString() + " "
                            + projectVersionDescription.statusMessage() + " " +
modelArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);
                }

            }


        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not start model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
        String projectArn = null;
```

```
        Integer minInferenceUnits = null;
        Integer maxInferenceUnits = null;




        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
 <min_inference_units> <max_inference_units>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 model that you want to start. \n\n"
                + "   model_arn - The ARN of the model version that you want to
 start.\n\n"
                + "   min_inference_units - The number of inference units to
 start the model with.\n\n"
                + "   max_inference_units - The maximum number of inference
 units that Custom Labels can use to "
                + "   automatically scale the model. If the value is null,
 automatic scaling doesn't happen.\n\n";

        if (args.length < 3  || args.length >4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];
        minInferenceUnits=Integer.parseInt(args[2]);

        if (args.length == 4) {
            maxInferenceUnits = Integer.parseInt(args[3]);
        }

        try {


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Start the model.
```

```
            startMyModel(rekClient, projectArn, modelArn, minInferenceUnits,
 maxInferenceUnits);


            System.out.println(String.format("Model started: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }



    }

}
```

Amazon Rekognition Custom Labels 모델 중지

콘솔을 사용하거나 버전 작업을 사용하여 Amazon Rekognition 사용자 지정 레이블 모델 실행을 중단할 수 있습니다. StopProject

주제

- Amazon Rekognition Custom Labels 모델 중지(콘솔)
- Amazon Rekognition Custom Labels 모델 중지(SDK)

Amazon Rekognition Custom Labels 모델 중지(콘솔)

다음 절차를 사용하여 콘솔에서 Amazon Rekognition Custom Labels 모델의 실행을 중지합니다. 콘솔에서 직접 모델을 중지하거나 콘솔에서 제공하는 AWS SDK 코드를 사용할 수 있습니다.

모델을 중지하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.

2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 리소스 페이지에서 중지하려는 훈련된 모델이 포함된 프로젝트를 선택합니다.
6. 모델 항목에서 중지하려는 모델을 선택합니다.
7. 모델 사용 탭을 선택합니다.
8. Stop model using the console

    1. 모델 시작 또는 중지 항목에서 중지를 선택합니다.
    2. 모델 중지 대화 상자에서 중지를 입력하여 모델 중지를 확인합니다.
    3. 중지를 선택하여 모델을 중지합니다.

    Stop model using the AWS SDK

    모델 사용 항목에서 다음을 수행하세요.

    1. API 코드를 선택합니다.
    2. AWS CLI 또는 Python을 선택합니다.
    3. 모델 중지에서 예제 코드를 복사합니다.
    4. 예제 코드를 사용하여 모델을 중지하세요. 자세한 정보는 Amazon Rekognition Custom Labels 모델 중지(SDK)을 참조하세요.
9. 페이지 상단에서 프로젝트 이름을 선택하면 프로젝트 개요 페이지로 돌아갑니다.
10. 모델 항목에서 모델의 상태를 확인합니다. 모델 상태가 중지로 표시되면 모델이 중지된 것입니다.

Amazon Rekognition Custom Labels 모델 중지(SDK)

StopProjectVersion API를 호출하고 `ProjectVersionArn` 입력 파라미터에 모델의 Amazon 리소스 이름 (ARN) 을 전달하여 모델을 중지합니다.

모델을 중지하는 데 시간이 걸릴 수 있습니다. 현재 상태를 확인하려면 `DescribeProjectVersions`를 사용하세요.

모델을 중지하려면(SDK)

1. 아직 설치하지 않았다면 및 AWS SDK를 설치하고 구성하십시오. AWS CLI 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 다음 예제 코드를 사용하여 모델 실행을 중지합니다.

   CLI

   `project-version-arn`의 값을 중지하려는 모델 버전의 ARN으로 변경합니다.

   ```
   aws rekognition stop-project-version --project-version-arn "model arn" \
     --profile custom-labels-access
   ```

   Python

   다음 예제는 이미 실행 중인 모델을 중지합니다.

   다음 명령줄 파라미터를 제공하세요.

   - `project_arn`: 중지하려는 모델이 포함된 프로젝트의 ARN
   - `model_arn`: 중지하려는 모델의 ARN

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Shows how to stop a running Amazon Lookout for Vision model.
   """

   import argparse
   import logging
   import time
   import boto3

   from botocore.exceptions import ClientError


   logger = logging.getLogger(__name__)
   ```

```
def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]

    # Get the model status.
    models=rek_client.describe_project_versions(ProjectArn=project_arn,
    VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:
        logger.info("Status: %s",model['StatusMessage'])
        return model["Status"]

    # No model found.
    logger.exception("Model %s not found.", model_arn)
    raise Exception("Model %s not found.", model_arn)


def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn:  The ARN of the model (ProjectVersion) that you want to
 stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])

        # stops when hosting has stopped or failure.
```

```
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
                finished = True
                continue

            error_message = f"Error stopping model. Unexepected state: {status}"
            logger.exception(error_message)
            raise Exception(error_message)

        logger.info("finished. Status %s", status)
        return status

    except ClientError as err:
        logger.exception("Couldn't stop model - %s: %s",
           model_arn,err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )

def main():
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Stop the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

if __name__ == "__main__":
    main()
```

Java V2

다음 명령줄 파라미터를 제공하세요.

- `project_arn`: 중지하려는 모델이 포함된 프로젝트의 ARN
- `model_arn`: 중지하려는 모델의 ARN

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
```

```
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionResponse;


import java.util.logging.Level;
import java.util.logging.Logger;

public class StopModel {

    public static final Logger logger =
 Logger.getLogger(StopModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void stopMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn)
            throws Exception, RekognitionException {

        try {
```

```
            logger.log(Level.INFO, "Stopping {0}", modelArn);

            StopProjectVersionRequest stopProjectVersionRequest =
StopProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            StopProjectVersionResponse response =
rekClient.stopProjectVersion(stopProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString());

            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);

            // wait until model stops

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            boolean stopped = false;

            // Wait until create finishes

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    ProjectVersionStatus status =
projectVersionDescription.status();

                    logger.log(Level.INFO, "stopping model: {0} ", modelArn);
```

```
                    switch (status) {

                    case STOPPED:
                        logger.log(Level.INFO, "Model stopped");
                        stopped = true;
                        break;

                    case STOPPING:
                        Thread.sleep(5000);
                        break;

                    case FAILED:
                        String error = "Model stopping failed: " +
projectVersionDescription.statusAsString() + " "
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, error);
                        throw new Exception(error);

                    default:
                        String unexpectedError = "Unexpected stopping state: "
                                + projectVersionDescription.statusAsString() + "
"
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, unexpectedError);
                        throw new Exception(unexpectedError);
                    }
                }

            } while (stopped == false);

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not stop model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
```

```
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 model that you want to stop. \n\n"
                + "   model_arn - The ARN of the model version that you want to
 stop.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];


        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Stop model
            stopMyModel(rekClient, projectArn, modelArn);

            System.out.println(String.format("Model stopped: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }
```

```
    }

}
```

실행 시간 및 사용된 추론 단위 보고

2022년 8월 이후에 모델을 학습하고 시작한 경우 `InServiceInferenceUnits` Amazon CloudWatch 메트릭을 사용하여 모델을 실행한 시간과 해당 시간 동안 사용된 추론 단위 수를 확인할 수 있습니다.

> **Note**
>
> 한 AWS 지역에 모델이 하나뿐인 경우, 착신 성공 횟수를 추적하여 모델의 실행 시간을 확인할 수도 있습니다. `StartprojectVersion` `StopProjectVersion` CloudWatch 지표에 모델에 대한 정보가 포함되지 않기 때문에 AWS 지역에서 모델을 두 개 이상 실행하는 경우에는 이 방법이 작동하지 않습니다.
>
> 또는 `StartProjectVersion` 및 `StopProjectVersion` (이벤트 기록 `requestParameters` 필드의 모델 ARN 포함) 에 대한 통화를 추적하는 AWS CloudTrail 데 사용할 수 있습니다. CloudTrail 이벤트는 90일로 제한되지만 CloudTrailLake에는 최대 7년까지 이벤트를 저장할 수 있습니다.

다음 절차는 다음에 대한 그래프를 생성합니다.

- 모델이 실행된 시간
- 모델이 사용한 추론 단위 수

과거 최대 15개월의 기간을 선택할 수 있습니다. 지표 보존 기간에 대한 자세한 내용은 지표 보존 기간을 참조하세요.

모델 지속 시간 및 모델에 사용된 추론 단위를 결정하려면

1. https://console.aws.amazon.com/cloudwatch/ 에서 AWS Management Console 로그인하고 CloudWatch 콘솔을 여십시오.
2. 탐색 창의 지표에서 모든 지표를 선택합니다.
3. 콘텐츠 창에서 소스 탭을 선택합니다.
4. 대시보드 버튼이 선택되어 있는지 확인하세요.

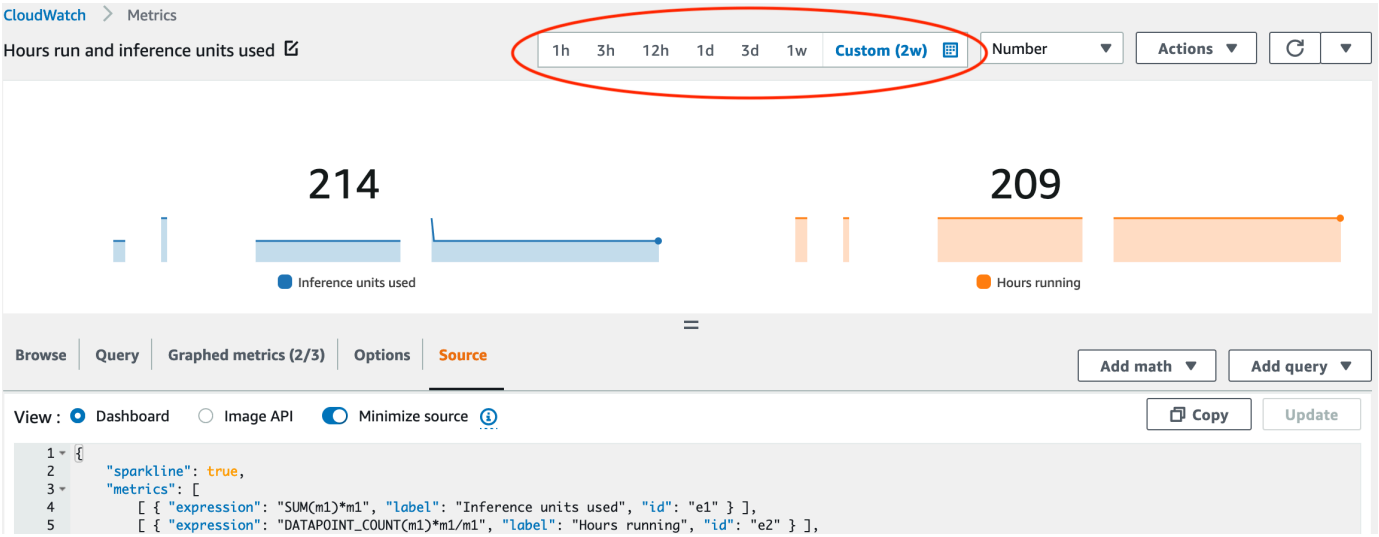
5. 편집 상자에서 기존 JSON을 다음 JSON으로 바꾸세요. 다음 값을 변경합니다.

   - `Project_Name`: 그래프로 표시할 모델이 포함된 프로젝트
   - `Version_Name`: 그래프로 표시할 모델의 버전
   - `AWS_Region`— 모델이 포함된 AWS 지역. 페이지 상단의 내비게이션 바에서 AWS 지역 선택기를 확인하여 CloudWatch 콘솔이 동일한 지역에 있는지 확인하십시오. 필요에 따라 업데이트하세요.

```
{
    "sparkline": true,
    "metrics": [
        [
            {
                "expression": "SUM(m1)*m1",
                "label": "Inference units used",
                "id": "e1"
            }
        ],
        [
            {
                "expression": "DATAPOINT_COUNT(m1)*m1/m1",
                "label": "Hours running",
                "id": "e2"
            }
        ],
        [
            "AWS/Rekognition",
            "InServiceInferenceUnits",
            "ProjectName",
            "Project_Name",
            "VersionName",
            "Version_Name",
            {
                "id": "m1",
                "visible": false
            }
        ]
    ],
    "view": "singleValue",
    "stacked": false,
    "region": "AWS_Region",
```

```
    "stat": "Average",
    "period": 3600,
    "title": "Hours run and inference units used"
}
```

6. 업데이트를 선택합니다.
7. 페이지 상단에서 타임라인을 선택합니다. 타임라인에서 사용된 추론 단위 및 실행 시간에 대한 숫자를 확인할 수 있을 것입니다. 그래프의 간격은 모델이 실행되지 않은 시간을 나타냅니다. 아래 콘솔의 스크린샷은 일정 기간 동안 사용된 추론 단위와 실행 시간을 보여줍니다. 사용자 지정 시간은 2주로 설정되어 있고 최고값은 214개의 추론 단위와 209시간의 실행 시간으로 설정되어 있습니다.



8. (선택 사항) 대시보드에 그래프를 추가하려면 작업을 선택한 후 대시보드에 추가 - 개선됨을 선택합니다.

훈련된 모델을 사용한 이미지 분석

학습된 Amazon Rekognition 사용자 지정 레이블 모델로 이미지를 분석하려면 레이블 API를 호출합니다. DetectCustom `DetectCustomLabels`의 결과는 이미지에 특정 객체, 장면 또는 개념이 포함되어 있다는 예측입니다.

`DetectCustomLabels`를 직접 호출하려면 다음을 지정해야 합니다.

- 사용할 Amazon Rekognition Custom Labels 모델의 Amazon 리소스 이름(ARN)
- 모델이 예측에 사용할 이미지 입력 이미지를 이미지 바이트 배열(base64 인코딩 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. 자세한 정보는 이미지를 참조하세요.

사용자 지정 레이블은 Custom Label 객체의 배열로 반환됩니다. 각 사용자 지정 레이블은 이미지에 있는 단일 객체, 장면 또는 개념을 나타냅니다. 사용자 지정 레이블에는 다음이 포함됩니다.

- 이미지에 있는 객체, 장면 또는 개념의 레이블.
- 이미지에 있는 객체의 경계 상자. 경계 상자 좌표는 원본 이미지에서 객체가 어디 있는지를 나타냅니다. 좌표 값은 전체 이미지 크기의 비율입니다. 자세한 내용은 을 참조하십시오. BoundingBox `DetectCustomLabels`모델이 객체 위치를 감지하도록 훈련된 경우에만 경계 상자를 반환합니다.
- Amazon Rekognition Custom Labels가 레이블과 경계 상자의 정확성에 대해 갖고 있는 신뢰도.

탐지 신뢰도를 기준으로 레이블을 필터링하려면 `MinConfidence`에 원하는 신뢰 수준을 값으로 지정하세요. 예를 들어 예측에 대한 높은 신뢰도가 필요한 경우 `MinConfidence`에 높은 값을 지정하세요. 신뢰도에 관계없이 모든 레이블을 가져오려면 `MinConfidence` 값을 0으로 지정하세요.

모델의 성능은 부분적으로 모델 훈련 중에 계산된 재현율 및 정밀도 지표에 의해 측정됩니다. 자세한 정보는 모델 평가를 위한 지표을 참조하세요.

모델의 정밀도를 높이려면 `MinConfidence`의 값을 더 높게 설정하세요. 자세한 정보는 거짓 긍정 감소(정밀도 향상)을 참조하세요.

모델의 재현율을 높이려면 `MinConfidence`에 대해 더 낮은 값을 사용하세요. 자세한 정보는 거짓 긍정 감소(기억력 향상)을 참조하세요.

`MinConfidence`에 값을 지정하지 않으면 Amazon Rekognition Custom Labels는 해당 레이블에 대해 가정된 임계값을 기반으로 레이블을 반환합니다. 자세한 정보는 추정 임계값을 참조하세요. 모델의 훈

련 결과에서 레이블에 대한 추정 임계값 값을 가져올 수 있습니다. 자세한 정보는 모델 훈련(콘솔)을 참조하세요.

MinConfidence 입력 파라미터를 사용하면 직접 호출에 대해 원하는 임계값을 지정할 수 있습니다. 신뢰도가 MinConfidence의 값 미만인 것으로 감지된 레이블은 응답으로 반환되지 않습니다. 또한 레이블에 대해 가정된 임계값은 응답에 레이블이 포함되는 데 영향을 주지 않습니다.

> **Note**
>
> Amazon Rekognition Custom Labels 지표는 가정된 임계값을 0-1 사이의 부동 소수점 값으로 표시합니다. MinConfidence의 범위는 임계값을 백분율 값(0-100)으로 정규화합니다. 의 DetectCustomLabels 신뢰도 응답도 백분율로 반환됩니다.

특정 레이블의 임계값을 지정할 수도 있습니다. 예를 들어, 정밀도 측정치를 레이블 A에는 사용할 수 있지만 레이블 B에는 사용할 수 없는 경우가 있습니다. 다른 임계값(MinConfidence)을 지정할 때는 다음 사항을 고려하세요.

- 단일 레이블(A)에만 관심이 있는 경우 MinConfidence의 값을 원하는 임계값으로 설정하세요. 응답에는 신뢰도가 MinConfidence보다 큰 경우에만 레이블 A에 대한 예측이 다른 레이블과 함께 반환됩니다. 반환되는 다른 레이블은 모두 필터링해야 합니다.
- 여러 레이블에 서로 다른 임계값을 적용하려면 다음을 수행하세요.
  1. MinConfidence의 값은 0을 사용하세요. 값이 0이면 탐지 신뢰도에 관계없이 모든 레이블이 반환됩니다.
  2. 반환된 각 레이블에 대해 레이블 신뢰도가 레이블에 대해 원하는 임계값보다 큰지 확인하여 원하는 임계값을 적용합니다.

자세한 정보는 훈련된 Amazon Rekognition Custom Labels 모델 개선을 참조하세요.

DetectCustomLabels에서 반환된 신뢰도 값이 너무 낮으면 모델을 다시 훈련하는 것을 고려하세요. 자세한 정보는 Amazon Rekognition Custom Labels 모델 훈련을 참조하세요. MaxResults 입력 파라미터를 지정하여 DetectCustomLabels에서 반환되는 사용자 지정 레이블의 수를 제한할 수 있습니다. 결과는 가장 높은 신뢰도에서 가장 낮은 신뢰도 순으로 정렬되어 반환됩니다.

DetectCustomLabels를 직접 호출하는 다른 예제를 보려면 예제 항목을 참조하세요.

DetectCustomLabels 확보에 대한 자세한 내용은 DetectCustomLabels 보호 항목을 참조하세요.

사용자 지정 레이블(API) 감지하기

1. 아직 하지 않았다면 다음을 수행하세요.

   a. `DetectCustomLabels` 및 `AmazonS3ReadOnlyAccess`의 권한이 있는지 확인하세요. 자세한 정보는 SDK 권한 설정을 참조하세요.

   b. 및 AWS SDK를 설치 AWS CLI 및 구성합니다. 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.

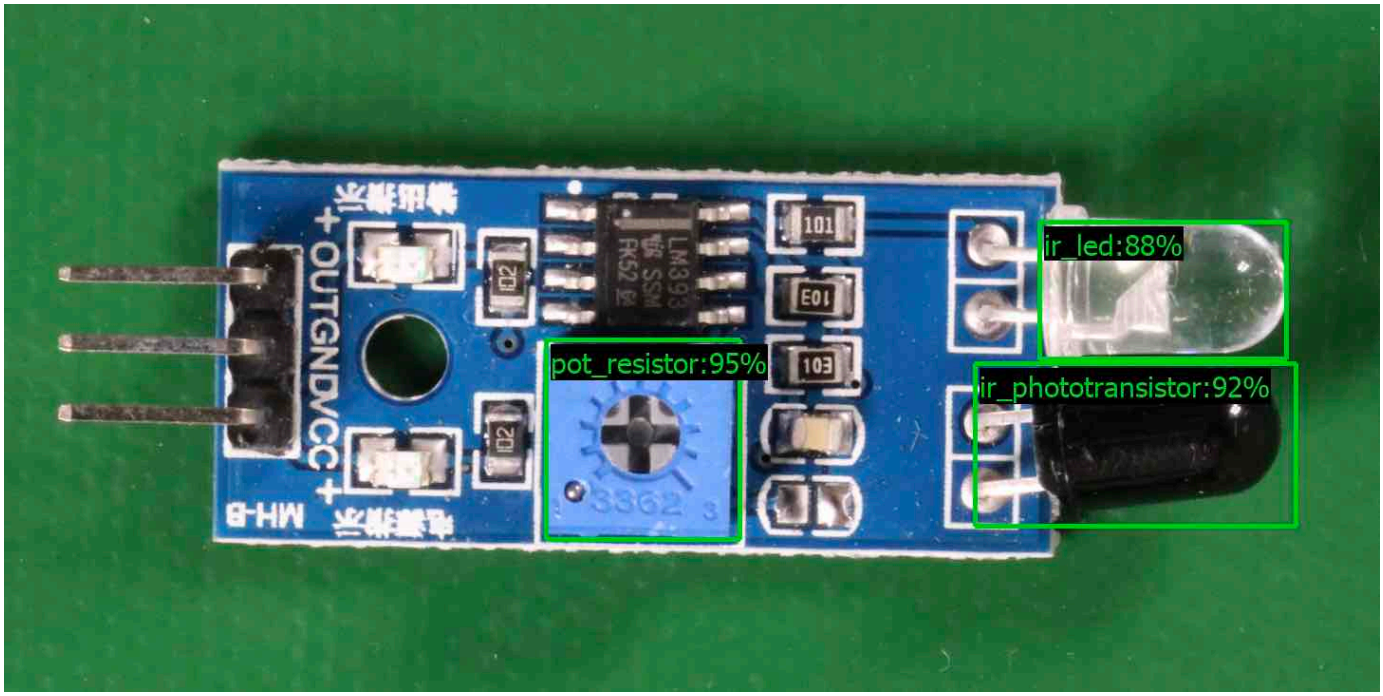
2. 모델을 훈련하고 배포하세요. 자세한 정보는 Amazon Rekognition Custom Labels 모델 생성을 참조하세요.

3. `DetectCustomLabels`를 직접 호출한 사용자가 2단계에서 사용한 모델에 액세스할 수 있는지 확인하세요. 자세한 정보는 DetectCustomLabels 보호을 참조하세요.

4. 분석할 이미지를 S3 버킷에 업로드합니다.

   이에 관한 지침은 Amazon Simple Storage Service 사용 설명서에서 Amazon S3에 객체 업로드를 참조하세요. Python, Java, Java 2 예제는 로컬 이미지 파일을 사용하여 원시 바이트를 사용하여 이미지를 전달하는 방법도 보여줍니다. 파일은 4MB보다 작아야 합니다.

5. 다음 예제를 사용하여 `DetectCustomLabels` 작업을 호출합니다. Python 및 Java 예제는 다음 이미지와 마찬가지로 이미지를 표시하고 분석 결과를 오버레이합니다. 다음 이미지에는 전위차계, 적외선 포토트랜지스터 및 LED 부품이 있는 회로 기판의 바운딩 박스와 레이블이 들어 있습니다.



AWS CLI

이 AWS CLI 명령은 DetectCustomLabels CLI 작업에 대한 JSON 출력을 표시합니다. 다음 입력 파라미터의 값을 변경하세요.

- bucket을 4단계에서 사용한 Amazon S3 버킷의 이름으로 변경하세요.
- image를 4단계에서 업로드한 입력 이미지 파일의 이름으로 변경하세요.
- projectVersionArn을 사용할 모델의 ARN으로 변경하세요.

```
aws rekognition detect-custom-labels --project-version-arn model_arn \
   --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \
   --min-confidence 70 \
   --profile custom-labels-access
```

Python

다음 예제 코드에는 이미지에 있는 경계 상자와 이미지 레벨 레이블이 표시되어 있습니다.

로컬 이미지를 분석하려면 프로그램을 실행하고 다음 명령줄 인수를 제공하세요.

- 이미지를 분석할 모델의 ARN.
- 로컬 이미지 파일의 이름 및 위치.

Amazon S3 버킷에 저장된 이미지를 분석하려면 프로그램을 실행하고 다음 명령줄 인수를 제공하세요.

- 이미지를 분석할 모델의 ARN.
- 4단계에서 사용한 Amazon S3 버킷 내 이미지의 이름 및 위치.
- --bucket ## ##: 4단계에서 사용한 Amazon S3 버킷.

참고로 이 예제에서는 현재 사용 중인 Pillow 버전이 8.0.0보다 크다고 가정합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Amazon Rekognition Custom Labels detection example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/detecting-custom-
labels.html
Shows how to detect custom labels by using an Amazon Rekognition Custom Labels
 model.
The image can be stored on your local computer or in an Amazon S3 bucket.
"""

import io
import logging
import argparse
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def analyze_local_image(rek_client, model, photo, min_confidence):
    """
    Analyzes an image stored as a local file.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param photo: The name and file path of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        logger.info("Analyzing local file: %s", photo)
        image = Image.open(photo)
        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}"
            )

        # get images bytes for call to detect_anomalies
```

```
        image_bytes = io.BytesIO()
        image.save(image_bytes, format=image.format)
        image_bytes = image_bytes.getvalue()

        response = rek_client.detect_custom_labels(Image={'Bytes': image_bytes},
                                                   MinConfidence=min_confidence,
                                                   ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as client_err:
        logger.error(format(client_err))
        raise
    except FileNotFoundError as file_error:
        logger.error(format(file_error))
        raise


def analyze_s3_image(rek_client, s3_connection, model, bucket, photo,
 min_confidence):
    """
    Analyzes an image stored in the specified S3 bucket.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param bucket: The name of the S3 bucket that contains the image that you
 want to analyze.
    :param photo: The name of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        # Get image from S3 bucket.

        logger.info("analyzing bucket: %s image: %s", bucket, photo)
        s3_object = s3_connection.Object(bucket, photo)
        s3_response = s3_object.get()

        stream = io.BytesIO(s3_response['Body'].read())
        image = Image.open(stream)

        image_type = Image.MIME[image.format]
```

```
        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}")

        ImageDraw.Draw(image)

        # Call DetectCustomLabels.
        response = rek_client.detect_custom_labels(
            Image={'S3Object': {'Bucket': bucket, 'Name': photo}},
            MinConfidence=min_confidence,
            ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as err:
        logger.error(format(err))
        raise


def show_image(image, response):
    """
    Displays the analyzed image and overlays analysis results
    :param image: The analyzed image
    :param response: the response from DetectCustomLabels
    """
    try:
        font_size = 40
        line_width = 5

        img_width, img_height = image.size
        draw = ImageDraw.Draw(image)

        # Calculate and display bounding boxes for each detected custom label.
        image_level_label_height = 0

        for custom_label in response['CustomLabels']:
            confidence = int(round(custom_label['Confidence'], 0))
            label_text = f"{custom_label['Name']}:{confidence}%"
            fnt = ImageFont.truetype('Tahoma.ttf', font_size)
```

```
            text_left, text_top, text_right, text_bottom = draw.textbbox((0, 0),
 label_text, fnt)
            text_width, text_height = text_right - text_left, text_bottom -
 text_top

            logger.info("Label: %s", custom_label['Name'])
            logger.info("Confidence:  %s", confidence)

            # Draw bounding boxes, if present
            if 'Geometry' in custom_label:
                box = custom_label['Geometry']['BoundingBox']
                left = img_width * box['Left']
                top = img_height * box['Top']
                width = img_width * box['Width']
                height = img_height * box['Height']

                logger.info("Bounding box")
                logger.info("\tLeft: {0:.0f}".format(left))
                logger.info("\tTop: {0:.0f}".format(top))
                logger.info("\tLabel Width: {0:.0f}".format(width))
                logger.info("\tLabel Height: {0:.0f}".format(height))

                points = (
                    (left, top),
                    (left + width, top),
                    (left + width, top + height),
                    (left, top + height),
                    (left, top))
                # Draw bounding box and label text
                draw.line(points, fill="limegreen", width=line_width)
                draw.rectangle([(left + line_width, top+line_width),
                                (left + text_width + line_width, top +
 line_width + text_height)], fill="black")
                draw.text((left + line_width, top + line_width),
                          label_text, fill="limegreen", font=fnt)

            # draw image-level label text.
            else:
                draw.rectangle([(10, image_level_label_height),
                                (text_width + 10, image_level_label_height
+text_height)], fill="black")
                draw.text((10, image_level_label_height),
                          label_text, fill="limegreen", font=fnt)
```

```
                image_level_label_height += text_height

        image.show()

    except Exception as err:
        logger.error(format(err))
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "image", help="The path and file name of the image that you want to
 analyze"
    )
    parser.add_argument(
        "--bucket",  help="The bucket that contains the image. If not supplied,
 image is assumed to be a local file.", required=False
    )


def main():

    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        label_count = 0
        min_confidence = 50

        session = boto3.Session(profile_name='custom-labels-access')
```

```
        rekognition_client = session.client("rekognition")

        if args.bucket is None:
            # Analyze local image.
            label_count = analyze_local_image(rekognition_client,
                                              args.model_arn,
                                              args.image,
                                              min_confidence)
        else:
            # Analyze image in S3 bucket.
            s3_connection = session.resource('s3')
            label_count = analyze_s3_image(rekognition_client,
                                           s3_connection,
                                           args.model_arn,
                                           args.bucket,
                                           args.image,
                                           min_confidence)

        print(f"Custom labels detected: {label_count}")

    except ClientError as client_err:
        print("A service client error occurred: " +
              format(client_err.response["Error"]["Message"]))

    except ValueError as value_err:
        print("A value error occurred: " + format(value_err))

    except FileNotFoundError as file_error:
        print("File not found error: " + format(file_error))

    except Exception as err:
        print("An error occurred: " + format(err))


if __name__ == "__main__":
    main()
```

Java

다음 예제 코드에는 이미지에 있는 경계 상자와 이미지 레벨 레이블이 표시되어 있습니다.

로컬 이미지를 분석하려면 프로그램을 실행하고 다음 명령줄 인수를 제공하세요.

- 이미지를 분석할 모델의 ARN.
- 로컬 이미지 파일의 이름 및 위치.

Amazon S3 버킷에 저장된 이미지를 분석하려면 프로그램을 실행하고 다음 명령줄 인수를 제공하세요.

- 이미지를 분석할 모델의 ARN.
- 4단계에서 사용한 Amazon S3 버킷 내 이미지의 이름 및 위치.
- 4단계에서 사용한 이미지가 포함된 Amazon S3 버킷.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.io.FileNotFoundException;
import java.awt.font.FontRenderContext;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
```

```
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CustomLabel;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.util.IOUtils;

// Calls DetectCustomLabels and displays a bounding box around each detected
 image.
public class DetectCustomLabels extends JPanel {

    private transient DetectCustomLabelsResult response;
    private transient Dimension dimension;
    private transient BufferedImage image;

    public static final Logger logger =
 Logger.getLogger(DetectCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public DetectCustomLabels(AmazonRekognition rekClient,
            AmazonS3 s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws AmazonRekognitionException,
 AmazonS3Exception, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });

        // Get image from S3 bucket and create BufferedImage
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, key);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        image = ImageIO.read(inputStream);
```

```
        // Set image size
        setWindowDimensions();

        DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
                .withProjectVersionArn(projectVersionArn)
                .withImage(new Image().withS3Object(new
S3Object().withName(key).withBucket(bucket)))
                .withMinConfidence(minConfidence);

        // Call DetectCustomLabels

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.getCustomLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public DetectCustomLabels(AmazonRekognition rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, AmazonRekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes and buffered image
        ByteBuffer imageBytes;
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }

        // Get image for display
        InputStream imageBytesStream;
        imageBytesStream = new ByteArrayInputStream(imageBytes.array());

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image = ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);

        // Set image size
        setWindowDimensions();

        // Analyze image
```

```
        DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
                .withProjectVersionArn(projectVersionArn)
                .withImage(new Image()
                        .withBytes(imageBytes))
                .withMinConfidence(minConfidence);

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.getCustomLabels());

        drawLabels();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found");
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
        } else {
            for (CustomLabel customLabel : customLabels) {
                logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                        new Object[] { customLabel.getName(),
customLabel.getConfidence() });
            }

        }
    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }
```

```
        setPreferredSize(dimension);

    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public void drawLabels() {
        // Draws bounding boxes (if present) and label text.

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.getCustomLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.getName();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.getGeometry() != null) {
```

```
                BoundingBox box = customLabel.getGeometry().getBoundingBox();
                float left = imageWidth * box.getLeft();
                float top = imageHeight * box.getTop();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.getWidth())),
                        Math.round((imageHeight * box.getHeight())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    public static void main(String args[]) throws Exception {

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;
        float minConfidence = 50;
```

```
        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        DetectCustomLabels panel = null;

        try {

            AWSCredentialsProvider provider =new
 ProfileCredentialsProvider("custom-labels-access");

            AmazonRekognition rekClient =
 AmazonRekognitionClientBuilder.standard()
                    .withCredentials(provider)
                    .withRegion(Regions.US_WEST_2)
                    .build();


            AmazonS3 s3client = AmazonS3ClientBuilder.standard()
            .withCredentials(provider)
            .withRegion(Regions.US_WEST_2)
            .build();
```

```
            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new DetectCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
                panel = new DetectCustomLabels(rekClient, s3client,
projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (AmazonRekognitionException rekError) {
            String errorMessage = "Rekognition client error: " +
rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (AmazonS3Exception s3Error) {
            String errorMessage = "S3 error: " + s3Error.getErrorMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
```

```
}
```

Java V2

다음 예제 코드에는 이미지에 있는 경계 상자와 이미지 레벨 레이블이 표시되어 있습니다.

로컬 이미지를 분석하려면 프로그램을 실행하고 다음 명령줄 인수를 제공하세요.

- `projectVersionArn`: 이미지를 분석할 모델의 ARN.
- `photo`: 로컬 이미지 파일의 이름 및 위치.

S3 버킷에 저장된 이미지를 분석하려면 프로그램을 실행하고 다음 명령줄 인수를 제공하세요.

- 이미지를 분석할 모델의 ARN.
- 4단계에서 사용한 S3 버킷 내 이미지의 이름 및 위치.
- 4단계에서 사용한 이미지가 포함된 Amazon S3 버킷.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import java.awt.*;
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.util.logging.Level;
import java.util.logging.Logger;

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {

    private transient BufferedImage image;
    private transient DetectCustomLabelsResponse response;
    private transient Dimension dimension;
    public static final Logger logger =
 Logger.getLogger(ShowCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public ShowCustomLabels(RekognitionClient rekClient,
            S3Client s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws RekognitionException,
 NoSuchBucketException, NoSuchKeyException, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });
        // Get image from S3 bucket and create BufferedImage
```

```
        GetObjectRequest requestObject =
GetObjectRequest.builder().bucket(bucket).key(key).build();
        ResponseBytes<GetObjectResponse> result =
s3client.getObject(requestObject, ResponseTransformer.toBytes());
        ByteArrayInputStream bis = new
ByteArrayInputStream(result.asByteArray());
        image = ImageIO.read(bis);

        // Set image size
        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();

        Image s3Image = Image.builder().s3Object(s3Object).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public ShowCustomLabels(RekognitionClient rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, RekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);
        // Get image bytes and buffered image
        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
        ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
        image = ImageIO.read(inputStream);

        setWindowDimensions();
```

```
        // Construct request parameter for DetectCustomLabels
        Image localImageBytes = Image.builder().bytes(imageBytes).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws bounding boxes (if present) and label text.
    public void drawLabels() {

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
```

```
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.customLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.name();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.geometry() != null) {

                BoundingBox box = customLabel.geometry().boundingBox();
                float left = imageWidth * box.left();
                float top = imageHeight * box.top();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                        Math.round((imageHeight * box.height())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
```

```
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found:");
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");

        }
        else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                    new Object[] { customLabel.name(),
customLabel.confidence() } );
            }
        }
    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public static void main(String args[]) throws Exception {

        String photo = null;
```

```
        String bucket = null;
        String projectVersionArn = null;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        float minConfidence = 50;

        ShowCustomLabels panel = null;

        try {
            // Get the Rekognition client


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            S3Client s3Client = S3Client.builder()
```

```
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new ShowCustomLabels(rekClient, projectVersionArn,
 photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
                panel = new ShowCustomLabels(rekClient, s3Client,
 projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (RekognitionException rekError) {

            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchKeyException bucketError) {
```

```
            String errorMessage = String.format("Image not found: %s in bucket
 %s.", photo, bucket);
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchBucketException bucketError) {
            String errorMessage = "Bucket not found: " + bucket;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

DetectCustomLabels 작업 요청

DetectCustomLabels 작업에서 사용자는 입력 이미지를 base64로 인코딩된 바이트 배열 또는 Amazon S3 버킷에 저장된 이미지로 제공합니다. 다음 예제 JSON 요청은 Amazon S3 버킷에서 불러온 이미지를 표시합니다.

```
{
    "ProjectVersionArn": "string",
     "Image":{
        "S3Object":{
            "Bucket":"string",
            "Name":"string",
            "Version":"string"
         }
    },
    "MinConfidence": 90,
    "MaxLabels": 10,
}
```

DetectCustomLabels 운영 응답

DetectCustomLabels 작업의 다음 JSON 응답에는 다음 이미지에서 감지된 사용자 지정 레이블이 표시되어 있습니다.

```
{
```

```
    "CustomLabels": [
        {
            "Name": "MyLogo",
            "Confidence": 77.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

Amazon Rekognition Custom Labels 리소스 관리

이 항목은 Amazon Rekognition Custom Labels 모델을 훈련하고 사용하는 데 사용하는 워크플로의 개요를 제공합니다. 또한 AWS SDK를 사용하여 모델을 훈련하고 사용하는 방법에 대한 개요 정보도 포함되어 있습니다.

Amazon Rekognition Custom Labels 프로젝트 관리

Amazon Rekognition Custom Labels 내에서는 프로젝트를 사용하여 특정 용례에 맞게 생성한 모델을 관리합니다. 프로젝트는 데이터 세트, 모델 훈련, 모델 버전, 모델 평가 및 프로젝트 모델 실행을 관리합니다.

주제

- Amazon Rekognition Custom Labels 프로젝트 삭제
- 프로젝트 설명(SDK)
- AWS CloudFormation으로 프로젝트 생성

Amazon Rekognition Custom Labels 프로젝트 삭제

Amazon Rekognition 콘솔을 사용하거나 API를 호출하여 프로젝트를 삭제할 수 있습니다. DeleteProject 프로젝트를 삭제하려면 먼저 관련 모델을 각각 삭제해야 합니다. 삭제된 프로젝트 또는 모델은 복구할 수 없습니다.

주제

- Amazon Rekognition Custom Labels 프로젝트 삭제(콘솔)
- Amazon Rekognition Custom Labels 프로젝트 삭제(SDK)

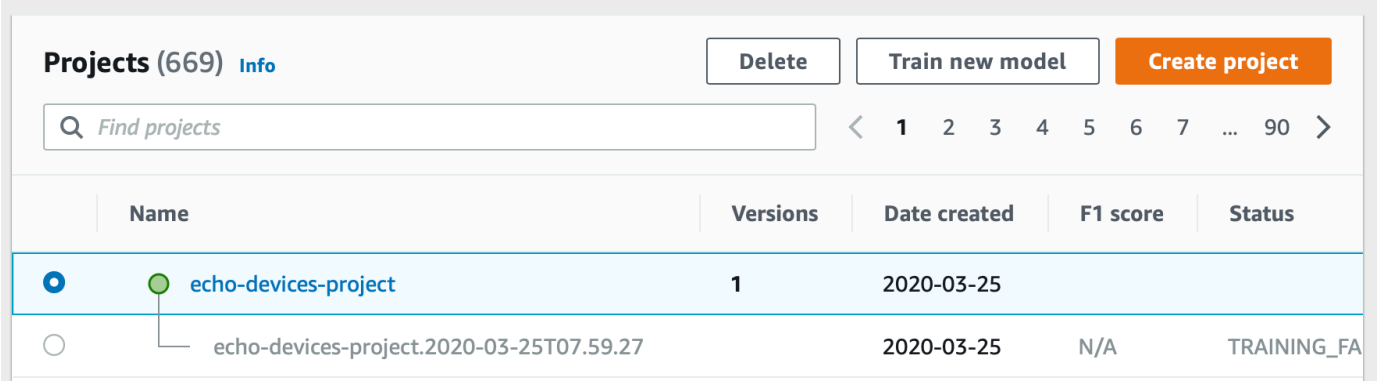
Amazon Rekognition Custom Labels 프로젝트 삭제(콘솔)

프로젝트 페이지 또는 프로젝트의 세부 정보 페이지에서 프로젝트를 삭제할 수 있습니다. 다음 절차는 프로젝트 페이지를 사용하여 프로젝트를 삭제하는 방법을 보여줍니다.

Amazon Rekognition Custom Labels 콘솔은 프로젝트 삭제 중에 관련 모델 및 데이터 세트를 자동으로 삭제합니다. 모델이 실행 중이거나 훈련 중인 경우에는 프로젝트를 삭제할 수 없습니다. 모델 실행을 중지하려면 Amazon Rekognition Custom Labels 모델 중지(SDK) 항목을 참조하세요. 프로젝트가 훈련 중이면 완료될 때까지 기다린 후 프로젝트를 삭제하세요.

프로젝트를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. Get started를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 프로젝트 페이지에서 삭제할 프로젝트의 라디오 버튼을 선택합니다. 2020-03-25에 생성된 버전 1개와 삭제, 새 모델 교육 또는 프로젝트 생성 옵션이 포함된 프로젝트 목록이 표시됩니다 echo-devices-project.

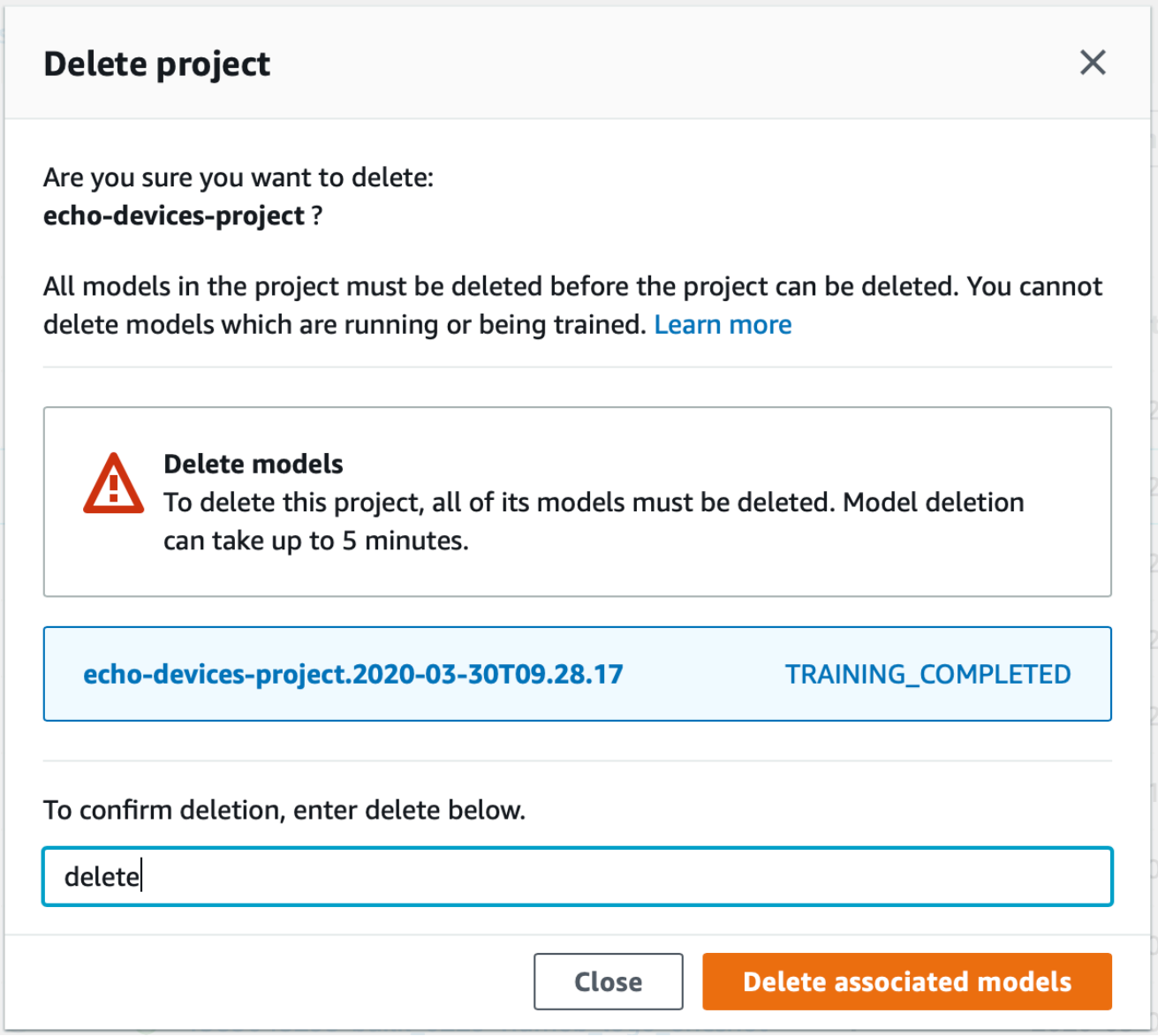


6. 페이지 상단에서 삭제를 선택합니다. 프로젝트 삭제 대화 상자가 표시됩니다.
7. 프로젝트에 연결된 모델이 없는 경우:
   a. 삭제를 입력하여 프로젝트가 삭제됩니다.
   b. 삭제를 선택하여 프로젝트를 삭제하세요.
8. 프로젝트에 연결된 모델 또는 데이터 세트가 있는 경우:
   a. 삭제를 입력하여 모델 및 데이터 세트 삭제를 확인합니다.
   b. 모델에 데이터 세트, 모델 또는 둘 다 있는지 여부에 따라 연결된 모델 삭제 또는 연결된 데이터 세트 삭제 또는 연결된 데이터 세트 및 모델 삭제를 선택합니다. 모델 삭제를 완료하는 데 시간이 걸릴 수 있습니다.

> **Note**
>
> 콘솔은 훈련 중이거나 실행 중인 모델을 삭제할 수 없습니다. 실행 중으로 표시된 모델을 모두 중지한 후 다시 시도하고, 훈련 중으로 표시된 모델의 훈련이 완료될 때까지 기다리세요.

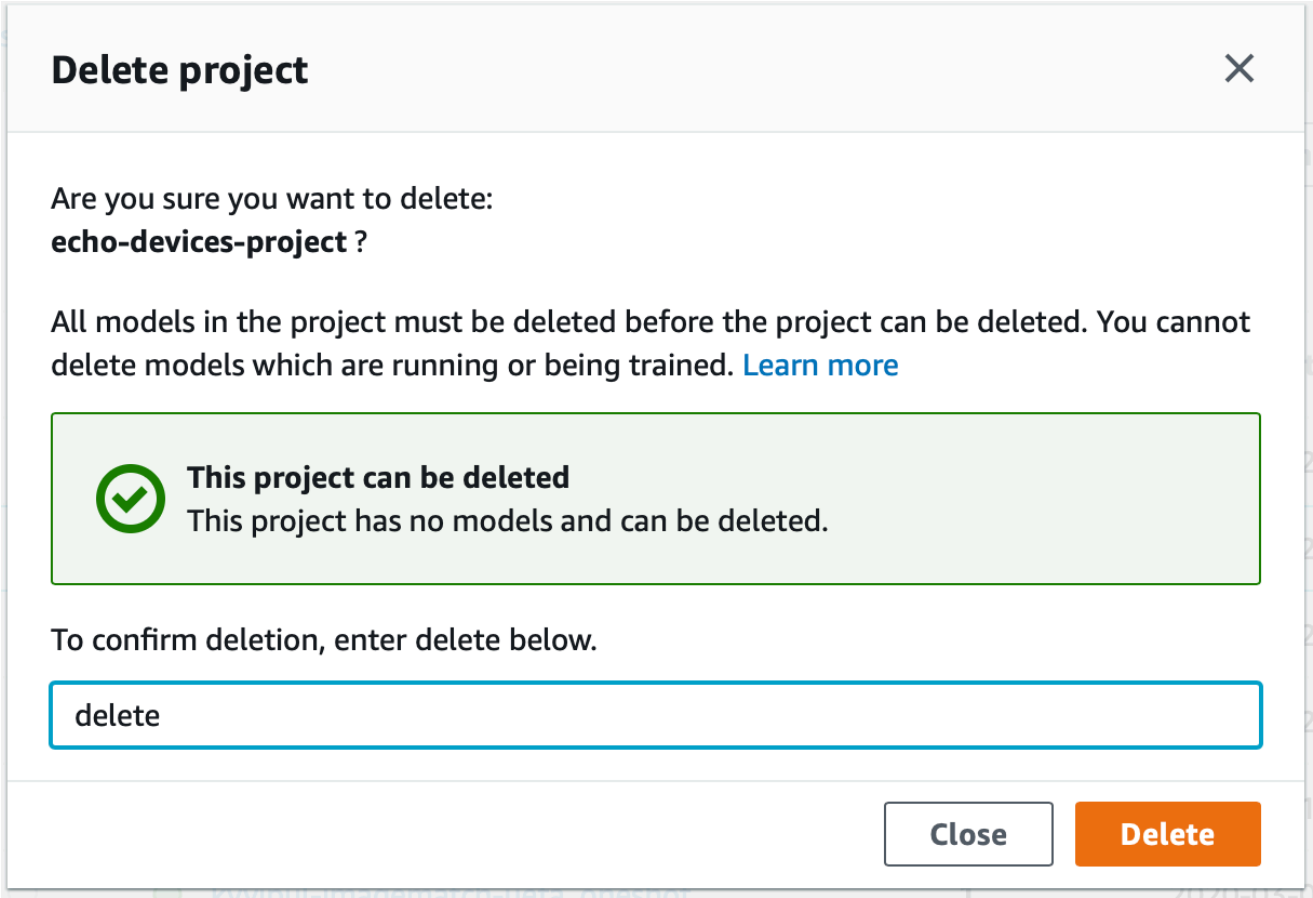
모델 삭제 중에 대화 상자를 닫아도 모델은 삭제됩니다. 나중에 이 절차를 반복하여 프로젝트를 삭제할 수 있습니다.

모델 삭제 패널에는 관련 모델을 삭제하라는 명시적인 지침이 제공됩니다.



c. 삭제를 입력하여 프로젝트 삭제를 확인합니다.

d. 삭제를 선택하여 프로젝트를 삭제하세요.



Amazon Rekognition Custom Labels 프로젝트 삭제(SDK)

삭제하려는 프로젝트의 Amazon 리소스 이름 (ARN) 을 DeleteProject호출하여 제공하여 Amazon Rekognition 사용자 지정 레이블 프로젝트를 삭제합니다. 계정에 있는 프로젝트의 ARN을 가져오려면 전화하십시오. AWS DescribeProjects 응답에는 ProjectDescription객체 배열이 포함됩니다. 프로젝트 ARN은 `ProjectArn` 필드입니다. 프로젝트 이름을 사용하여 프로젝트의 ARN을 식별할 수 있습니다. 예: `arn:aws:rekognition:us-east-1:123456789010:project/`*`project name`*`/1234567890123`

프로젝트를 삭제하려면 먼저 프로젝트의 모든 모델 및 데이터 세트를 삭제해야 합니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 삭제(SDK) 및 데이터 세트 삭제 섹션을 참조하세요.

프로젝트를 삭제하는 데 다소 시간이 걸릴 수 있습니다. 이 기간 동안의 프로젝트 상태는 `DELETING`입니다. 후속 호출에 삭제한 프로젝트가 DescribeProjects포함되지 않으면 프로젝트가 삭제됩니다.

프로젝트 (SDK)를 삭제하려면

1. 아직 설치하지 않았다면 및 AWS SDK를 설치하고 구성하세요. AWS CLI 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 프로젝트를 삭제하려면 다음 절차에 따르세요.

   AWS CLI

   `project-arn`의 값을 삭제하려는 프로젝트의 이름으로 변경합니다.

   ```
   aws rekognition delete-project --project-arn project_arn \
     --profile custom-labels-access
   ```

   Python

   다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

   - `project_arn`: 삭제하려는 프로젝트의 ARN

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Amazon Rekognition Custom Labels project example used in the service
    documentation:
   https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-
   project.html
   Shows how to delete an existing Amazon Rekognition Custom Labels project.
   You must first delete any models and datasets that belong to the project.
   """

   import argparse
   import logging
   import time
   import boto3


   from botocore.exceptions import ClientError
   ```

```
logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)

        response = rek_client.delete_project(ProjectArn=project_arn)

        logger.info("project status: %s",response['Status'])

        deleted = False

        logger.info("waiting for project deletion: %s", project_arn)

        # Get the project name
        start = find_forward_slash(project_arn, 1) + 1
        end = find_forward_slash(project_arn, 2)
        project_name = project_arn[start:end]

        project_names = [project_name]

        while deleted is False:
```

```
            project_descriptions = rek_client.describe_projects(
                ProjectNames=project_names)['ProjectDescriptions']

            if len(project_descriptions) == 0:
                deleted = True

            else:
                time.sleep(5)

        logger.info("project deleted: %s",project_arn)

        return True

    except ClientError as err:
        logger.exception(
            "Couldn't delete project - %s: %s",
            project_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that you want to delete."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Deleting project: {args.project_arn}")
```

```
        # Delete the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_project(rekognition_client,
                       args.project_arn)

        print(f"Finished deleting project: {args.project_arn}")

    except ClientError as err:
        error_message = f"Problem deleting project: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- `project_arn`: 삭제하려는 프로젝트의 ARN

```
/*
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.List;
import java.util.Objects;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectResponse;
```

```
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProject {

    public static final Logger logger =
 Logger.getLogger(DeleteProject.class.getName());

    public static void deleteMyProject(RekognitionClient rekClient, String
 projectArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting project: {0}", projectArn);

            // Delete the project

            DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder().projectArn(projectArn).build();
            DeleteProjectResponse response =
 rekClient.deleteProject(deleteProjectRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Wait until deletion finishes

            Boolean deleted = false;

            do {

        DescribeProjectsRequest describeProjectsRequest =
 DescribeProjectsRequest.builder().build();
                    DescribeProjectsResponse describeResponse =
 rekClient.describeProjects(describeProjectsRequest);
                    List<ProjectDescription> projectDescriptions =
 describeResponse.projectDescriptions();

                    deleted = true;
```

```
                    for (ProjectDescription projectDescription :
 projectDescriptions) {

                        if (Objects.equals(projectDescription.projectArn(),
 projectArn)) {
                            deleted = false;
                            logger.log(Level.INFO, "Not deleted: {0}",
 projectDescription.projectArn());
                            Thread.sleep(5000);
                            break;
                        }
                    }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Project deleted: {0} ", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>\n\n" + "Where:\n"
             + "   project_arn - The ARN of the project that you want to delete.
\n\n";

        if (args.length != 1) {
             System.out.println(USAGE);
          System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
                .region(Region.US_WEST_2)
```

```
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .build();

            // Delete the project.
            deleteMyProject(rekClient, projectArn);

            System.out.println(String.format("Project deleted: %s",
 projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }
    }

}
```

프로젝트 설명(SDK)

`DescribeProjects` API를 사용하여 프로젝트에 대한 정보를 가져올 수 있습니다.

프로젝트를 설명하려면(SDK)

1. 아직 설치하지 않았다면 및 AWS SDK를 설치하고 구성하세요. AWS CLI 자세한 정보는 4단계: 및 SDK 설정 AWS CLIAWS을 참조하세요.
2. 프로젝트를 설명하려면 다음 예제 코드를 사용하세요. `project_name`을 설명하려는 프로젝트 이름으로 바꿉니다. `--project-names`를 지정하지 않은 경우 모든 프로젝트에 대한 설명이 반환됩니다.

AWS CLI

```
aws rekognition describe-projects --project-names project_name \
  --profile custom-labels-access
```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- project_name : 설명하고자 하는 프로젝트의 이름입니다. 이름을 지정하지 않은 경우 모든 프로젝트에 대한 설명이 반환됩니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels project.
"""
import argparse
import logging
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_info(project):
    """
    Displays information about a Custom Labels project.
    :param project: The project that you want to display information about.
    """
    print(f"Arn: {project['ProjectArn']}")
    print(f"Status: {project['Status']}")

    if len(project['Datasets']) == 0:
        print("Datasets: None")
    else:
        print("Datasets:")
```

```
    for dataset in project['Datasets']:
        print(f"\tCreated: {str(dataset['CreationTimestamp'])}")
        print(f"\tType: {dataset['DatasetType']}")
        print(f"\tARN: {dataset['DatasetArn']}")
        print(f"\tStatus: {dataset['Status']}")
        print(f"\tStatus message: {dataset['StatusMessage']}")
        print(f"\tStatus code: {dataset['StatusMessageCode']}")
        print()
    print()


def describe_projects(rek_client, project_name):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The project you want to describe. Pass None to describe
 all projects.
    """

    try:
        # Describe the project
        if project_name is None:
            logger.info("Describing all projects.")
        else:
            logger.info("Describing project: %s.",project_name)

        if project_name is None:
            response = rek_client.describe_projects()
        else:
            project_names = json.loads('["' + project_name + '"]')
            response = rek_client.describe_projects(ProjectNames=project_names)

        print('Projects\n--------')
        if len(response['ProjectDescriptions']) == 0:
            print("Project(s) not found.")
        else:
            for project in response['ProjectDescriptions']:
                display_project_info(project)

        logger.info("Finished project description.")

    except ClientError as err:
        logger.exception(
            "Couldn't describe project - %s: %s",
```

```
            project_name,err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "--project_name",  help="The name of the project that you want to
 describe.", required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Describing projects: {args.project_name}")

        # Describe the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_projects(rekognition_client,
                          args.project_name)

        if args.project_name is None:
            print("Finished describing all projects.")
        else:
            print("Finished describing project %s.", args.project_name)

    except ClientError as err:
```

```
        error_message = f"Problem describing project: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- project_name: 설명하고자 하는 프로젝트의 ARN 이름을 지정하지 않은 경우 모든 프로젝트에 대한 설명이 반환됩니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DescribeProjects {

    public static final Logger logger =
 Logger.getLogger(DescribeProjects.class.getName());
```

```
   public static void describeMyProjects(RekognitionClient rekClient, String
projectName) {

       DescribeProjectsRequest descProjects = null;

       // If a single project name is supplied, build projectNames argument

       List<String> projectNames = new ArrayList<String>();

       if (projectName == null) {
           descProjects = DescribeProjectsRequest.builder().build();
       } else {
           projectNames.add(projectName);
           descProjects =
DescribeProjectsRequest.builder().projectNames(projectNames).build();
       }

       // Display useful information for each project.

       DescribeProjectsResponse resp =
rekClient.describeProjects(descProjects);

       for (ProjectDescription projectDescription : resp.projectDescriptions())
{

           System.out.println("ARN: " + projectDescription.projectArn());
           System.out.println("Status: " +
projectDescription.statusAsString());
           if (projectDescription.hasDatasets()) {
               for (DatasetMetadata datasetDescription :
projectDescription.datasets()) {
                   System.out.println("\tdataset Type: " +
datasetDescription.datasetTypeAsString());
                   System.out.println("\tdataset ARN: " +
datasetDescription.datasetArn());
                   System.out.println("\tdataset Status: " +
datasetDescription.statusAsString());
               }
           }
           System.out.println();
       }

   }
```

```
    public static void main(String[] args) {

        String projectArn = null;

        // Get command line arguments

        final String USAGE = "\n" + "Usage: " + "<project_name>\n\n" + "Where:
\n"
                + "   project_name - (Optional) The name of the project that you
 want to describe. If not specified, all projects "
                + "are described.\n\n";

        if (args.length > 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        if (args.length == 1) {
            projectArn = args[0];
        }

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

AWS CloudFormation으로 프로젝트 생성

Amazon Rekognition Custom Labels는 리소스를 모델링하고 AWS 설정하는 데 도움이 되는 서비스인 와 AWS CloudFormation통합되어 있으므로 리소스와 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있습니다. 원하는 모든 리소스를 설명하는 템플릿을 생성하면 해당 AWS 리소스를 자동으로 프로비저닝하고 AWS CloudFormation 구성할 수 있습니다.

Amazon Rekognition 사용자 지정 라벨 프로젝트를 프로비저닝하고 구성하는 AWS CloudFormation 데 사용할 수 있습니다.

를 사용하면 템플릿을 재사용하여 AWS CloudFormation Amazon Rekognition 사용자 지정 레이블 프로젝트를 일관되고 반복적으로 설정할 수 있습니다. 프로젝트를 한 번 설명한 다음 여러 AWS 계정 및 지역에 동일한 프로젝트를 반복해서 프로비저닝하면 됩니다.

Amazon Rekognition 사용자 지정 라벨 및 템플릿 AWS CloudFormation

Amazon Rekognition Custom Labels 및 관련 서비스에 대한 프로젝트를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이 템플릿은 스택에 프로비저닝하려는 리소스를 설명합니다. AWS CloudFormation JSON이나 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하여 템플릿을 시작하는 데 도움을 받을 수 있습니다. AWS CloudFormation 자세한 내용은 AWS CloudFormation 사용 설명서에서 AWS CloudFormation Designer란 무엇입니까?를 참조하세요.

JSON 및 YAML 템플릿의 예를 비롯한 Amazon Rekognition Custom Labels 프로젝트에 대한 참조 정보는 Rekognition 리소스 유형 참조를 참조하세요.

에 대해 자세히 알아보십시오. AWS CloudFormation

자세히 AWS CloudFormation알아보려면 다음 리소스를 참조하십시오.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation API Reference
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

데이터 세트 관리

데이터 세트에는 모델을 훈련하거나 테스트하는 데 사용하는 이미지와 지정된 레이블이 들어 있습니다. 이 항목은 Amazon Rekognition Custom Labels 콘솔 및 AWS SDK를 사용하여 데이터 세트를 관리하는 방법을 보여줍니다.

주제

프로젝트에 데이터 세트 추가

훈련 데이터 세트 또는 테스트 데이터 세트를 기존 프로젝트에 추가할 수 있습니다. 기존 데이터 세트를 바꾸려면 먼저 기존 데이터 세트를 삭제하세요. 자세한 내용은 데이터 세트 삭제 섹션을 참조하세요. 그런 다음 새 데이터 세트를 추가합니다.

주제

프로젝트에 데이터 세트 추가(콘솔)

Amazon Rekognition Custom Labels 콘솔을 사용하여 훈련 또는 테스트 데이터 세트를 프로젝트에 추가할 수 있습니다.

프로젝트에 데이터 세트를 추가하려면

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다.

3. 왼쪽 탐색 창에서 프로젝트를 선택합니다. 프로젝트 보기가 표시됩니다.
4. 데이터 세트를 추가할 프로젝트를 선택합니다.
5. 왼쪽 탐색 창의 프로젝트 이름 아래에서 데이터 세트를 선택합니다.
6. 프로젝트에 기존 데이터 세트가 없는 경우 데이터 세트 생성 페이지가 표시됩니다. 다음을 수행합니다.

   a. 데이터 세트 생성 페이지에서 이미지 소스 정보를 입력합니다. 자세한 내용은 the section called “이미지가 포함된 데이터 세트 생성” 섹션을 참조하세요.

   b. 데이터 세트 생성을 선택하여 데이터 세트를 생성합니다.
7. 프로젝트에 기존 데이터 세트(훈련 또는 테스트)가 있는 경우 프로젝트 세부 정보 페이지가 표시됩니다. 다음을 수행합니다.

   a. 프로젝트 세부 정보 페이지에서 작업을 선택합니다.

   b. 훈련 데이터 세트를 추가하려면 훈련 데이터 세트 생성을 선택합니다.

   c. 테스트 데이터 세트를 추가하려면 테스트 데이터 세트 생성을 선택합니다.

   d. 데이터 세트 생성 페이지에서 이미지 소스 정보를 입력합니다. 자세한 내용은 the section called “이미지가 포함된 데이터 세트 생성” 섹션을 참조하세요.

   e. 데이터 세트 생성을 선택하여 데이터 세트를 생성합니다.
8. 데이터 세트에 이미지를 추가합니다. 자세한 내용은 더 많은 이미지 추가(콘솔) 섹션을 참조하세요.
9. 데이터 세트에 레이블을 추가합니다. 자세한 내용은 새 레이블 추가(콘솔) 섹션을 참조하세요.
10. 이미지에 레이블을 추가합니다. 이미지 수준 레이블을 추가하는 경우 the section called “이미지에 이미지 수준 레이블 지정” 항목을 참조하세요. 경계 상자를 추가하는 경우 경계 상자로 객체에 레이블 지정 항목을 참조하세요. 자세한 내용은 데이터 세트 목적 설정 섹션을 참조하세요.

프로젝트에 데이터 세트 추가(SDK)

다음과 같은 방법으로 기존 프로젝트에 훈련 데이터 또는 테스트 데이터 세트를 추가할 수 있습니다.

- 매니페스트 파일을 사용하여 데이터 세트를 생성합니다. 자세한 내용은 SageMaker Ground Truth 매니페스트 파일 (SDK) 을 사용하여 데이터세트 만들기 섹션을 참조하세요.
- 빈 데이터 세트를 만든 다음 데이터 세트를 채웁니다. 다음 예제는 빈 데이터 세트를 생성하는 방법을 보여줍니다. 빈 데이터 세트를 만든 후 항목을 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요.

프로젝트(SDK)에 데이터 세트를 추가하려면

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제를 사용하여 데이터 세트에 JSON 라인을 추가하세요.

   CLI

   project_arn 항목을 데이터 세트를 추가하려는 프로젝트로 바꿉니다. dataset_type 항목을 TRAIN 항목으로 바꾸어 훈련 데이터 세트를 생성하거나 TEST 항목으로 바꾸어 테스트 데이터 세트를 생성하세요.

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --profile custom-labels-access
   ```

   Python

   다음 코드를 사용하여 데이터 세트를 생성하세요. 다음 명령줄 옵션을 제공하세요.

   - project_arn: 테스트 데이터 세트를 추가하려는 프로젝트의 ARN입니다.
   - type: 생성하려는 데이터 세트의 유형(훈련 또는 테스트)

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   import argparse
   import logging
   import time
   import boto3

   from botocore.exceptions import ClientError

   logger = logging.getLogger(__name__)

   def create_empty_dataset(rek_client, project_arn, dataset_type):
       """
       Creates an empty Amazon Rekognition Custom Labels dataset.
       :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
   ```

```
   :param project_arn: The ARN of the project in which you want to create a
dataset.
   :param dataset_type: The type of the dataset that you want to create (train
or test).
   """

   try:
       #Create the dataset.
       logger.info("Creating empty %s dataset for project %s",
           dataset_type, project_arn)

       dataset_type=dataset_type.upper()

       response = rek_client.create_dataset(
           ProjectArn=project_arn, DatasetType=dataset_type
       )

       dataset_arn=response['DatasetArn']

       logger.info("dataset ARN: %s", dataset_arn)

       finished=False
       while finished is False:

           dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

           status=dataset['DatasetDescription']['Status']

           if status == "CREATE_IN_PROGRESS":

               logger.info(("Creating dataset: %s ", dataset_arn))
               time.sleep(5)
               continue

           if status == "CREATE_COMPLETE":
               logger.info("Dataset created: %s", dataset_arn)
               finished=True
               continue

           if status == "CREATE_FAILED":
               error_message = f"Dataset creation failed: {status} :
{dataset_arn}"
               logger.exception(error_message)
               raise Exception(error_message)
```

```
            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception("Couldn't create dataset: %s", err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the empty dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the empty dataset that you want to
 create (train or test)."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating empty {args.dataset_type} dataset for project
 {args.project_arn}")
```

```
        # Create the empty dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

        print(f"Finished creating empty dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용하여 데이터 세트를 생성하세요. 다음 명령줄 옵션을 제공하세요.

- `project_arn`: 테스트 데이터 세트를 추가하려는 프로젝트의 ARN입니다.
- `type`: 생성하려는 데이터 세트의 유형(훈련 또는 테스트)

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
```

```
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateEmptyDataset {

    public static final Logger logger =
 Logger.getLogger(CreateEmptyDataset.class.getName());

    public static String createMyEmptyDataset(RekognitionClient rekClient,
 String projectArn, String datasetType)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating empty {0} dataset for project :
 {1}",
                    new Object[] { datasetType.toString(), projectArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
 datasetType);
                throw new Exception("Unrecognized dataset type: " +
 datasetType);

            }
```

```
            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
                    .datasetType(requestDatasetType).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until updates finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
```

```
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
 datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
 response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
 e.getMessage());
            throw e;
        }

    }


    public static void main(String args[]) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n
\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the empty dataset that you want
 to create (train or test).\n\n";


        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }
```

```
        projectArn = args[0];
        datasetType = args[1];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyEmptyDataset(rekClient, projectArn,
 datasetType);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. 데이터 세트에 이미지 추가 자세한 내용은 더 많은 이미지 추가(SDK) 섹션을 참조하세요.

데이터 세트에 더 많은 이미지 추가

Amazon Rekognition Custom Labels 콘솔을 사용하거나 `UpdateDatasetEntries` API를 호출하여 데이터 세트에 더 많은 이미지를 추가할 수 있습니다.

주제

- 더 많은 이미지 추가(콘솔)
- 더 많은 이미지 추가(SDK)

더 많은 이미지 추가(콘솔)

Amazon Rekognition Custom Labels 콘솔을 사용하면 로컬 컴퓨터에서 이미지를 업로드합니다. 이미지는 데이터 세트를 생성하는 데 사용된 이미지가 저장되는 Amazon S3 버킷 위치(콘솔 또는 외부)에 추가됩니다.

데이터 세트에 이미지를 더 추가하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다.
3. 왼쪽 탐색 창에서 프로젝트를 선택합니다. 프로젝트 보기가 표시됩니다.
4. 사용하려는 프로젝트를 선택합니다.
5. 왼쪽 탐색 창의 프로젝트 이름 아래에서 데이터 세트를 선택합니다.
6. 작업을 선택하고 이미지를 추가할 데이터 세트를 선택합니다.
7. 데이터 세트에 업로드할 이미지를 선택합니다. 로컬 컴퓨터에서 업로드할 이미지를 선택하거나 드래그할 수 있습니다. 한 번에 최대 30개의 이미지를 업로드할 수 있습니다.
8. 이미지 업로드를 선택합니다.
9. 변경 사항 저장을 선택합니다.
10. 이미지에 레이블을 지정합니다. 자세한 내용은 이미지 레이블 지정 섹션을 참조하세요.

더 많은 이미지 추가(SDK)

`UpdateDatasetEntries` 항목은 매니페스트 파일에 JSON 라인을 업데이트하거나 추가합니다. JSON 라인을 `GroundTruth` 필드에 byte64로 인코딩된 데이터 객체로 전달합니다. AWS SDK를 사용하여 `UpdateDatasetEntries` 항목을 호출하는 경우 SDK가 데이터를 자동으로 인코딩합니다. 각 JSON 라인에는 지정된 레이블 또는 경계 상자 정보와 같은 단일 이미지에 대한 정보가 들어 있습니다. 예:

```
{"source-ref":"s3://bucket/image","BB":{"annotations":
[{"left":1849,"top":1039,"width":422,"height":283,"class_id":0},
```

```
{"left":1849,"top":1340,"width":443,"height":415,"class_id":1},
{"left":2637,"top":1380,"width":676,"height":338,"class_id":2},
{"left":2634,"top":1051,"width":673,"height":338,"class_id":3}],"image_size":
[{"width":4000,"height":2667,"depth":3}]},"BB-metadata":{"job-name":"labeling-job/
BB","class-map":
{"0":"comparator","1":"pot_resistor","2":"ir_phototransistor","3":"ir_led"},"human-
annotated":"yes","objects":[{"confidence":1},{"confidence":1},{"confidence":1},
{"confidence":1}],"creation-date":"2021-06-22T10:11:18.006Z","type":"groundtruth/
object-detection"}}
```

자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

source-ref 필드를 키로 사용하여 업데이트하려는 이미지를 식별합니다. 데이터 세트에 일치하는 source-ref 필드 값이 없는 경우 JSON 라인이 새 이미지로 추가됩니다.

데이터 세트에 더 많은 이미지를 추가하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제를 사용하여 데이터 세트에 JSON 라인을 추가하세요.

   CLI

   GroundTruth의 값을 사용하려는 JSON 라인으로 바꿉니다. JSON 라인 내의 모든 특수 문자는 이스케이프해야 합니다.

```
aws rekognition update-dataset-entries\
  --dataset-arn dataset_arn \
  --changes '{"GroundTruth" : "{\"source-ref\":\"s3://your_bucket/your_image
\",\"BB\":{\"annotations\":[{\"left\":1776,\"top\":1017,\"width\":458,\"height
\":317,\"class_id\":0},{\"left\":1797,\"top\":1334,\"width\":418,\"height
\":415,\"class_id\":1},{\"left\":2597,\"top\":1361,\"width\":655,\"height
\":329,\"class_id\":2},{\"left\":2581,\"top\":1020,\"width\":689,\"height
\":338,\"class_id\":3}],\"image_size\":[{\"width\":4000,\"height\":2667,
\"depth\":3}]},\"BB-metadata\":{\"job-name\":\"labeling-job/BB\",\"class-map
\":{\"0\":\"comparator\",\"1\":\"pot_resistor\",\"2\":\"ir_phototransistor\",
\"3\":\"ir_led\"},\"human-annotated\":\"yes\",\"objects\":[{\"confidence\":1},
{\"confidence\":1},{\"confidence\":1},{\"confidence\":1}],\"creation-date\":
\"2021-06-22T10:10:48.492Z\",\"type\":\"groundtruth/object-detection\"}}" }' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- dataset_arn: 업데이트하려는 데이터 세트의 ARN입니다.
- updates_file: JSON 라인 업데이트가 포함된 파일입니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that yuo want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
 updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)
```

```
        with open(updates_file) as f:
            manifest_file = f.read()


        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']
            status_message=dataset['DatasetDescription']['StatusMessage']

            if status == "UPDATE_IN_PROGRESS":

                logger.info("Updating dataset: %s ", dataset_arn)
                time.sleep(5)
                continue

            if status == "UPDATE_COMPLETE":
                logger.info("Dataset updated: %s : %s : %s",
                    status, status_message, dataset_arn)
                finished=True
                continue

            if status == "UPDATE_FAILED":
                error_message = f"Dataset update failed: {status} :
{status_message} : {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset update:
{status} : {status_message} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)
```

```
        logger.info("Added entries to dataset")

        return status, status_message


    except ClientError as err:
        logger.exception("Couldn't update dataset: %s", err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to update."
    )

    parser.add_argument(
        "updates_file", help="The manifest file of JSON Lines that contains the
 updates."
    )

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        #get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Updating dataset {args.dataset_arn} with entries from
 {args.updates_file}.")

        # Update the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status, status_message=update_dataset_entries(rekognition_client,
```

```
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")


    except ClientError as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")

    except Exception as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

- dataset_arn: 업데이트하려는 데이터 세트의 ARN입니다.
- update_file: JSON 라인 업데이트가 포함된 파일입니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetChanges;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesRequest;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesResponse;

import java.io.FileInputStream;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

public class UpdateDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(UpdateDatasetEntries.class.getName());

    public static String updateMyDataset(RekognitionClient rekClient, String
 datasetArn,
            String updateFile
            ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Updating dataset {0}",
                    new Object[] { datasetArn});


            InputStream sourceStream = new FileInputStream(updateFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            DatasetChanges datasetChanges = DatasetChanges.builder()
                    .groundTruth(sourceBytes).build();

            UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
 UpdateDatasetEntriesRequest.builder()
                    .changes(datasetChanges)
                    .datasetArn(datasetArn)
                    .build();

            UpdateDatasetEntriesResponse response =
 rekClient.updateDatasetEntries(updateDatasetEntriesRequest);

            boolean updated = false;

            //Wait until update completes
```

```
            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(datasetArn).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

                switch (status) {

                case UPDATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset updated");
                    updated = true;
                    break;

                case UPDATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case UPDATE_FAILED:
                    String error = "Dataset update failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
datasetArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected update state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
datasetArn;
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }
```

```
            } while (updated == false);

            return datasetArn;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not update dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String updatesFile = null;
        String datasetArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
 <updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
 update.\n\n"
                + "   update_file - The file that includes in JSON Line updates.
\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        datasetArn = args[0];
        updatesFile = args[1];


        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

             // Update the dataset
```

```
            datasetArn = updateMyDataset(rekClient, datasetArn, updatesFile);

            System.out.println(String.format("Dataset updated: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

기존 데이터 세트를 사용하여 데이터 세트 생성(SDK)

다음 절차는 CreateDataset 작업을 사용하여 기존 데이터 세트에서 데이터 세트를 생성하는 방법을 보여줍니다.

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제 코드를 사용하면 다른 데이터 세트를 복사하여 데이터 세트를 만들 수 있습니다.

   AWS CLI

   다음 코드를 사용하여 데이터 세트를 생성하세요. 다음을 바꿉니다.

   - `project_arn`: 데이터 세트를 추가하려는 프로젝트의 ARN입니다.
   - `dataset_type`: 프로젝트에서 만들려는 데이터 세트 유형(`TRAIN` 또는 `TEST`)으로 바꿉니다.
   - `dataset_arn`: 복사하려는 데이터 세트의 ARN으로 바꿉니다.

```
aws rekognition create-dataset --project-arn project_arn \
```

```
  --dataset-type dataset_type \
  --dataset-source '{ "DatasetArn" : "dataset_arn" }' \
  --profile custom-labels-access
```

Python

다음 예제에서는 기존 데이터 세트를 사용하여 데이터 세트를 생성하고 해당 ARN을 표시합니다.

프로그램을 실행하려면 다음 명령줄 인수를 제공하세요.

- `project_arn`: 사용하려는 프로젝트의 ARN
- `dataset_type`: 만들려는 프로젝트의 데이터 세트 유형(`train` 또는 `test`)
- `dataset_arn`: 데이터 세트를 생성하는 데 사용할 기존 데이터 세트의 ARN

```
# Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def create_dataset_from_existing_dataset(rek_client, project_arn, dataset_type,
 dataset_arn):
    """
    Creates an Amazon Rekognition Custom Labels dataset using an existing
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
```

```
    :param dataset_arn: The ARN of the existing dataset that you want to use.
    """

    try:
        # Create the dataset

        dataset_type=dataset_type.upper()

        logger.info(
            "Creating %s dataset for project %s from dataset %s.",
                dataset_type,project_arn, dataset_arn)

        dataset_source = json.loads(
            '{ "DatasetArn": "' + dataset_arn + '"}'
        )

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type,
DatasetSource=dataset_source
        )

        dataset_arn = response['DatasetArn']

        logger.info("New dataset ARN: %s", dataset_arn)

        finished = False
        while finished is False:

            dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

            status = dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":

                logger.info(("Creating dataset: %s ", dataset_arn))
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished = True
                continue

            if status == "CREATE_FAILED":
```

```
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception(
            "Couldn't create dataset: %s",err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to copy from."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
```

```
    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn = create_dataset_from_existing_dataset(rekognition_client,
                                    args.project_arn,
                                    args.dataset_type,
                                    args.dataset_arn)

        print(f"Finished creating dataset: {dataset_arn}")

    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

다음 예제에서는 기존 데이터 세트를 사용하여 데이터 세트를 생성하고 해당 ARN을 표시합니다.

프로그램을 실행하려면 다음 명령줄 인수를 제공하세요.

- `project_arn`: 사용하려는 프로젝트의 ARN
- `dataset_type`: 만들려는 프로젝트의 데이터 세트 유형(`train` 또는 `test`)

- dataset_arn: 데이터 세트를 생성하는 데 사용할 기존 데이터 세트의 ARN

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetExisting {

    public static final Logger logger =
 Logger.getLogger(CreateDatasetExisting.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String existingDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 dataset {2} ",
                    new Object[] { datasetType.toString(), projectArn,
 existingDatasetArn });
```

```
            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
                throw new Exception("Unrecognized dataset type: " +
datasetType);

            }

            DatasetSource datasetSource =
DatasetSource.builder().datasetArn(existingDatasetArn).build();

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until create finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();
```

```
                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }
```

```
    public static void main(String[] args) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;
        String datasetSourceArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
 <dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
 create (train or test).\n\n"
                + "   dataset_arn - the ARN of the dataset that you want to copy
 from.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        datasetSourceArn = args[2];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 datasetSourceArn);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
```

```
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

데이터 세트 설명(SDK)

`DescribeDataset` API를 사용하여 데이터 세트에 대한 정보를 가져올 수 있습니다.

데이터 세트를 설명하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제 코드를 사용하여 데이터 세트를 설명하세요.

   AWS CLI

   `dataset-arn`의 값을 설명하려는 데이터 세트의 ARN으로 변경합니다.

   ```
   aws rekognition describe-dataset --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   Python

   다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

   - dataset_arn: 설명하려는 데이터 세트의 ARN입니다.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   ```

```
Shows how to describe an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_dataset(rek_client, dataset_arn):
    """
    Describes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to describe.

    """

    try:
        # Describe the dataset
        logger.info("Describing dataset %s", dataset_arn)

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        description = dataset['DatasetDescription']

        print(f"Created: {str(description['CreationTimestamp'])}")
        print(f"Updated: {str(description['LastUpdatedTimestamp'])}")
        print(f"Status: {description['Status']}")
        print(f"Status message: {description['StatusMessage']}")
        print(f"Status code: {description['StatusMessageCode']}")
        print("Stats:")
        print(
            f"\tLabeled entries: {description['DatasetStats']
['LabeledEntries']}")
        print(
            f"\tTotal entries: {description['DatasetStats']['TotalEntries']}")
        print(f"\tTotal labels: {description['DatasetStats']['TotalLabels']}")

    except ClientError as err:
        logger.exception("Couldn't describe dataset: %s",
                         err.response['Error']['Message'])
```

```
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to describe."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Describing dataset {args.dataset_arn}")

        # Describe the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_dataset(rekognition_client, args.dataset_arn)

        print(f"Finished describing dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message=f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
```

```
if __name__ == "__main__":
    main()
```

Java V2

- dataset_arn: 설명하려는 데이터 세트의 ARN입니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStats;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeDataset {

    public static final Logger logger =
 Logger.getLogger(DescribeDataset.class.getName());

    public static void describeMyDataset(RekognitionClient rekClient, String
 datasetArn) {

        try {

            DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();
```

```
            DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();
            DatasetStats datasetStats = datasetDescription.datasetStats();

            System.out.println("ARN: " + datasetArn);
            System.out.println("Created: " +
datasetDescription.creationTimestamp().toString());
            System.out.println("Updated: " +
datasetDescription.lastUpdatedTimestamp().toString());
            System.out.println("Status: " +
datasetDescription.statusAsString());
            System.out.println("Message: " +
datasetDescription.statusMessage());
            System.out.println("Total Labels: " +
datasetStats.totalLabels().toString());
            System.out.println("Total entries: " +
datasetStats.totalEntries().toString());
            System.out.println("Entries with labels: " +
datasetStats.labeledEntries().toString());
            System.out.println("Entries with at least 1 error: " +
datasetStats.errorEntries().toString());

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
describe.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }
```

```
        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the dataset.
            describeMyDataset(rekClient, datasetArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

데이터 세트 항목 나열(SDK)

`ListDatasetEntries` API를 사용하여 데이터 세트에 있는 각 이미지의 JSON 라인을 나열할 수 있습니다. 자세한 내용은 매니페스트 파일 생성 섹션을 참조하세요.

데이터 세트 항목을 나열하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제 코드를 사용하여 데이터 세트의 항목을 나열할 수 있습니다.

   AWS CLI

   `dataset-arn`의 값을 나열하려는 데이터 세트의 ARN으로 변경합니다.

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \
  --profile custom-labels-access
```

오류가 있는 JSON 라인만 나열하려면 has-errors 항목을 지정하세요.

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \
  --has-errors \
  --profile custom-labels-access
```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- dataset_arn: 나열하려는 데이터 세트의 ARN입니다.
- show_errors_only: 오류만 표시하려면 true 항목을 지정하세요. 그렇지 않으면 false

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to list the entries in an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def list_dataset_entries(rek_client, dataset_arn, show_errors):
    """
    Lists the entries in an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataet that you want to use.
    """
```

```
    try:
        # List the entries.
        logger.info("Listing dataset entries for the dataset %s.", dataset_arn)

        finished = False
        count = 0
        next_token = ""
        show_errors_only = False

        if show_errors.lower() == "true":
            show_errors_only = True

        while finished is False:

            response = rek_client.list_dataset_entries(
                DatasetArn=dataset_arn,
                HasErrors=show_errors_only,
                MaxResults=100,
                NextToken=next_token)

            count += len(response['DatasetEntries'])

            for entry in response['DatasetEntries']:
                print(entry)

            if 'NextToken' not in response:
                finished = True
                logger.info("No more entries. Total:%s", count)
            else:
                next_token = next_token = response['NextToken']
                logger.info("Getting more entries. Total so far :%s", count)

    except ClientError as err:
        logger.exception(
            "Couldn't list dataset: %s",
             err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """
```

```
    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to list."

    )

    parser.add_argument(
        "show_errors_only", help="true if you want to see errors only. false
 otherwise."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing entries for  dataset {args.dataset_arn}")

        # List the dataset entries.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_dataset_entries(rekognition_client,
                             args.dataset_arn,
                             args.show_errors_only)

        print(f"Finished listing entries for dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
```

```
if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- dataset_arn: 나열하려는 데이터 세트의 ARN입니다.
- show_errors_only: 오류만 표시하려면 `true` 항목을 지정하세요. 그렇지 않으면 `false`

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.paginators.ListDatasetEntriesIterable;


import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ListDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(ListDatasetEntries.class.getName());

    public static void listMyDatasetEntries(RekognitionClient rekClient, String
 datasetArn, boolean showErrorsOnly)
            throws Exception, RekognitionException {
```

```
        try {

            logger.log(Level.INFO, "Listing dataset {0}", new Object[]
{ datasetArn });

            ListDatasetEntriesRequest listDatasetEntriesRequest =
ListDatasetEntriesRequest.builder()

.hasErrors(showErrorsOnly).datasetArn(datasetArn).maxResults(1).build();

            ListDatasetEntriesIterable datasetEntriesList = rekClient
                    .listDatasetEntriesPaginator(listDatasetEntriesRequest);

            datasetEntriesList.stream().flatMap(r ->
r.datasetEntries().stream())
                    .forEach(datasetEntry ->
System.out.println(datasetEntry.toString()));

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not update dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        boolean showErrorsOnly = false;
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
<updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
update.\n\n"
                + "   show_errors_only - true to show only errors. false
otherwise.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        datasetArn = args[0];
```

```
        if (args[1].toLowerCase().equals("true")) {

            showErrorsOnly = true;
        }

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // list the dataset entries.

            listMyDatasetEntries(rekClient, datasetArn, showErrorsOnly);

            System.out.println(String.format("Finished listing entries for :
 %s", datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

훈련 데이터 세트 배포(SDK)

Amazon Rekognition Custom Labels가 모델을 훈련하려면 훈련 데이터 세트와 테스트 데이터 세트가 필요합니다.

API를 사용하는 경우 DistributeDataSetEntries API를 사용하여 훈련 데이터 세트의 20%를 빈 테스트 데이터 세트에 배포할 수 있습니다. 사용 가능한 매니페스트 파일이 하나뿐인 경우 훈련 데이터 세트를 배포하는 것이 유용할 수 있습니다. 단일 매니페스트 파일을 사용하여 훈련 데이터 세트를 만드세요. 그런 다음 빈 테스트 데이터 세트를 만들고 `DistributeDatasetEntries` 항목을 사용하여 테스트 데이터 세트를 채웁니다.

> **Note**
>
> Amazon Rekognition Custom Labels 콘솔을 사용하고 데이터 세트 하나로 프로젝트를 시작하는 경우, Amazon Rekognition Custom Labels는 훈련 중에 훈련 데이터 세트를 분할(배포)하여 테스트 데이터 세트를 생성합니다. 훈련 데이터 세트 항목의 20%가 테스트 데이터 세트로 이동합니다.

훈련 데이터 세트를 배포하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 프로젝트 생성 자세한 내용은 Amazon Rekognition Custom Labels 프로젝트 생성(SDK) 섹션을 참조하세요.
3. 훈련 데이터 세트를 생성합니다. 데이터 세트에 관한 자세한 내용은 훈련 및 테스트 데이터 세트 생성 항목을 참조하세요.
4. 빈 테스트 데이터 세트를 생성합니다.
5. 다음 예제 코드를 사용하여 훈련 데이터 세트 항목의 20%를 테스트 데이터 세트에 배포합니다. DescribeProjects를 호출하여 프로젝트 데이터 세트의 Amazon 리소스 이름(ARN)을 가져올 수 있습니다. 예제 코드는 프로젝트 설명(SDK) 항목을 참조하세요.

   AWS CLI

   `training_dataset-arn` 및 `test_dataset_arn`의 값을 사용하려는 데이터 세트의 ARN으로 변경합니다.

   ```
   aws rekognition distribute-dataset-entries --datasets ['{"Arn":
    "training_dataset_arn"}, {"Arn": "test_dataset_arn"}'] \
     --profile custom-labels-access
   ```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- training_dataset_arn: 항목을 배포하는 데 사용되는 훈련 데이터 세트의 ARN입니다.
- test_dataset_arn: 항목을 배포하는 데 사용되는 테스트 데이터 세트의 ARN입니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def check_dataset_status(rek_client, dataset_arn):
    """
    Checks the current status of a dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The dataset that you want to check.
    :return: The dataset status and status message.
    """
    finished = False
    status = ""
    status_message = ""

    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        status = dataset['DatasetDescription']['Status']
        status_message = dataset['DatasetDescription']['StatusMessage']

        if status == "UPDATE_IN_PROGRESS":
```

```
            logger.info("Distributing dataset: %s ", dataset_arn)
            time.sleep(5)
            continue

        if status == "UPDATE_COMPLETE":
            logger.info(
                "Dataset distribution complete: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            continue

        if status == "UPDATE_FAILED":
            logger.exception(
                "Dataset distribution failed: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            break

        logger.exception(
            "Failed. Unexpected state for dataset distribution: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        status_message = "An unexpected error occurred while distributing the
 dataset"
        break

    return status, status_message


def distribute_dataset_entries(rek_client, training_dataset_arn,
 test_dataset_arn):
    """
    Distributes 20% of the supplied training dataset into the supplied test
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param training_dataset_arn: The ARN of the training dataset that you
 distribute entries from.
    :param test_dataset_arn: The ARN of the test dataset that you distribute
 entries to.
    """

    try:
        # List dataset labels.
```

```
        logger.info("Distributing training dataset entries (%s) into test
 dataset (%s).",
            training_dataset_arn,test_dataset_arn)


        datasets = json.loads(
            '[{"Arn" : "' + str(training_dataset_arn) + '"},{"Arn" : "' +
 str(test_dataset_arn) + '"}]')

        rek_client.distribute_dataset_entries(
            Datasets=datasets
        )

        training_dataset_status, training_dataset_status_message =
 check_dataset_status(
            rek_client, training_dataset_arn)
        test_dataset_status, test_dataset_status_message = check_dataset_status(
            rek_client, test_dataset_arn)

        if training_dataset_status == 'UPDATE_COMPLETE' and test_dataset_status
 == "UPDATE_COMPLETE":
            print("Distribution complete")
        else:
            print("Distribution failed:")
            print(
                f"\ttraining dataset: {training_dataset_status} :
 {training_dataset_status_message}")
            print(
                f"\ttest dataset: {test_dataset_status} :
 {test_dataset_status_message}")

    except ClientError as err:
        logger.exception(
            "Couldn't distribute dataset: %s",err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
```

```
        "training_dataset_arn", help="The ARN of the training dataset that you
 want to distribute from."
    )

    parser.add_argument(
        "test_dataset_arn", help="The ARN of the test dataset that you want to
 distribute to."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Distributing training dataset entries
 ({args.training_dataset_arn}) "\
            f"into test dataset ({args.test_dataset_arn}).")

        # Distribute the datasets.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        distribute_dataset_entries(rekognition_client,
                                   args.training_dataset_arn,
                                   args.test_dataset_arn)

        print("Finished distributing datasets.")

    except ClientError as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem listing dataset labels: {err}")
    except Exception as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem distributing datasets: {err}")
```

```
if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- training_dataset_arn: 항목을 배포하는 데 사용되는 훈련 데이터 세트의 ARN입니다.
- test_dataset_arn: 항목을 배포하는 데 사용되는 테스트 데이터 세트의 ARN입니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DistributeDataset;
import
 software.amazon.awssdk.services.rekognition.model.DistributeDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DistributeDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(DistributeDatasetEntries.class.getName());

    public static DatasetStatus checkDatasetStatus(RekognitionClient rekClient,
 String datasetArn)
```

```
            throws Exception, RekognitionException {

        boolean distributed = false;
        DatasetStatus status = null;

        // Wait until distribution completes

        do {

            DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();
            DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

            status = datasetDescription.status();

            logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

            switch (status) {

            case UPDATE_COMPLETE:
                logger.log(Level.INFO, "Dataset updated");
                distributed = true;
                break;

            case UPDATE_IN_PROGRESS:
                Thread.sleep(5000);
                break;

            case UPDATE_FAILED:
                String error = "Dataset distribution failed: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, error);
                break;

            default:
                String unexpectedError = "Unexpected distribution state: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
```

```
                logger.log(Level.SEVERE, unexpectedError);

            }

        } while (distributed == false);

        return status;

    }

    public static void distributeMyDatasetEntries(RekognitionClient rekClient,
String trainingDatasetArn,
            String testDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Distributing {0} dataset to {1} ",
                    new Object[] { trainingDatasetArn, testDatasetArn });

            DistributeDataset distributeTrainingDataset =
DistributeDataset.builder().arn(trainingDatasetArn).build();

            DistributeDataset distributeTestDataset =
DistributeDataset.builder().arn(testDatasetArn).build();

            ArrayList<DistributeDataset> datasets = new ArrayList();

            datasets.add(distributeTrainingDataset);
            datasets.add(distributeTestDataset);

            DistributeDatasetEntriesRequest distributeDatasetEntriesRequest =
DistributeDatasetEntriesRequest.builder()
                    .datasets(datasets).build();

            rekClient.distributeDatasetEntries(distributeDatasetEntriesRequest);

            DatasetStatus trainingStatus = checkDatasetStatus(rekClient,
trainingDatasetArn);
            DatasetStatus testStatus = checkDatasetStatus(rekClient,
testDatasetArn);

            if (trainingStatus == DatasetStatus.UPDATE_COMPLETE && testStatus ==
DatasetStatus.UPDATE_COMPLETE) {
```

```
                logger.log(Level.INFO, "Successfully distributed dataset: {0}",
 trainingDatasetArn);

            } else {

                throw new Exception("Failed to distribute dataset: " +
 trainingDatasetArn);
            }

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not distribute dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String trainingDatasetArn = null;
        String testDatasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<training_dataset_arn>
 <test_dataset_arn>\n\n" + "Where:\n"
                + "   training_dataset_arn - the ARN of the dataset that you
 want to distribute from.\n\n"
                + "   test_dataset_arn - the ARN of the dataset that you want to
 distribute to.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        trainingDatasetArn = args[0];
        testDatasetArn = args[1];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
```

```
                .build();

            // Distribute the dataset
            distributeMyDatasetEntries(rekClient, trainingDatasetArn,
 testDatasetArn);

            System.out.println("Datasets distributed.");

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

데이터 세트 삭제

프로젝트에서 훈련 및 테스트 데이터 세트를 삭제할 수 있습니다.

주제

- 데이터 세트 삭제(콘솔)
- Amazon Rekognition Custom Labels 데이터 세트 삭제(SDK)

데이터 세트 삭제(콘솔)

데이터 세트를 삭제하려면 다음 절차를 따르세요. 이후, 프로젝트에 데이터 세트(훈련 또는 테스트)가 하나 남아 있으면 프로젝트 세부 정보 페이지가 표시됩니다. 프로젝트에 남은 데이터 세트가 없는 경우 데이터 세트 생성 페이지가 표시됩니다.

훈련 데이터 세트를 삭제한 경우 모델을 훈련하기 전에 프로젝트에 사용할 새 훈련 데이터 세트를 만들어야 합니다. 자세한 내용은 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성 섹션을 참조하세요.

테스트 데이터 세트를 삭제하면 새 테스트 데이터 세트를 만들지 않고도 모델을 훈련할 수 있습니다. 훈련 중, 훈련 데이터 세트가 분할되어 프로젝트에 사용할 새 테스트 데이터 세트가 생성됩니다. 훈련 데이터 세트를 분할하면 훈련에 사용할 수 있는 이미지 수가 줄어듭니다. 품질을 유지하려면 모델을 훈련하기 전에 새 테스트 데이터 세트를 만드는 것이 좋습니다. 자세한 내용은 프로젝트에 데이터 세트 추가 섹션을 참조하세요.

데이터 세트를 삭제하려면

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 왼쪽 창에서 사용자 지정 레이블 사용을 선택합니다. Amazon Rekognition Custom Labels 랜딩 페이지가 표시됩니다.
3. 왼쪽 탐색 창에서 프로젝트를 선택합니다. 프로젝트 보기가 표시됩니다.
4. 삭제하려는 데이터 세트가 들어 있는 프로젝트를 선택합니다.
5. 왼쪽 탐색 창의 프로젝트 이름 아래에서 데이터 세트를 선택합니다.
6. 작업을 선택합니다.
7. 훈련 데이터 세트를 삭제하려면 훈련 데이터 세트 삭제를 선택합니다.
8. 테스트 데이터 세트를 삭제하려면 테스트 데이터 세트 삭제를 선택합니다.
9. 훈련 또는 테스트 데이터 세트 삭제 대화 상자에 삭제를 입력하여 데이터 세트 삭제를 확인합니다.
10. 훈련 또는 테스트 데이터 세트 삭제를 선택하여 데이터 세트를 삭제합니다.

Amazon Rekognition Custom Labels 데이터 세트 삭제(SDK)

Amazon Rekognition Custom Labels 데이터 세트를 삭제할 때는 DeleteDataset를 호출하고 삭제할 데이터 세트의 Amazon 리소스 이름(ARN)을 제공하여 삭제할 수 있습니다. 프로젝트 내 훈련 및 테스트 데이터 세트의 ARN을 가져오려면 DescribeProjects를 호출하세요. 응답에는 ProjectDescription 객체 배열이 포함됩니다. 데이터 세트 ARN(`DatasetArn`) 및 데이터 세트 유형(`DatasetType`)이 `Datasets` 목록에 있습니다.

훈련 데이터 세트를 삭제한 경우 모델을 훈련하기 전에 프로젝트에 사용할 새 훈련 데이터 세트를 만들어야 합니다. 테스트 데이터 세트를 삭제한 경우 모델을 훈련하기 전에 새 테스트 데이터 세트를 만들어야 합니다. 자세한 내용은 프로젝트에 데이터 세트 추가(SDK) 섹션을 참조하세요.

데이터 세트를 삭제하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.

2. 다음 코드를 사용하여 데이터 세트를 삭제하세요.

AWS CLI

`dataset-arn`의 값을 삭제하려는 데이터 세트의 ARN으로 변경합니다.

```
aws rekognition delete-dataset --dataset-arn dataset-arn \
  --profile custom-labels-access
```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- dataset_arn: 삭제하려는 데이터 세트의 ARN입니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to delete an Amazon Rekognition Custom Labels dataset.
"""
import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_dataset(rek_client, dataset_arn):
    """
    Deletes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to delete.
    """

    try:
        # Delete the dataset,
        logger.info("Deleting dataset: %s", dataset_arn)
```

```
        rek_client.delete_dataset(DatasetArn=dataset_arn)

        deleted = False

        logger.info("waiting for dataset deletion %s", dataset_arn)

        # Dataset might not be deleted yet, so wait.
        while deleted is False:
            try:
                rek_client.describe_dataset(DatasetArn=dataset_arn)
                time.sleep(5)
            except ClientError as err:
                if err.response['Error']['Code'] == 'ResourceNotFoundException':
                    logger.info("dataset deleted: %s", dataset_arn)
                    deleted = True
                else:
                    raise

        logger.info("dataset deleted: %s", dataset_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete dataset - %s: %s",
                         dataset_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to delete."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
```

```
    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Deleting dataset: {args.dataset_arn}")

        # Delete the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_dataset(rekognition_client,
                       args.dataset_arn)

        print(f"Finished deleting dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem deleting dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- dataset_arn: 삭제하려는 데이터 세트의 ARN입니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteDataset {

    public static final Logger logger =
 Logger.getLogger(DeleteDataset.class.getName());

    public static void deleteMyDataset(RekognitionClient rekClient, String
 datasetArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting dataset: {0}", datasetArn);

            // Delete the dataset

            DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder().datasetArn(datasetArn).build();

            DeleteDatasetResponse response =
 rekClient.deleteDataset(deleteDatasetRequest);

            // Wait until deletion finishes

            DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();

            Boolean deleted = false;

            do {

                try {

                    rekClient.describeDataset(describeDatasetRequest);
                    Thread.sleep(5000);
                } catch (RekognitionException e) {
```

```
                    String errorCode = e.awsErrorDetails().errorCode();
                    if (errorCode.equals("ResourceNotFoundException")) {
                        logger.log(Level.INFO, "Dataset deleted: {0}",
datasetArn);
                        deleted = true;
                    } else {
                        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                        throw e;
                    }

                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Dataset deleted: {0} ", datasetArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
delete.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


            // Delete the dataset
            deleteMyDataset(rekClient, datasetArn);

            System.out.println(String.format("Dataset deleted: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Amazon Rekognition Custom Labels 모델 관리

Amazon Rekognition Custom Labels 모델은 새 이미지에서 객체, 장면 및 개념의 존재를 예측하는 수학적 모델입니다. 모델 훈련에 사용된 이미지에서 패턴을 찾아 이를 수행합니다. 이 항목은 모델을 훈련하고, 성능을 평가하고, 개선하는 방법을 보여줍니다. 또한 모델을 사용할 수 있게 만드는 방법과 더 이상 필요하지 않을 때 모델을 삭제하는 방법도 보여줍니다.

주제

- Amazon Rekognition Custom Labels 모델 삭제
- 모델 태그 지정

- 모델 설명(SDK)
- Amazon Rekognition Custom Labels 모델 복사(SDK)

Amazon Rekognition Custom Labels 모델 삭제

Amazon Rekognition Custom Labels 콘솔을 사용하거나 DeleteProjectVersion API를 사용하여 모델을 삭제할 수 있습니다. 실행 중이거나 훈련 중인 모델은 삭제할 수 없습니다. 실행 중인 모델을 중지하려면 StopProjectVersion API를 사용하세요. 자세한 내용은 Amazon Rekognition Custom Labels 모델 중지(SDK) 섹션을 참조하세요. 모델이 훈련 중이면 완료될 때까지 기다린 후 모델을 삭제하세요.

삭제된 모델은 복구할 수 없습니다.

주제

- Amazon Rekognition Custom Labels 모델 삭제(콘솔)
- Amazon Rekognition Custom Labels 모델 삭제(SDK)

Amazon Rekognition Custom Labels 모델 삭제(콘솔)

다음 절차는 프로젝트 세부 정보 페이지에서 모델을 삭제하는 방법을 보여줍니다. 모델의 세부 정보 페이지에서도 모델을 삭제할 수 있습니다.

모델을 삭제하려면(콘솔)

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 사용자 지정 레이블 사용을 선택합니다.
3. 시작하기를 선택합니다.
4. 왼쪽 탐색 창에서 프로젝트를 선택합니다.
5. 삭제하려는 모델이 들어 있는 프로젝트를 선택합니다. 프로젝트 세부 정보 페이지가 열립니다.
6. 모델 항목에서 삭제하려는 모델을 선택합니다.

   > **Note**
   >
   > 모델을 선택할 수 없으면 모델이 실행 중이거나 훈련 중이라서 삭제할 수 없다는 뜻입니다. 상태 필드를 확인하고 실행 중인 모델을 중지한 후 다시 시도하거나 훈련이 완료될 때까지 기다리세요.

7. 모델 삭제를 선택하면 모델 삭제 대화 상자가 표시됩니다.

8. 삭제를 입력하여 삭제를 확인합니다.
9. 삭제를 선택하여 모델을 삭제합니다. 모델 삭제를 완료하는 데 시간이 걸릴 수 있습니다.

> **Note**
>
> 모델 삭제 중에 대화 상자를 닫아도 모델은 삭제됩니다.

Amazon Rekognition Custom Labels 모델 삭제(SDK)

DeleteProjectVersion을 호출하고 삭제할 모델의 Amazon 리소스 이름(ARN)을 제공하여 Amazon Rekognition Custom Labels 모델을 삭제합니다. Amazon Rekognition Custom Labels 콘솔의 모델 세부 정보 페이지에 있는 모델 사용 항목에서 모델 ARN을 가져올 수 있습니다. 또는 DescribeProjectVersions를 호출하고 다음을 제공하세요.

- 모델이 연결된 프로젝트(`ProjectArn`)의 ARN
- 모델의 버전 이름(`VersionNames`)

모델 ARN은 `DescribeProjectVersions` 응답의 ProjectVersionDescription 객체에 있는 `ProjectVersionArn` 필드입니다.

실행 중이거나 훈련 중인 모델은 삭제할 수 없습니다. 모델이 실행 중인지 훈련 중인지 확인하려면 DescribeProjectVersions를 호출하고 모델의 ProjectVersionDescription 객체의 `Status` 필드를 확인하세요. 실행 중인 모델을 중지하려면 StopProjectVersion API를 사용하세요. 자세한 내용은 Amazon Rekognition Custom Labels 모델 중지(SDK) 섹션을 참조하세요. 모델을 삭제하려면 먼저 훈련이 완료될 때까지 기다려야 합니다.

모델을 삭제하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 코드를 사용하여 모델을 삭제하세요.

   AWS CLI

   `project-version-arn`의 값을 삭제하려는 프로젝트의 이름으로 변경합니다.

```
aws rekognition delete-project-version --project-version-arn model_arn \
```

```
  --profile custom-labels-access
```

Python

다음 명령줄 파라미터를 제공하세요.

- `project_arn`: 삭제하려는 모델이 포함된 프로젝트의 ARN
- `model_arn`: 삭제하려는 모델 버전의 ARN

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to delete an existing Amazon Rekognition Custom Labels model.
"""


import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_model(rek_client, project_arn, model_arn):
    """
```

```
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """

    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]

        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)

        logger.info("model deleted: %s", model_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete model - %s: %s",
                         model_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
```

```
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to delete."
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
 delete."
    )


def confirm_model_deletion(model_arn):
    """
    Confirms deletion of the model. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

    start = input("Enter delete to delete your model: ")
    if start == "delete":
        return True
    else:
        return False


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_model_deletion(args.model_arn) is True:
            print(f"Deleting model: {args.model_arn}")

            # Delete the model.
            session = boto3.Session(profile_name='custom-labels-access')
```

```
            rekognition_client = session.client("rekognition")

            delete_model(rekognition_client,
                         args.project_arn,
                         args.model_arn)

            print(f"Finished deleting model: {args.model_arn}")
        else:
            print(f"Not deleting model {args.model_arn}")

    except ClientError as err:
        print(f"Problem deleting model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

- `project_arn`: 삭제하려는 모델이 포함된 프로젝트의 ARN
- `model_arn`: 삭제하려는 모델 버전의 ARN

```
//Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.services.rekognition.RekognitionClient;

import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
public class DeleteModel {

   public static final Logger logger =
 Logger.getLogger(DeleteModel.class.getName());

   public static int findForwardSlash(String modelArn, int n) {

       int start = modelArn.indexOf('/');
       while (start >= 0 && n > 1) {
           start = modelArn.indexOf('/', start + 1);
           n -= 1;
       }
       return start;

   }

   public static void deleteMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn)
           throws InterruptedException {

       try {

           logger.log(Level.INFO, "Deleting model: {0}", projectArn);

           // Delete the model

           DeleteProjectVersionRequest deleteProjectVersionRequest =
 DeleteProjectVersionRequest.builder()
                   .projectVersionArn(modelArn).build();

           DeleteProjectVersionResponse response =
                   rekClient.deleteProjectVersion(deleteProjectVersionRequest);

           logger.log(Level.INFO, "Status: {0}", response.status());

           // Get the model version

           int start = findForwardSlash(modelArn, 3) + 1;
           int end = findForwardSlash(modelArn, 4);

           String versionName = modelArn.substring(start, end);

           Boolean deleted = false;
```

```
            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            // Wait until model is deleted.

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                if
(describeProjectVersionsResponse.projectVersionDescriptions().size()==0) {
                    logger.log(Level.INFO, "Waiting for model deletion: {0}",
modelArn);
                    Thread.sleep(5000);
                } else {
                    deleted = true;
                    logger.log(Level.INFO, "Model deleted: {0}", modelArn);
                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Model deleted: {0}", modelArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn> <model_arn>\n\n"
+ "Where:\n"
                + "   project_arn - The ARN of the project that contains the
model that you want to delete.\n\n"
```

```
                + "   model_version - The ARN of the model that you want to
 delete.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String modelVersion = args[1];

        try {

            RekognitionClient rekClient = RekognitionClient.builder().build();

            // Delete the model
            deleteMyModel(rekClient, projectArn, modelVersion);

            System.out.println(String.format("model deleted: %s",
 modelVersion));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

모델 태그 지정

태그를 사용하여 Amazon Rekognition 모델을 식별, 구성, 검색 및 필터링할 수 있습니다. 각 태그는 사용자 정의 키와 값으로 구성된 레이블입니다. 예를 들어, 모델에 대한 청구를 결정하는 데 도움이 되도록 모델에 `Cost center` 키를 태그하고 적절한 비용 센터 번호를 값으로 추가할 수 있습니다. 자세한 내용은 AWS 리소스에 태그 지정 섹션을 참조하세요.

태그 사용 목적

- 비용 할당 태그를 사용하여 모델의 청구를 추적할 수 있습니다. 자세한 내용은 비용 할당 태그 사용을 참조하세요.
- Identity and Access Management(IAM)를 사용하여 모델에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 리소스 태그를 사용한 AWS 리소스 액세스 제어를 참조하세요.
- 모델 관리를 자동화합니다. 예를 들어, 비용을 절감하기 위해 업무 외 시간에 개발 모델의 가동을 중단하는 자동화된 시작 또는 중지 스크립트를 실행할 수 있습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 실행 섹션을 참조하세요.

Amazon Rekognition 콘솔을 사용하거나 AWS SDK를 사용하여 모델에 태그를 지정할 수 있습니다.

주제

- 모델 태그 지정(콘솔)
- 모델 태그 보기
- 모델 태그 지정(SDK)

모델 태그 지정(콘솔)

Rekognition 콘솔을 사용하여 모델에 태그를 추가하고, 모델에 첨부된 태그를 보고, 태그를 제거할 수 있습니다.

태그 추가 및 삭제

이 절차는 기존 모델에 태그를 추가하거나 기존 모델에서 태그를 제거하는 방법을 설명합니다. 새 모델을 훈련할 때 태그를 새 모델에 추가할 수도 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 훈련 섹션을 참조하세요.

콘솔을 사용하여 기존 모델에 태그를 추가하거나 기존 모델에서 태그를 제거하려면

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.

2. 시작하기를 선택합니다.
3. 탐색 창에서 프로젝트를 선택합니다.
4. 프로젝트 리소스 페이지에서 태그를 지정할 모델이 들어 있는 프로젝트를 선택합니다.
5. 탐색 창에 있는 이전에 선택한 프로젝트에서 모델을 선택합니다.
6. 모델 항목에서 태그를 지정하려는 모델을 선택합니다.
7. 모델의 세부 정보 페이지에서 태그 탭을 선택합니다.
8. 태그 섹션에서 태그 관리를 선택합니다.
9. 태그 관리 페이지에서 새 태그 추가를 선택하세요.
10. 키와 값을 입력합니다.
    a. 키에 키 이름을 입력합니다.
    b. 값에 값을 입력합니다.
11. 태그를 더 추가하려면 9 및 10단계를 반복합니다.
12. (선택 사항) 태그를 제거하려면 제거하려는 태그 옆에 있는 제거를 선택합니다. 이전에 저장한 태그를 제거하는 경우 변경 내용을 저장하면 해당 태그가 제거됩니다.
13. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

모델 태그 보기

Amazon Rekognition 콘솔을 사용하여 모델에 연결된 태그를 볼 수 있습니다.

프로젝트 내 모든 모델에 첨부된 태그를 보려면 AWS SDK를 사용해야 합니다. 자세한 내용은 모델 태그 나열 섹션을 참조하세요.

모델에 연결된 태그를 보려면

1. https://console.aws.amazon.com/rekognition/에서 Amazon Rekognition 콘솔을 엽니다.
2. 시작하기를 선택합니다.
3. 탐색 창에서 프로젝트를 선택합니다.
4. 프로젝트 리소스 페이지에서 태그를 보고 싶은 모델이 들어 있는 프로젝트를 선택합니다.
5. 탐색 창에 있는 이전에 선택한 프로젝트에서 모델을 선택합니다.
6. 모델 항목에서 태그를 보려는 모델을 선택합니다.
7. 모델의 세부 정보 페이지에서 태그 탭을 선택합니다. 태그는 태그 항목에 표시됩니다.

모델 태그 지정(SDK)

AWS SDK를 사용하여 다음을 수행할 수 있습니다.

- 새 모델에 태그 추가
- 기존 모델에 태그 추가
- 모델에 지정된 태그 나열
- 모델에서 태그 제거

다음 AWS CLI 예제의 태그는 다음과 같은 형식입니다.

```
--tags '{"key1":"value1","key2":"value2"}'
```

또는 이 형식을 사용할 수 있습니다.

```
--tags key1=value1,key2=value2
```

AWS CLI 항목을 설치하지 않은 경우 4단계: 및 SDK 설정 AWS CLIAWS 항목을 참조하세요.

새 모델에 태그 추가

CreateProjectVersion 작업을 사용하여 모델을 생성할 때 태그를 추가할 수 있습니다. 배열 입력 매개 변수에 하나 이상의 태그를 지정합니다. Tags

```
aws rekognition create-project-version --project-arn project arn \
  --version-name version_name \
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }' \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

모델 생성 및 훈련에 대한 자세한 정보는 모델 훈련(SDK) 항목을 참조하세요.

기존 모델에 태그 추가

기존 모델에 하나 이상의 태그를 추가하려면 TagResource 작업을 사용합니다. 모델의 Amazon 리소스 이름(ARN)(ResourceArn)과 추가할 태그(Tags)를 지정합니다. 다음 예제는 태그 2개를 추가하는 방법을 보여줍니다.

```
aws rekognition tag-resource --resource-arn resource-arn \
```

```
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

CreateProjectVersion을 호출하여 모델의 ARN을 가져올 수 있습니다.

모델 태그 나열

모델에 연결된 태그를 나열하려면 listTagsForResource 작업을 사용하고 모델의 ARN을 지정하세요 (ResourceArn). 응답은 지정된 모델에 연결된 태그 키 및 값의 맵입니다.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \
  --profile custom-labels-access
```

출력에는 모델에 연결된 태그 목록이 표시됩니다.

```
{
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

프로젝트에서 특정 태그가 있는 모델을 확인하려면 DescribeProjectVersions 항목을 호출하여 모델 목록을 가져오세요. 그런 다음 DescribeProjectVersions 응답에서 각 모델에 ListTagsForResource 항목을 호출하세요. ListTagsForResource의 응답을 검사하여 필요 태그가 있는지 확인하세요.

다음 Python 3 예제는 모든 프로젝트에서 특정 태그 키와 값을 검색하는 방법을 보여줍니다. 결과에는 일치하는 키가 있는 프로젝트 ARN과 모델 ARN이 포함됩니다.

태그 값을 검색하려면

1. 다음 코드를 find_tag.py 이름의 파일에 저장합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""
```

```
import logging
import argparse
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def find_tag_in_projects(rekognition_client, key, value):
    """
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
 key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False

        projects = rekognition_client.describe_projects()
        # Iterate through each project and models within a project.
        for project in projects["ProjectDescriptions"]:
            logger.info("Searching project: %s ...", project["ProjectArn"])

            models = rekognition_client.describe_project_versions(
                ProjectArn=(project["ProjectArn"])
            )

            for model in models["ProjectVersionDescriptions"]:
                logger.info("Searching model %s", model["ProjectVersionArn"])

                tags = rekognition_client.list_tags_for_resource(
                    ResourceArn=model["ProjectVersionArn"]
                )

                logger.info(
                    "\tSearching model: %s for tag: %s value: %s.",
                    model["ProjectVersionArn"],
                    key,
```

```
                    value,
                )
                # Check if tag exists.

                if key in tags["Tags"]:
                    if tags["Tags"][key] == value:
                        found = True
                        logger.info(
                            "\t\tMATCH: Project: %s: model version %s",
                            project["ProjectArn"],
                            model["ProjectVersionArn"],
                        )
                        found_tags.append(
                            {
                                "Project": project["ProjectArn"],
                                "ModelVersion": model["ProjectVersionArn"],
                            }
                        )

        if found is False:
            logger.info("No match for Tag %s with value %s.", key, value)
        return found_tags
    except ClientError as err:
        logger.info("Problem finding tags: %s. ", format(err))
        raise


def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")

    args = parser.parse_args()
    key = args.tag
    value = args.value
```

```
    print(f"Searching your models for tag: {key} with value: {value}.")


    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    # Get tagged models for all projects.
    tagged_models = find_tag_in_projects(rekognition_client, key, value)

    print("Matched models\n--------------")
    if len(tagged_models) > 0:
        for model in tagged_models:
            print(
                "Project: {project}\nModel version: {version}\n".format(
                    project=model["Project"], version=model["ModelVersion"]
                )
            )

    else:
        print("No matches found.")

    print("Done.")


if __name__ == "__main__":
    main()
```

2. 명령 프롬프트에서 다음을 입력합니다. *#*와 *##* 찾으려는 키 이름과 키 값으로 바꿉니다.

```
python find_tag.py key value
```

모델에서 태그 삭제

모델에서 하나 이상의 태그를 제거하려면 UntagResource 작업을 사용합니다. 제거하려는 모델의 ARN(ResourceArn)과 태그 키(Tag-Keys)를 지정합니다.

```
aws rekognition untag-resource --resource-arn resource-arn \
  --tag-keys '["key1","key2"]' \
  --profile custom-labels-access
```

또는 이 형식에서 tag-keys 항목을 지정하세요.

```
--tag-keys key1,key2
```

모델 설명(SDK)

DescribeProjectVersions API를 사용하여 모델의 버전에 대한 정보를 가져올 수 있습니다. VersionName 항목을 지정하지 않으면 DescribeProjectVersions 항목은 프로젝트의 모든 모델 버전에 대한 설명을 반환합니다.

모델을 설명하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 예제 코드를 사용하여 모델 버전을 설명하세요.

   AWS CLI

   project-arn의 값을 설명하려는 프로젝트의 ARN으로 변경합니다. version-name의 값을 설명하려는 모델의 버전으로 변경합니다.

   ```
   aws rekognition describe-project-versions --project-arn project_arn \
     --version-names version_name \
     --profile custom-labels-access
   ```

   Python

   다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

   - project_arn: 설명하고자 하는 모델의 ARN입니다.
   - model_version: 설명할 모델의 버전입니다.

   예: python describe_model.py *project_arn model_version*

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Shows how to describe an Amazon Rekognition Custom Labels model.
   """
   ```

```
import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_model(rek_client, project_arn, version_name):
    """
    Describes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the prject that contains the model.
    :param version_name: The version name of the model that you want to
 describe.
    """

    try:
        # Describe the model
        logger.info("Describing model: %s for project %s",
                    version_name, project_arn)

        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print(f"Created: {str(model['CreationTimestamp'])} ")
            print(f"ARN: {str(model['ProjectVersionArn'])} ")
            if 'BillableTrainingTimeInSeconds' in model:
                print(
                    f"Billing training time (minutes):
 {str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
                print(
                    f"\tSummary location: s3://{evaluation_results['Summary']
['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

            if 'ManifestSummary' in model:
                print(
```

```
                    f"Manifest summary location: s3://{model['ManifestSummary']
['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
            if 'OutputConfig' in model:
                print(
                    f"Training output location: s3://{model['OutputConfig']
['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
            if 'MinInferenceUnits' in model:
                print(
                    f"Minimum inference units:
 {str(model['MinInferenceUnits'])}")
            if 'MaxInferenceUnits' in model:
                print(
                    f"Maximum Inference units:
 {str(model['MaxInferenceUnits'])}")

            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])

    except ClientError as err:
        logger.exception(
            "Couldn't describe model: %s", err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
 describe."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
```

```
    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Describing model: {args.version_name} for project
 {args.project_arn}.")

        # Describe the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_model(rekognition_client, args.project_arn,
                       args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
 {args.project_arn}.")

    except ClientError as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- project_arn: 설명하고자 하는 모델의 ARN입니다.
- model_version: 설명할 모델의 버전입니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.EvaluationResult;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeModel {

    public static final Logger logger =
 Logger.getLogger(DescribeModel.class.getName());

    public static void describeMyModel(RekognitionClient rekClient, String
 projectArn, String versionName) {

        try {

            // If a single version name is supplied, build request argument

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
 null;

            if (versionName == null) {
                describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder().projectArn(projectArn)
```

```
                        .build();
            } else {
                describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .versionNames(versionName).build();
            }

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
rekClient
                    .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {

                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());

                if (projectVersionDescription.billableTrainingTimeInSeconds() !=
null) {
                    System.out.println(
                            "Billable minutes: " +
(projectVersionDescription.billableTrainingTimeInSeconds() / 60));
                }

                if (projectVersionDescription.evaluationResult() != null) {
                    EvaluationResult evaluationResult =
projectVersionDescription.evaluationResult();

                    System.out.println("F1 Score: " +
evaluationResult.f1Score());
                    System.out.println("Summary location: s3://" +
evaluationResult.summary().s3Object().bucket() + "/"
                            + evaluationResult.summary().s3Object().name());
                }

                if (projectVersionDescription.manifestSummary() != null) {
                    GroundTruthManifest manifestSummary =
projectVersionDescription.manifestSummary();
```

```
                    System.out.println("Manifest summary location: s3://" +
 manifestSummary.s3Object().bucket() + "/"
                            + manifestSummary.s3Object().name());

                }

                if (projectVersionDescription.outputConfig() != null) {
                    OutputConfig outputConfig =
 projectVersionDescription.outputConfig();
                    System.out.println(
                            "Training output: s3://" + outputConfig.s3Bucket() +
 "/" + outputConfig.s3KeyPrefix());
                }

                if (projectVersionDescription.minInferenceUnits() != null) {
                    System.out.println("Min inference units: " +
 projectVersionDescription.minInferenceUnits());
                }

                System.out.println();

            }

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String args[]) {

        String projectArn = null;
        String versionName = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 models you want to describe.\n\n"
                + "   version_name - (optional) The version name of the model
 that you want to describe. \n\n"
                + "                  If you don't specify a value, all model
 versions are described.\n\n";
```

```
        if (args.length > 2 || args.length == 0) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];

        if (args.length == 2) {
            versionName = args[1];
        }

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the model
            describeMyModel(rekClient, projectArn, versionName);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Amazon Rekognition Custom Labels 모델 복사(SDK)

CopyProjectVersion 작업을 사용하여 Amazon Rekognition Custom Labels 모델 버전을 원본 Amazon Rekognition Custom Labels 프로젝트에서 대상 프로젝트로 복사할 수 있습니다. 대상 프로젝트는 다른 AWS 계정이나 같은 AWS 계정에 있을 수 있습니다. 일반적인 시나리오는 테스트된 모델을 개발 AWS 계정에서 프로덕션 AWS 계정으로 복사하는 것입니다.

또는 소스 데이터 세트를 사용하여 대상 계정에서 모델을 훈련할 수도 있습니다. 이 `CopyProjectVersion` 작업을 사용하면 다음과 같은 이점이 있습니다.

- 모델 동작이 일관적입니다. 모델 훈련은 비결정적이며, 동일한 데이터 세트로 훈련된 두 모델이 동일한 예측을 내린다고 보장할 수 없습니다. `CopyProjectVersion` 항목을 사용하여 모델을 복사하면 복사된 모델의 동작이 소스 모델과 일치하는지 확인할 수 있으므로 모델을 다시 테스트할 필요가 없습니다.
- 모델 훈련이 필요하지 않습니다. 모델을 성공적으로 훈련할 때마다 비용이 청구되므로 이렇게 하면 비용이 절약됩니다.

모델을 다른 AWS 계정에 복사하려면 대상 AWS 계정에 Amazon Rekognition Custom Labels 프로젝트가 있어야 합니다. 프로젝트 생성에 대한 자세한 내용은 프로젝트 생성 항목을 참조하세요. 대상 AWS 계정에서 프로젝트를 생성해야 합니다.

프로젝트 정책은 복사하려는 모델 버전에 대한 복사 권한을 설정하는 리소스 기반 정책입니다. 대상 프로젝트가 원본 프로젝트와 다른 AWS 계정에 있는 경우 프로젝트 정책을 사용해야 합니다.

동일한 계정 내에서 모델 버전을 복사할 때는 프로젝트 정책을 사용할 필요가 없습니다. 하지만 이러한 리소스를 더 잘 제어하려면 계정 간 프로젝트에 프로젝트 정책을 사용하도록 선택할 수 있습니다.

PutProjectPolicy 작업을 호출하여 프로젝트 정책을 소스 프로젝트에 연결합니다.

다른 AWS 리전의 프로젝트에 모델을 복사하는 데는 `CopyProjectVersion` 항목을 사용할 수 없습니다. 또한 Amazon Rekognition Custom Labels 콘솔에서는 모델을 복사할 수 없습니다. 이 경우 소스 모델을 훈련하는 데 사용된 데이터 세트를 사용하여 대상 프로젝트에서 모델을 훈련할 수 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 훈련 섹션을 참조하세요.

소스 프로젝트에서 대상 프로젝트로 모델을 복사하려면 다음을 수행하세요.

모델을 복사하려면

1. 프로젝트 정책 문서를 생성합니다.
2. 프로젝트 정책을 소스 프로젝트에 연결합니다.
3. `CopyProjectVersion` 작업으로 모델을 복사합니다.

프로젝트에서 프로젝트 정책을 제거하려면 DeleteProjectPolicy를 호출하세요. 프로젝트에 첨부된 프로젝트 정책 목록을 가져오려면 ListProjectPolicies를 호출하세요.

주제

- 프로젝트 정책 문서 생성
- 프로젝트 정책(SDK) 연결
- 모델 복사(SDK)
- 프로젝트 정책 나열(SDK)
- 프로젝트 정책 삭제(SDK)

프로젝트 정책 문서 생성

Rekognition Custom Labels는 프로젝트 정책이라고 하는 리소스 기반 정책을 사용하여 모델 버전의 복사 권한을 관리합니다. 프로젝트 정책은 JSON 형식 문서입니다.

프로젝트 정책은 소스 프로젝트에서 대상 프로젝트로 모델 버전을 복사할 수 있는 보안 주체 권한을 허용하거나 거부합니다. 대상 프로젝트가 다른 AWS 계정에 있는 경우 프로젝트 정책이 필요합니다. 대상 프로젝트가 원본 프로젝트와 동일한 AWS 계정에 있고 특정 모델 버전에 대한 액세스를 제한하려는 경우에도 마찬가지입니다. 예를 들어 AWS 계정 내 특정 IAM 역할에 대한 복사 권한을 거부하고자 할 수 있습니다.

다음 예제는 보안 주체 `arn:aws:iam::111111111111:role/Admin` 항목이 모델 버전 `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080` 항목을 복사할 수 있도록 허용합니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::111111111111:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

> **Note**
>
> Action, Resource, Principal, 및 Effect 항목은 프로젝트 정책 문서의 필수 필드입니다. 유일하게 지원되는 action 항목은 rekognition:CopyProjectVersion입니다. NotAction, NotResource, 및 NotPrincipal 항목은 금지된 필드이므로 프로젝트 정책 문서에 없어야 합니다.

프로젝트 정책을 지정하지 않으면 보안 주체에게 CopyProjectVersion 호출 권한을 부여하는 AmazonRekognitionCustomLabelsFullAccess 항목과 같은 ID 기반 정책이 있는 경우에 소스 프로젝트와 같은 AWS 계정의 보안 주체가 여전히 모델을 복사할 수 있습니다.

다음 절차는 프로젝트 정책(SDK) 연결의 Python 예제와 함께 사용할 수 있는 프로젝트 정책 문서 파일을 만듭니다. put-project-policy AWS CLI 명령을 사용하는 경우 프로젝트 정책을 JSON 문자열로 제공합니다.

프로젝트 정책 문서를 생성하려면

1. 텍스트 편집기에서 다음 문서를 생성합니다. 다음 값을 변경합니다.

   - 효과: 복사 권한을 부여하려면 ALLOW 항목을 지정하세요. 복사 권한을 거부하려면 DENY 항목을 지정하세요.
   - 보안 주체: Resource에서 지정한 모델 버전의 권한을 허용하거나 거부하고 싶은 보안 주체로 변경하세요. 예를 들어 다른 AWS 계정의 AWS 계정 보안 주체를 지정할 수 있습니다. 사용할 수 있는 보안 주체에는 제한이 없습니다. 자세한 내용은 보안 주체 지정을 참조하세요.
   - 리소스: 복사 권한을 지정하려는 모델 버전의 Amazon 리소스 이름(ARN) 입니다. 소스 프로젝트 내의 모든 모델 버전에 권한을 부여하려면 다음 arn:aws:rekognition:*region*:*account*:project/*source project*/version/* 형식을 사용하세요.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"ALLOW or DENY",
      "Principal":{
        "AWS":"principal"
      },
```

```
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"Model version ARN"
    }
  ]
}
```

2. 프로젝트 정책을 컴퓨터에 저장합니다.
3. 프로젝트 정책(SDK) 연결의 지침에 따라 프로젝트 정책을 소스 프로젝트에 연결합니다.

프로젝트 정책(SDK) 연결

PutProjectPolicy 작업을 호출하여 Amazon Rekognition Custom Labels 프로젝트에 프로젝트 정책을 연결합니다.

추가하려는 각 프로젝트 정책마다 PutProjectPolicy 항목을 호출하여 여러 프로젝트 정책을 프로젝트에 연결하세요. 최대 5개의 프로젝트 정책을 프로젝트에 연결할 수 있습니다. 더 많은 프로젝트 정책을 추가해야 하는 경우 한도 증가를 요청할 수 있습니다.

고유한 프로젝트 정책을 프로젝트에 처음 첨부할 때는 PolicyRevisionId 입력 파라미터에 수정 ID를 지정하지 마세요. PutProjectPolicy의 응답은 Amazon Rekognition Custom Labels가 사용자를 위해 생성하는 프로젝트 정책의 수정 ID입니다. 수정 ID를 사용하여 프로젝트 정책의 최신 개정 버전을 업데이트하거나 삭제할 수 있습니다. Amazon Rekognition Custom Labels는 프로젝트 정책의 최신 개정 버전만 보관합니다. 프로젝트 정책의 이전 개정 버전을 업데이트하거나 삭제하려고 하면 InvalidPolicyRevisionIdException 오류가 발생합니다.

기존 프로젝트 정책을 업데이트하려면 PolicyRevisionId 입력 파라미터에 프로젝트 정책의 개정 ID를 지정합니다. ListProjectPolicies를 호출하여 프로젝트의 프로젝트 정책에 대한 개정 ID를 가져올 수 있습니다.

프로젝트 정책을 소스 프로젝트에 연결한 후 소스 프로젝트에서 대상 프로젝트로 모델을 복사할 수 있습니다. 자세한 내용은 모델 복사(SDK) 섹션을 참조하세요.

프로젝트에서 프로젝트 정책을 제거하려면 DeleteProjectPolicy를 호출하세요. 프로젝트에 첨부된 프로젝트 정책 목록을 가져오려면 ListProjectPolicies를 호출하세요.

프로젝트 정책을 프로젝트에 연결하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 프로젝트 정책 문서를 생성합니다.

3. 다음 코드를 사용하여 복사하려는 모델 버전이 포함된 신뢰하는 AWS 계정의 프로젝트에 프로젝트 정책을 첨부합니다. 프로젝트 ARN을 가져오려면 DescribeProjects를 호출하세요. 모델 버전 ARN을 가져오려면 DescribeProjectVersions를 호출하세요.

   AWS CLI

   다음 값을 변경합니다.

   - `project-arn` 항목을 복사하려는 모델 버전이 포함된 신뢰하는 AWS 계정의 소스 프로젝트 ARN으로 변경합니다.
   - `policy-name` 항목을 선택한 정책 이름으로 변경합니다.
   - `principal` 항목을 `Model version ARN`에서 지정한 모델 버전에 대한 액세스를 허용하거나 거부하려는 보안 주체로 변경합니다.
   - `project-version-arn` 항목을 복사하려는 모델 버전의 ARN으로 변경합니다.

   기존 프로젝트 정책을 업데이트하려면 `policy-revision-id` 파라미터를 지정하고 원하는 프로젝트 정책의 개정 ID를 제공하세요.

```
aws rekognition put-project-policy \
  --project-arn project-arn \
  --policy-name policy-name \
  --policy-document '{ "Version":"2012-10-17", "Statement":
[{ "Effect":"ALLOW or DENY", "Principal":{ "AWS":"principal" },
 "Action":"rekognition:CopyProjectVersion", "Resource":"project-version-
arn" }]}' \
  --profile custom-labels-access
```

   Python

   다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

   - `project_arn`: 프로젝트 정책을 연결할 소스 프로젝트의 ARN
   - `policy_name`: 사용자가 선택한 정책 이름
   - `project_policy`: 프로젝트 정책 문서가 들어 있는 파일
   - `policy_revision_id`: (선택 사항) 기존 프로젝트 정책의 개정 버전을 업데이트하려면 프로젝트 정책의 개정 ID를 지정하세요.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to attach a project policy to an Amazon Rekognition Custom Labels
 project.
"""

import boto3
import argparse
import logging
import json
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def put_project_policy(rek_client, project_arn, policy_name,
 policy_document_file, policy_revision_id=None):
    """
    Attaches a project policy to an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param policy_name: A name for the project policy.
    :param project_arn: The Amazon Resource Name (ARN) of the source project
    that you want to attach the project policy to.
    :param policy_document_file: The JSON project policy document to
    attach to the source project.
    :param policy_revision_id: (Optional) The revision of an existing policy to
 update.
    Pass None to attach new policy.
    :return The revision ID for the project policy.
    """

    try:

        policy_document_json = ""
        response = None
```

```
        with open(policy_document_file, 'r') as policy_document:
            policy_document_json = json.dumps(json.load(policy_document))

        logger.info(
            "Attaching %s project_policy to project %s.",
            policy_name, project_arn)

        if policy_revision_id is None:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

 PolicyDocument=policy_document_json)

        else:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

 PolicyDocument=policy_document_json,

 PolicyRevisionId=policy_revision_id)

        new_revision_id = response['PolicyRevisionId']

        logger.info(
            "Finished creating project policy %s. Revision ID: %s",
            policy_name, new_revision_id)

        return new_revision_id

    except ClientError as err:
        logger.exception(
            "Couldn't attach %s project policy to project %s: %s }",
            policy_name, project_arn, err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
```

```
        "project_arn",  help="The Amazon Resource Name (ARN) of the project "
        "that you want to attach the project policy to."
    )
    parser.add_argument(
        "policy_name",  help="A name for the project policy."

    )

    parser.add_argument(
        "project_policy",  help="The file containing the project policy JSON"
    )

    parser.add_argument(
        "--policy_revision_id",  help="The revision of an existing policy to
 update. "
        "If you don't supply a value, a new project policy is created.",
        required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Attaching policy to {args.project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")


        # Attach a new policy or update an existing policy.

        response = put_project_policy(rekognition_client,
                                      args.project_arn,
                                      args.policy_name,
```

```
                                        args.project_policy,
                                        args.policy_revision_id)

        print(
            f"project policy {args.policy_name} attached to project
 {args.project_arn}")
        print(f"Revision ID: {response}")

    except ClientError as err:
        print("Problem attaching project policy: %s", err)


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- project_arn: 프로젝트 정책을 연결할 소스 프로젝트의 ARN
- project_policy_name: 사용자가 선택한 정책 이름
- project_policy_document: 프로젝트 정책 문서가 들어 있는 파일
- project_policy_revision_id: (선택 사항) 기존 프로젝트 정책의 개정 버전을 업데이트하려면 프로젝트 정책의 개정 ID를 지정하세요.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import
 software.amazon.awssdk.services.rekognition.model.PutProjectPolicyRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class PutProjectPolicy {

    public static final Logger logger =
 Logger.getLogger(PutProjectPolicy.class.getName());

    public static void putMyProjectPolicy(RekognitionClient rekClient, String
 projectArn, String projectPolicyName,
             String projectPolicyFileName, String projectPolicyRevisionId)
 throws IOException {

        try {

            Path filePath = Path.of(projectPolicyFileName);



            String policyDocument = Files.readString(filePath);

            String[] logArguments = new String[] { projectPolicyFileName,
 projectPolicyName };

            PutProjectPolicyRequest putProjectPolicyRequest = null;

            logger.log(Level.INFO, "Attaching Project policy: {0} to project:
 {1}", logArguments);

            // Attach the project policy.

            if (projectPolicyRevisionId == null) {
                putProjectPolicyRequest =
 PutProjectPolicyRequest.builder().projectArn(projectArn)

 .policyName(projectPolicyName).policyDocument(policyDocument).build();
            } else {
                putProjectPolicyRequest =
 PutProjectPolicyRequest.builder().projectArn(projectArn)

 .policyName(projectPolicyName).policyRevisionId(projectPolicyRevisionId)
                        .policyDocument(policyDocument)
```

```
                        .build();
            }

            rekClient.putProjectPolicy(putProjectPolicyRequest);

            logger.log(Level.INFO, "Attached Project policy: {0} to project:
{1}", logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: "
                + "<project_arn> <project_policy_name> <policy_document>
<project_policy_revision_id>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to
attach the project policy to.\n\n"
                + "   project_policy_name - A name for the project policy.\n\n"
                + "   project_policy_document - The file name of the project
policy.\n\n"
                + "   project_policy_revision_id - (Optional) The revision ID of
the project policy that you want to update.\n\n";

        if (args.length < 3 || args.length > 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyDocument = args[2];
        String projectPolicyRevisionId = null;

        if (args.length == 4) {
            projectPolicyRevisionId = args[3];
        }
```

```
        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Attach the project policy.
            putMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyDocument,
                    projectPolicyRevisionId);

            System.out.println(
                    String.format("project policy %s: attached to project: %s",
 projectPolicyName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (IOException intError) {
            logger.log(Level.SEVERE, "Exception while reading policy document:
 {0}", intError.getMessage());
            System.exit(1);
        }

    }

}
```

4. 모델 복사(SDK)의 지침에 따라 모델 버전을 복사하세요.

모델 복사(SDK)

CopyProjectVersion API를 사용하여 소스 프로젝트에서 대상 프로젝트로 모델 버전을 복사할 수 있습니다. 대상 프로젝트는 다른 AWS 계정에 있을 수 있지만 동일한 AWS 리전이어야 합니다. 대상 프

로젝트가 다른 AWS 계정에 있는 경우(또는 AWS 계정 내에서 복사한 모델 버전에 대해 특정 권한을 부여하려는 경우) 프로젝트 정책을 소스 프로젝트에 연결해야 합니다. 자세한 내용은 프로젝트 정책 문서 생성 섹션을 참조하세요. `CopyProjectVersion` API를 사용하려면 Amazon S3 버킷에 대한 액세스 권한이 필요합니다.

복사된 모델에는 원본 모델의 훈련 결과가 포함되지만 원본 데이터 세트는 포함되지 않습니다.

적절한 권한을 설정하지 않는 한 소스 AWS 계정은 대상 계정으로 복사한 모델에 대한 소유권이 없습니다.

모델을 복사하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 프로젝트 정책(SDK) 연결의 지침에 따라 프로젝트 정책을 소스 프로젝트에 연결합니다.
3. 모델을 다른 AWS 계정에 복사하는 경우 대상 AWS 계정에 프로젝트가 있는지 확인하세요.
4. 다음 코드를 사용하여 모델 버전을 대상 프로젝트에 복사하세요.

   AWS CLI

   다음 값을 변경합니다.

   - `source-project-arn` 항목을 복사하려는 모델 버전이 포함된 소스 프로젝트 ARN으로 변경합니다.
   - `source-project-version-arn` 항목을 복사하려는 모델 버전의 ARN으로 변경합니다.
   - `destination-project-arn` 항목을 모델을 복사할 대상 프로젝트의 ARN으로 변경합니다.
   - `version-name` 항목을 대상 프로젝트에 있는 모델의 버전 이름으로 변경합니다.
   - `bucket` 항목을 소스 모델의 훈련 결과를 복사할 S3 버킷으로 변경합니다.
   - `folder` 항목을 소스 모델의 훈련 결과를 복사하려는 `bucket` 내의 폴더로 변경합니다.
   - (선택 사항) `kms-key-id` 항목을 모델의 AWS Key Management Service 키 ID로 변경합니다.
   - (선택 사항) `key` 항목을 선택한 태그 키로 변경합니다.
   - (선택 사항) `value` 항목을 선택한 태그 값으로 변경합니다.

```
aws rekognition copy-project-version \
```

```
  --source-project-arn source-project-arn \
  --source-project-version-arn source-project-version-arn \
  --destination-project-arn destination-project-arn \
  --version-name version-name \
  --output-config '{"S3Bucket":"bucket","S3KeyPrefix":"folder"}' \
  --kms-key-id arn:myKey \
  --tags '{"key":"key"}' \
  --profile custom-labels-access
```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- source_project_arn: 복사하려는 모델 버전이 포함된 소스 AWS 계정의 소스 프로젝트 ARN
- source_project_version-arn: 복사하려는 소스 AWS 계정의 모델 버전 ARN
- destination_project_arn: 모델을 복사할 대상 프로젝트의 ARN
- destination_version_name: 대상 프로젝트에 있는 모델의 버전 이름
- training_results: 소스 모델의 훈련 결과를 복사할 S3 위치
- (선택 사항) kms_key_id 항목을 모델의 AWS Key Management Service 키 ID로 변경합니다.
- (선택 사항) tag_name 항목을 선택한 태그 키로 변경합니다.
- (선택 사항) tag_value 항목을 선택한 태그 값으로 변경합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_model(
    rekognition_client, source_project_arn, source_project_version_arn,
        destination_project_arn, training_results, destination_version_name):
```

```
    """
    Copies a version of a Amazon Rekognition Custom Labels model.

    :param rekognition_client: A Boto3 Amazon Rekognition Custom Labels client.
    :param source_project_arn: The ARN of the source project that contains the
    model that you want to copy.
    :param source_project_version_arn: The ARN of the model version that you
want
    to copy.
    :param destination_project_Arn: The ARN of the project that you want to copy
the model
    to.
    :param training_results: The Amazon S3 location where training results for
the model
    should be stored.
    return: The model status and version.
    """
    try:
        logger.info("Copying model...%s from %s to %s ",
source_project_version_arn,
                    source_project_arn,
                    destination_project_arn)

        output_bucket, output_folder = training_results.replace(
            "s3://", "").split("/", 1)
        output_config = {"S3Bucket": output_bucket,
                         "S3KeyPrefix": output_folder}

        response = rekognition_client.copy_project_version(
            DestinationProjectArn=destination_project_arn,
            OutputConfig=output_config,
            SourceProjectArn=source_project_arn,
            SourceProjectVersionArn=source_project_version_arn,
            VersionName=destination_version_name
        )

        destination_model_arn = response["ProjectVersionArn"]

        logger.info("Destination model ARN: %s", destination_model_arn)

        # Wait until training completes.
        finished = False
        status = "UNKNOWN"
        while finished is False:
```

```
            model_description =
 rekognition_client.describe_project_versions(ProjectArn=destination_project_arn,
                    VersionNames=[destination_version_name])
            status = model_description["ProjectVersionDescriptions"][0]
["Status"]

            if status == "COPYING_IN_PROGRESS":
                logger.info("Model copying in progress...")
                time.sleep(60)
                continue

            if status == "COPYING_COMPLETED":
                logger.info("Model was successfully copied.")

            if status == "COPYING_FAILED":
                logger.info(
                    "Model copy failed: %s ",
                    model_description["ProjectVersionDescriptions"][0]
["StatusMessage"])

            finished = True
    except ClientError:
        logger.exception("Couldn't copy model.")
        raise
    else:
        return destination_model_arn, status


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "source_project_arn",
        help="The ARN of the project that contains the model that you want to
 copy."
    )

    parser.add_argument(
        "source_project_version_arn",
        help="The ARN of the model version that you want to copy."
    )
```

```
    parser.add_argument(
        "destination_project_arn",
        help="The ARN of the project which receives the copied model."
    )

    parser.add_argument(
        "destination_version_name",
        help="The version name for the model in the destination project."
    )

    parser.add_argument(
        "training_results",
        help="The S3 location in the destination account that receives the
 training results for the copied model."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Copying model version {args.source_project_version_arn} to project
 {args.destination_project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        # Copy the model.

        model_arn, status = copy_model(rekognition_client,
                                       args.source_project_arn,
                                       args.source_project_version_arn,
                                       args.destination_project_arn,
                                       args.training_results,
```

```
                                        args.destination_version_name,
                                        )

        print(f"Finished copying model: {model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        print(f"Problem copying model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- source_project_arn: 복사하려는 모델 버전이 포함된 소스 AWS 계정의 소스 프로젝트 ARN
- source_project_version-arn: 복사하려는 소스 AWS 계정의 모델 버전 ARN
- destination_project_arn: 모델을 복사할 대상 프로젝트의 ARN
- destination_version_name: 대상 프로젝트에 있는 모델의 버전 이름
- output_bucket: 소스 모델 버전의 훈련 결과를 복사하려는 S3 버킷
- output_folder: 소스 모델 버전의 훈련 결과를 복사하려는 S3 내의 폴더

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CopyModel {

    public static final Logger logger =
 Logger.getLogger(CopyModel.class.getName());

    public static ProjectVersionDescription copyMyModel(RekognitionClient
 rekClient,
            String sourceProjectArn,
            String sourceProjectVersionArn,
            String destinationProjectArn,
            String versionName,
            String outputBucket,
            String outputFolder) throws InterruptedException {

        try {

            OutputConfig outputConfig =
 OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            String[] logArguments = new String[] { versionName,
 sourceProjectArn, destinationProjectArn };

            logger.log(Level.INFO, "Copying model {0} for from project {1} to
 project {2}", logArguments);

            CopyProjectVersionRequest copyProjectVersionRequest =
 CopyProjectVersionRequest.builder()
                    .sourceProjectArn(sourceProjectArn)
                    .sourceProjectVersionArn(sourceProjectVersionArn)
                    .versionName(versionName)
                    .destinationProjectArn(destinationProjectArn)
                    .outputConfig(outputConfig)
                    .build();
```

```
            CopyProjectVersionResponse response =
rekClient.copyProjectVersion(copyProjectVersionRequest);

            logger.log(Level.INFO, "Destination model ARN: {0}",
response.projectVersionArn());
            logger.log(Level.INFO, "Copying model...");

            // wait until copying completes.

            boolean finished = false;

            ProjectVersionDescription copiedModel = null;

            while (Boolean.FALSE.equals(finished)) {
                DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                        .versionNames(versionName)
                        .projectArn(destinationProjectArn)
                        .build();

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    copiedModel = projectVersionDescription;

                    switch (projectVersionDescription.status()) {

                        case COPYING_IN_PROGRESS:
                            logger.log(Level.INFO, "Copying model...");
                            Thread.sleep(5000);
                            continue;

                        case COPYING_COMPLETED:
                            finished = true;
                            logger.log(Level.INFO, "Copying completed");
                            break;
```

```
                        case COPYING_FAILED:
                            finished = true;
                            logger.log(Level.INFO, "Copying failed...");
                            break;

                        default:
                            finished = true;
                            logger.log(Level.INFO, "Unexpected copy status %s",
                                    projectVersionDescription.statusAsString());
                            break;

                    }

                }

            }

            logger.log(Level.INFO, "Finished copying model {0} for from project
{1} to project {2}", logArguments);

            return copiedModel;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String sourceProjectArn = null;
        String sourceProjectVersionArn = null;
        String destinationProjectArn = null;
        String versionName = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: "
                + "<source_project_arn> <source_project_version_arn>
<destination_project_arn> <version_name> <output_bucket> <output_folder>\n\n"
                + "Where:\n"
```

```
                + "   source_project_arn - The ARN of the project that contains
 the model that you want to copy. \n\n"
                + "   source_project_version_arn - The ARN of the project that
 contains the model that you want to copy. \n\n"
                + "   destination_project_arn - The ARN of the destination
 project that you want to copy the model to. \n\n"
                + "   version_name - A version name for the copied model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
 training output. \n\n"
                + "   output_folder - The folder within the bucket that the
 training output is stored in. \n\n";

        if (args.length != 6) {
            System.out.println(USAGE);
            System.exit(1);
        }

        sourceProjectArn = args[0];
        sourceProjectVersionArn = args[1];
        destinationProjectArn = args[2];
        versionName = args[3];
        bucket = args[4];
        location = args[5];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Copy the model.
            ProjectVersionDescription copiedModel = copyMyModel(rekClient,
                    sourceProjectArn,
                    sourceProjectVersionArn,
                    destinationProjectArn,
                    versionName,
                    bucket,
                    location);

            System.out.println(String.format("Model copied: %s Status: %s",
                    copiedModel.projectVersionArn(),
```

```
                    copiedModel.statusMessage()));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

프로젝트 정책 나열(SDK)

ListProjectPolicies 작업을 사용하여 Amazon Rekognition Custom Labels 프로젝트에 연결된 프로젝트 정책을 나열할 수 있습니다.

프로젝트에 연결된 프로젝트 정책을 나열하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 코드를 사용하여 프로젝트 정책을 나열합니다.

   AWS CLI

   `project-arn` 항목을 첨부된 프로젝트 정책을 나열하려는 프로젝트의 Amazon 리소스 이름으로 변경합니다.

   ```
   aws rekognition list-project-policies \
     --project-arn project-arn \
     --profile custom-labels-access
   ```

   Python

   다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- project_arn: 첨부된 프로젝트 정책을 나열하려는 프로젝트의 Amazon 리소스 이름

예: python list_project_policies.py *project_arn*

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to list the project policies in an Amazon Rekogntion Custom Labels
 project.
"""


import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_policy(project_policy):
    """
    Displays information about a Custom Labels project policy.
    :param project_policy: The project policy (ProjectPolicy)
    that you want to display information about.
    """
    print(f"Policy name: {(project_policy['PolicyName'])}")
    print(f"Project Arn: {project_policy['ProjectArn']}")
    print(f"Document: {(project_policy['PolicyDocument'])}")
    print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
    print()



def list_project_policies(rek_client, project_arn):
    """
```

```
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n--------')
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
 NextToken=pagination_token)

            for project in response['ProjectPolicies']:
                display_project_policy(project)

            if 'NextToken' in response:
                pagination_token = response['NextToken']
            else:
                finished = True

        logger.info("Finished listing project policies.")

    except ClientError as err:
        logger.exception(
            "Couldn't list policies for - %s: %s",
            project_arn,err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
```

```
        "project_arn",  help="The Amazon Resource Name of the project for which
 you want to list project policies."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing project policies in: {args.project_arn}")

        # List the project policies.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_project_policies(rekognition_client,
                              args.project_arn)

    except ClientError as err:
        print(f"Problem list project_policies: {err}")


if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- project_arn: 나열하려는 프로젝트 정책이 있는 프로젝트의 ARN

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesRequest;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectPolicy;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class ListProjectPolicies {

    public static final Logger logger =
 Logger.getLogger(ListProjectPolicies.class.getName());

    public static void listMyProjectPolicies(RekognitionClient rekClient, String
 projectArn) {

        try {

            logger.log(Level.INFO, "Listing project policies for project: {0}",
 projectArn);

            // List the project policies.

            Boolean finished = false;
            String nextToken = null;

            while (Boolean.FALSE.equals(finished)) {

                ListProjectPoliciesRequest listProjectPoliciesRequest =
 ListProjectPoliciesRequest.builder()
                        .maxResults(5)
                        .projectArn(projectArn)
                        .nextToken(nextToken)
                        .build();
```

```
                ListProjectPoliciesResponse response =
 rekClient.listProjectPolicies(listProjectPoliciesRequest);

                for (ProjectPolicy projectPolicy : response.projectPolicies()) {

                    System.out.println(String.format("Name: %s",
 projectPolicy.policyName()));
                    System.out.println(String.format("Revision ID: %s\n",
 projectPolicy.policyRevisionId()));

                }

                nextToken = response.nextToken();

                if (nextToken == null) {
                    finished = true;

                }

            }

            logger.log(Level.INFO, "Finished listing project policies for
 project: {0}", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn> \n\n" + "Where:
 \n"
                + "   project_arn - The ARN of the project with the project
 policies that you want to list.\n\n";
        ;

        if (args.length != 1) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // List the project policies.
            listMyProjectPolicies(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

프로젝트 정책 삭제(SDK)

DeleteProjectPolicy 작업을 사용하여 Amazon Rekognition Custom Labels 프로젝트에서 기존 프로젝트 정책의 개정 버전을 삭제할 수 있습니다. 프로젝트에 연결된 프로젝트 정책의 모든 개정 버전을 삭제하려면 ListProjectPolicies를 사용하여 프로젝트에 연결된 각 프로젝트 정책의 개정 ID를 가져오세요. 그런 다음 각 정책 이름에 `DeletePolicy` 항목을 호출합니다.

프로젝트 정책 개정 버전을 삭제하려면(SDK)

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. 다음 코드를 사용하여 프로젝트 정책을 삭제합니다.

DeletePolicy에는 ProjectARN, PolicyName, PolicyRevisionId 항목이 필요합니다. ProjectARN 및 PolicyName 항목은 이 API에 필요합니다. PolicyRevisionId 항목은 선택 사항이지만, 원자성 업데이트 목적으로 넣을 수 있습니다.

AWS CLI

다음 값을 변경합니다.

- policy-name 항목을 삭제하려는 프로젝트 정책의 이름으로 변경합니다.
- policy-revision-id 항목을 삭제하려는 프로젝트 정책의 개정 ID로 변경합니다.
- project-arn 항목을 삭제하려는 프로젝트 정책의 개정 버전이 포함된 프로젝트의 Amazon 리소스 이름으로 변경합니다.

```
aws rekognition delete-project-policy \
    --policy-name policy-name \
    --policy-revision-id policy-revision-id \
    --project-arn project-arn \
  --profile custom-labels-access
```

Python

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- policy-name: 삭제하려는 프로젝트 정책의 이름
- project-arn: 삭제하려는 프로젝트 정책이 포함된 프로젝트의 Amazon 리소스 이름
- policy-revision-id: 삭제하려는 프로젝트 정책의 개정 ID

예: python delete_project_policy.py *policy_name project_arn policy_revision_id*

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
```

```
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to delete a revision of a project policy.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_project_policy(rekognition_client, policy_name,  project_arn,
 policy_revision_id=None):
    """
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revsion ID for the project policy that you
 want to delete.
    :param project_arn: The Amazon Resource Name of the project that contains
 the project policy
    that you want to delete.
    """
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)

        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise
```

```
def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you wany to delete project policy {policy_name} ?\n",
 policy_name)

    delete = input("Enter delete to delete your project policy: ")
    if delete == "delete":
        return True
    else:
        return False


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "policy_name", help="The ARN of the project that contains the project
 policy that you want to delete."
    )

    parser.add_argument(
        "project_arn", help="The ARN of the project project policy you want to
 delete."
    )

    parser.add_argument(
        "--policy_revision_id", help="(Optional) The revision ID of the project
 policy that you want to delete.",
        required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
```

```
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_project_policy_deletion(args.policy_name) is True:
            print(f"Deleting project_policy: {args.policy_name}")

            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            # Delete the project policy.

            delete_project_policy(rekognition_client,
                                  args.policy_name,
                                  args.project_arn,
                                  args.policy_revision_id)

            print(f"Finished deleting project policy: {args.policy_name}")
        else:
            print(f"Not deleting project policy {args.policy_name}")
    except ClientError as err:
        print(f"Couldn't delete project policy in {args.policy_name}: {err}")



if __name__ == "__main__":
    main()
```

Java V2

다음 코드를 사용합니다. 다음 명령줄 파라미터를 제공하세요.

- `policy-name`: 삭제하려는 프로젝트 정책의 이름
- `project-arn`: 삭제하려는 프로젝트 정책이 포함된 프로젝트의 Amazon 리소스 이름
- `policy-revision-id`: 삭제하려는 프로젝트 정책의 개정 ID

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectPolicyRequest;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProjectPolicy {

    public static final Logger logger =
 Logger.getLogger(DeleteProjectPolicy.class.getName());

    public static void deleteMyProjectPolicy(RekognitionClient rekClient, String
 projectArn,
            String projectPolicyName,
            String projectPolicyRevisionId)
            throws InterruptedException {

        try {
            String[] logArguments = new String[] { projectPolicyName,
 projectPolicyRevisionId };

            logger.log(Level.INFO, "Deleting: Project policy: {0} revision:
 {1}", logArguments);

            // Delete the project policy.

            DeleteProjectPolicyRequest deleteProjectPolicyRequest =
 DeleteProjectPolicyRequest.builder()
                    .policyName(projectPolicyName)
                    .policyRevisionId(projectPolicyRevisionId)
```

```
                    .projectArn(projectArn).build();

            rekClient.deleteProjectPolicy(deleteProjectPolicyRequest);

            logger.log(Level.INFO, "Deleted: Project policy: {0} revision: {1}",
 logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>
 <project_policy_name> <project_policy_revision_id>\n\n"
                + "Where:\n"
                + "   project_arn - The ARN of the project that has the project
 policy that you want to delete.\n\n"
                + "   project_policy_name - The name of the project policy that
 you want to delete.\n\n"
                + "   project_policy_revision_id - The revision of the project
 policy that you want to delete.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyRevisionId = args[2];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
```

```
            .build();

            // Delete the project policy.
            deleteMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyRevisionId);

            System.out.println(String.format("project policy deleted: %s
 revision: %s", projectPolicyName,
                    projectPolicyRevisionId));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

예제

이 항목에는 Amazon Rekognition Custom Labels와 함께 사용할 수 있는 예제에 대한 정보가 포함되어 있습니다.

예제	설명
모델 피드백 솔루션	사람의 검증을 통해 새 훈련 데이터 세트를 생성하여 모델을 개선하는 방법을 보여줍니다.
Amazon Rekognition Custom Labels 데모	`DetectCustomLabels` 에 대한 호출 결과를 표시하는 사용자 인터페이스 데모
비디오 분석	`DetectCustomLabels` 항목을 비디오에서 추출한 프레임과 함께 사용하는 방법을 보여줍니다.
AWS Lambda 함수를 사용한 이미지 분석	`DetectCustomLabels` 항목을 Lambda 함수와 함께 사용하는 방법을 보여줍니다.
CSV 파일로 매니페스트 파일 생성	CSV 파일을 사용하여 전체 이미지(분류)와 관련된 객체, 장면 및 개념을 찾는 데 적합한 매니페스트 파일을 생성하는 방법을 보여줍니다.

모델 피드백 솔루션

모델 피드백 솔루션을 사용하면 모델 예측에 피드백을 제공하고 사람의 검증을 통해 개선할 수 있습니다. 용례에 따라 이미지 수가 적은 훈련 데이터 세트로도 목적을 달성할 수 있습니다. 더 정확한 모델을 구축하려면 주석이 달린 더 큰 훈련 세트가 필요할 수 있습니다. 모델 피드백 솔루션을 사용하면 모델 지원을 통해 더 큰 데이터 세트를 만들 수 있습니다.

모델 피드백 솔루션을 설치하고 구성하려면 모델 피드백 솔루션을 참조하세요.

모델을 지속적으로 개선하기 위한 워크플로는 다음과 같습니다.

1. 모델의 첫 번째 버전을 학습시키세요(작은 학습 데이터 세트를 사용할 수도 있음).
2. 모델 피드백 솔루션에 주석이 없는 데이터 세트를 제공하세요.

3. 모델 피드백 솔루션은 현재 모델을 사용합니다. 새 데이터 세트에 주석을 달기 위해 사람의 검증 작업을 시작합니다.
4. 모델 피드백 솔루션은 사람의 피드백을 기반으로 새 모델을 생성하는 데 사용할 매니페스트 파일을 생성합니다.

Amazon Rekognition Custom Labels 데모

Amazon Rekognition Custom Labels 데모는 DetectCustomLabels API를 사용하여 로컬 컴퓨터의 이미지를 분석하는 사용자 인터페이스를 보여줍니다.

애플리케이션은 AWS 계정의 Amazon Rekognition Custom Labels 모델에 대한 정보를 보여줍니다. 실행 모델을 선택한 후 로컬 컴퓨터에서 이미지를 분석할 수 있습니다. 필요한 경우 모델을 시작할 수 있습니다. 모델 실행을 중지할 수도 있습니다. 애플리케이션은 Amazon Cognito, Amazon S3, Amazon CloudFront 등과 같은 다른 AWS 서비스와 통합된 것을 보여줍니다.

자세한 내용은 Amazon Rekognition Custom Labels 데모를 참조하세요.

비디오 분석

다음 예제는 `DetectCustomLabels` 항목과 비디오에서 추출한 프레임을 함께 사용하는 방법을 보여줍니다. 코드는 mov 및 mp4 형식의 비디오 파일을 사용하여 테스트되었습니다.

`DetectCustomLabels` 항목을 캡처한 프레임과 함께 사용

1. 아직 하지 않았다면 AWS CLI 및 AWS SDK를 설치하고 구성하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
2. `rekognition:DetectCustomLabels` 및 `AmazonS3ReadOnlyAccess`의 권한이 있는지 확인하세요. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
3. 다음 예제 코드를 사용하세요. `videoFile`의 값을 비디오 파일 이름으로 변경합니다. `projectVersionArn`의 값을 Amazon Rekognition Custom Labels 모델의 Amazon 리소스 이름(ARN)으로 변경합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to analyze a local video with an Amazon Rekognition Custom Labels model.
```

```
"""
import argparse
import logging
import json
import math
import cv2
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def analyze_video(rek_client, project_version_arn, video_file):
    """
    Analyzes a local video file with an Amazon Rekognition Custom Labels model.
    Creates a results JSON file based on the name of the supplied video file.
    :param rek_client: A Boto3 Amazon Rekognition client.
    :param project_version_arn: The ARN of the Custom Labels model that you want to
 use.
    :param video_file: The video file that you want to analyze.
    """

    custom_labels = []
    cap = cv2.VideoCapture(video_file)
    frame_rate = cap.get(5)  # Frame rate.
    while cap.isOpened():
        frame_id = cap.get(1)  # Current frame number.
        print(f"Processing frame id: {frame_id}")
        ret, frame = cap.read()
        if ret is not True:
            break
        if frame_id % math.floor(frame_rate) == 0:
            has_frame, image_bytes = cv2.imencode(".jpg", frame)

            if has_frame:
                response = rek_client.detect_custom_labels(
                    Image={
                        'Bytes': image_bytes.tobytes(),
                    },
                    ProjectVersionArn=project_version_arn
                )

            for elabel in response["CustomLabels"]:
```

```
                elabel["Timestamp"] = (frame_id/frame_rate)*1000
                custom_labels.append(elabel)

    print(custom_labels)

    with open(video_file + ".json", "w", encoding="utf-8") as f:
        f.write(json.dumps(custom_labels))

    cap.release()


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_version_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "video_file", help="The local path to the video that you want to analyze."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:
        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        analyze_video(rekognition_client,
                      args.project_version_arn, args.video_file)

    except ClientError as err:
```

```
        print(f"Couldn't analyze video: {err}")


if __name__ == "__main__":
    main()
```

AWS Lambda 함수를 사용한 이미지 분석

AWS Lambda은 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스입니다. 예를 들어, 애플리케이션 코드를 호스팅할 서버를 생성할 필요 없이 모바일 애플리케이션에서 제출된 이미지를 분석할 수 있습니다. 다음 지침은 DetectCustomLabels를 호출하는 Lambda 함수를 Python에서 생성하는 방법을 보여줍니다. 함수는 제공된 이미지를 분석하고 이미지에서 찾은 레이블 목록을 반환합니다. 지침에는 Amazon S3 버킷의 이미지 또는 로컬 컴퓨터에서 제공된 이미지를 사용하여 Lambda 함수를 호출하는 방법을 보여주는 예제 Python 코드가 포함되어 있습니다.

주제

- 1단계: AWS Lambda 함수 생성(콘솔)
- 2단계: (선택 사항) 계층 생성(콘솔)
- 3단계: Python 코드 추가(콘솔)
- 4단계: Lambda 함수 테스트

1단계: AWS Lambda 함수 생성(콘솔)

이 단계에서는 빈 AWS 함수와 함수가 `DetectCustomLabels` 작업을 호출하게 해주는 IAM 실행 역할을 생성합니다. 또한 분석용 이미지를 저장하는 Amazon S3 버킷에 대한 액세스 권한도 부여합니다. 그리고 다음에 대한 환경 변수를 지정합니다.

- Lambda 함수에서 사용하려는 Amazon Rekognition Custom Labels 모델
- 모델에 사용하려는 신뢰 한도

나중에 Lambda 함수에 소스 코드와 계층(선택 사항)을 추가합니다.

AWS Lambda 함수를 생성하려면(콘솔)

1. AWS Management Console에 로그인하고 AWS Lambdahttps://console.aws.amazon.com/lambda/에서 콘솔을 엽니다.

2. 함수 생성(Create function)을 선택합니다. 자세한 내용은 콘솔로 Lambda 함수 생성을 참조하세요.
3. 다음과 같은 옵션을 선택하세요.
   - 새로 작성을 선택합니다.
   - 함수 이름의 값을 입력합니다.
   - 런타임에는 Python 3.10을 선택합니다.
4. 함수 생성을 선택하여 AWS Lambda 함수를 생성합니다.
5. 함수 페이지에서 구성 탭을 선택합니다.
6. 환경 변수 창에서 편집을 선택합니다.
7. 다음 환경 변수를 추가합니다. 각 변수에 대해 환경 변수 추가를 선택한 다음 변수 키와 값을 입력합니다.

키	값
MODEL_ARN	Lambda 함수에서 사용하려는 모델의 Amazon 리소스 이름(ARN)입니다. Amazon Rekognition Custom Labels 콘솔의 모델 세부 정보 페이지에 있는 모델 사용 탭에서 모델 ARN을 가져올 수 있습니다.
CONFIDENCE	레이블 예측에 대한 모델 신뢰도의 최소값(0~100) Lambda 함수는 신뢰도 값이 이 값보다 낮은 레이블을 반환하지 않습니다.

8. 저장을 선택하여 환경 변수를 저장합니다.
9. 권한 창의 실행 역할에서 IAM 콘솔의 역할을 열 실행 역할을 선택합니다.
10. 권한 탭에서 권한 추가, 그다음 인라인 정책 생성을 선택합니다.
11. JSON을 선택하고 기존 정책을 다음 정책으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "rekognition:DetectCustomLabels",
            "Resource": "*",
            "Effect": "Allow",
```

```
            "Sid": "DetectCustomLabels"
        }
    ]
}
```

12. 다음을 선택합니다.
13. 정책 세부 정보에 DetectCustomLabels-Access와 같은 정책의 이름을 입력합니다.
14. 정책 생성을 선택합니다.
15. Amazon S3 버킷에 분석용 이미지를 저장하는 경우 10~14단계를 반복합니다.

    a. 11단계에서는 다음 정책을 사용하세요. *##/## ##*를 Amazon S3 버킷 및 분석하려는 이미지의 폴더 경로로 바꿉니다.

    ```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "S3Access",
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::bucket/folder path/*"
            }
        ]
    }
    ```

    b. 13단계에서는 S3Bucket-access와 같은 다른 정책 이름을 선택합니다.

2단계: (선택 사항) 계층 생성(콘솔)

이 단계는 예제를 실행하는 데 필수 사항이 아닙니다. `DetectCustomLabels` 작업은 Python용 AWS SDK(Boto3)의 일부로 기본 Lambda Python 환경에 포함됩니다. Lambda 함수의 다른 부분에서 기본 Lambda Python 환경에 없는 최신 AWS 서비스 업데이트가 필요한 경우, 이 단계를 수행하여 최신 Boto3 SDK 릴리스를 함수에 계층으로 추가하세요.

먼저 Boto3 SDK가 포함된 .zip 파일 아카이브를 생성합니다. 그런 다음 계층을 생성하고 .zip 파일 아카이브를 계층에 추가합니다. 자세한 내용은 Lambda 함수에서 계층 사용을 참조하세요.

계층을 생성하고 추가하려면(콘솔)

1. 명령 프롬프트를 열고 다음 명령을 입력합니다.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. zip 파일(boto3-layer.zip)의 이름을 기록해 두세요. 이 절차의 6단계에서 필요합니다.
3. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
4. 탐색 창에서 계층을 선택합니다.
5. 계층 생성을 선택합니다.
6. Name(이름)과 Description(설명)을 입력합니다.
7. .zip 파일 업로드를 선택한 후 업로드를 선택합니다.
8. 대화 상자에서 이 절차의 1단계에서 만든 .zip 파일 아카이브(boto3-layer.zip)를 선택합니다.
9. 호환되는 런타임은 Python 3.9를 선택하세요.
10. 생성을 선택하여 계층을 생성합니다.
11. 탐색 창 메뉴 아이콘을 선택합니다.
12. 탐색 창에서 함수를 선택합니다.
13. 리소스 목록에서 1단계: AWS Lambda 함수 생성(콘솔) 항목에서 생성한 함수를 선택합니다.
14. 코드 탭을 선택합니다.
15. 계층 영역에서 계층 추가를 선택합니다.
16. 사용자 지정 계층을 선택합니다.
17. 사용자 지정 계층에서 6단계에서 입력한 계층 이름을 선택합니다.
18. 버전에서 계층 버전을 선택합니다. 계층 버전은 1이어야 합니다.
19. 추가를 선택합니다.

3단계: Python 코드 추가(콘솔)

이 단계에서는 Lambda 콘솔 코드 편집기를 사용하여 Lambda 함수에 Python 코드를 추가합니다. 해당 코드는 `DetectCustomLabels` 항목으로 제공된 이미지를 분석하고 이미지에서 찾은 레이블 목록을 반환합니다. 제공된 이미지는 Amazon S3 버킷에 있거나 byte64로 인코딩된 이미지 바이트로 제공될 수 있습니다.

Python 코드를 추가하려면(콘솔)

1. Lambda 콘솔을 사용하지 않는 경우 다음을 수행합니다.

a. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.

b. 1단계: AWS Lambda 함수 생성(콘솔)에서 생성한 Lambda 함수를 엽니다.

2. 코드 탭을 선택합니다.

3. 코드 소스에서 lambda_function.py의 코드를 다음과 같이 바꾸세요.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
An AWS lambda function that analyzes images with an the Amazon Rekognition
Custom Labels model.
"""
import json
import base64
from os import environ
import logging
import boto3

from botocore.exceptions import ClientError

# Set up logging.
logger = logging.getLogger(__name__)

# Get the model ARN and confidence.
model_arn = environ['MODEL_ARN']
min_confidence = int(environ.get('CONFIDENCE', 50))

# Get the boto3 client.
rek_client = boto3.client('rekognition')


def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """
```

```
    try:

        # Determine image source.
        if 'image' in event:
            # Decode the image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = {'Bytes': img_b64decoded}


        elif 'S3Object' in event:
            image = {'S3Object':
                     {'Bucket':  event['S3Object']['Bucket'],
                      'Name': event['S3Object']['Name']}
                     }

        else:
            raise ValueError(
                'Invalid source. Only image base 64 encoded image bytes or S3Object
are supported.')


        # Analyze the image.
        response = rek_client.detect_custom_labels(Image=image,
            MinConfidence=min_confidence,
            ProjectVersionArn=model_arn)

        # Get the custom labels
        labels = response['CustomLabels']

        lambda_response = {
            "statusCode": 200,
            "body": json.dumps(labels)
        }

    except ClientError as err:
        error_message = f"Couldn't analyze image. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message
```

```
            }
        }
        logger.error("Error function %s: %s",
            context.invoked_function_arn, error_message)

    except ValueError as val_error:
        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": "ValueError",
                "ErrorMessage": format(val_error)
            }
        }
        logger.error("Error function %s: %s",
            context.invoked_function_arn, format(val_error))

    return lambda_response
```

4. Lambda 함수를 배포하려면 배포를 선택합니다.

4단계: Lambda 함수 테스트

이 단계에서는 컴퓨터의 Python 코드를 사용하여 로컬 이미지 또는 Amazon S3 버킷의 이미지를 Lambda 함수에 전달합니다. 로컬 컴퓨터에서 전달된 이미지는 6291456바이트보다 작아야 합니다. 이미지가 더 크면 Amazon S3 버킷에 이미지를 업로드하고 이미지에 대한 Amazon S3 경로를 사용하여 스크립트를 호출합니다. Amazon S3 버킷에 이미지 파일을 업로드하는 방법에 대한 자세한 내용은 객체 업로드를 참조하세요.

Lambda 함수를 생성한 AWS 리전과 동일한 리전에서 코드를 실행하는지 확인합니다. Lambda 콘솔에 있는 함수 세부 정보 페이지의 탐색 모음에서 Lambda 함수의 AWS 리전을 볼 수 있습니다.

AWS Lambda 함수가 제한 시간 오류를 반환하는 경우 Lambda 함수 제한 시간 기간을 연장하세요. 자세한 내용은 함수 제한 시간 구성(콘솔)을 참조하세요.

코드에서 Lambda 함수를 간접 호출하는 방법에 대한 자세한 내용은 AWS Lambda 함수 간접 호출을 참조하세요.

Lambda 함수를 테스트하는 방법

1. `lambda:InvokeFunction` 권한이 있는지 확인합니다. 다음 정책을 사용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

Lambda 콘솔의 함수 개요에서 Lambda 함수 함수에 대한 ARN을 가져올 수 있습니다.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가합니다.

- AWS IAM Identity Center의 사용자 및 그룹:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.

- 자격 증명 공급자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(페더레이션)의 지침을 따르세요.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자에 권한 추가(콘솔)의 지침을 따르세요.

2. Python용 AWS SDK를 설치하고 구성합니다. 자세한 내용은 4단계: 및 SDK 설정 AWS CLIAWS 섹션을 참조하세요.
3. 1단계: AWS Lambda 함수 생성(콘솔)의 7단계에서 지정한 모델을 시작합니다.
4. 다음 코드를 `client.py` 이름의 파일에 저장합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
```

```
Purpose
Test code for running the Amazon Rekognition Custom Labels Lambda
function example code.
"""

import argparse
import logging
import base64
import json
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def analyze_image(function_name, image):
    """Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")
```

```
            lambda_payload = {"image": data}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))

    return json.loads(response['Payload'].read().decode())


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "function", help="The name of the AWS Lambda function that you want " \
        "to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")


def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
```

```
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


if __name__ == "__main__":
    main()
```

5. 코드를 실행합니다. 명령줄 인수에 Lambda 함수 이름과 분석하려는 이미지를 제공하세요. 로컬 이미지에 대한 경로 또는 Amazon S3 버킷에 저장된 이미지에 대한 S3 경로를 제공할 수 있습니다. 예제:

   ```
   python client.py function_name s3://bucket/path/image.jpg
   ```

   이미지가 Amazon S3 버킷에 있는 경우, 1단계: AWS Lambda 함수 생성(콘솔)의 15단계에서 지정한 것과 동일한 버킷인지 확인하세요.

   성공하면 이미지에서 찾은 레이블 목록이 출력됩니다. 레이블이 반환되지 않는 경우 1단계: AWS Lambda 함수 생성(콘솔)의 7단계에서 설정한 신뢰도 값을 낮추는 것이 좋습니다.

6. Lambda 함수 사용을 마쳤지만 다른 애플리케이션에서 해당 모델을 사용하지 않는 경우 모델을 중지하세요. 다음에 Lambda 함수를 사용하려는 경우 모델을 시작해야 한다는 점을 기억하세요.

보안

고객이 사용자 지정 레이블을 탐지하는 데 사용하는 프로젝트, 모델 및 DetectCustomLabels 작업의 관리를 보호할 수 있습니다.

Amazon Rekognition 보안에 대한 자세한 내용은 Amazon Rekognition 보안을 참조하세요.

Amazon Rekognition Custom Labels 프로젝트의 보안

자격 증명 기반 정책에 지정된 리소스 수준 권한을 지정하여 Amazon Rekognition 사용자 지정 레이블 프로젝트를 보호할 수 있습니다. 자세한 내용은 자격 증명 기반 정책 및 리소스 기반 정책을 참조하세요.

보호할 수 있는 Amazon Rekognition Custom Labels 리소스는 다음과 같습니다.

리소스	Amazon 리소스 이름 형식
프로젝트	arn:aws:rekognition:*:*:project/*project_name* /datetime
모델	arn:aws:rekognition:*:*:project/*project_name* /version/*name*/datetime

다음 예제 정책은 다음 목적으로 ID 권한을 부여하는 방법을 보여줍니다.

- 모든 프로젝트 설명
- 추론을 위해 특정 모델을 생성, 시작, 중지, 사용
- 프로젝트 생성 특정 모델을 생성, 설명
- 특정 프로젝트 생성 거부

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllResources",
            "Effect": "Allow",
            "Action": "rekognition:DescribeProjects",
```

```
            "Resource": "*"
        },
        {
            "Sid": "SpecificProjectVersion",
            "Effect": "Allow",
            "Action": [
                "rekognition:StopProjectVersion",
                "rekognition:StartProjectVersion",
                "rekognition:DetectCustomLabels",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
       },
        {
            "Sid": "SpecificProject",
            "Effect": "Allow",
            "Action": [
                "rekognition:CreateProject",
                "rekognition:DescribeProjectVersions",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/*"
        },
        {
            "Sid": "ExplicitDenyCreateProject",
            "Effect": "Deny",
            "Action": [
                "rekognition:CreateProject"
            ],
            "Resource": ["arn:aws:rekognition:*:*:project/SampleProject/*"]
        }
    ]
}
```

DetectCustomLabels 보호

사용자 지정 레이블을 탐지하는 데 사용되는 자격 증명은 Amazon Rekognition Custom Labels 모델을 관리하는 자격 증명과 다를 수 있습니다.

정책을 자격 증명에 적용하여 특정 자격 증명의 `DetectCustomLabels` 액세스를 보호할 수 있습니다. 다음 예시에서는 특정 모델에만 `DetectCustomLabels`에 대한 액세스를 제한합니다. 이 자격 증명은 다른 Amazon Rekognition 작업에 액세스할 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectCustomLabels"


            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        }
    ]
}
```

AWS 관리형 정책

Amazon Rekognition Custom Labels에 대한 액세스를 제어하는 데 사용할 수 있는 `AmazonRekognitionCustomLabelsFullAccess` AWS 관리형 정책을 제공합니다. 자세한 내용은 AWS 관리형 정책: AmazonRekognitionCustomLabelsFullAccess를 참조하세요.

Amazon Rekognition Custom Labels 지침 및 할당량

다음 항목은 Amazon Rekognition Custom Labels 사용과 관련된 지침과 할당량을 제공합니다.

지원되는 리전

Amazon Rekognition Custom Labels를 사용할 수 있는 모든 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트를 참조하세요.

할당량

Amazon Rekognition Custom Labels 제한 사항 목록은 다음과 같습니다. 변경 가능한 제한에 대한 자세한 내용은 AWS 서비스 제한을 참조하세요. 제한을 변경하려면 Create Case(사례 생성)을 참조하세요.

훈련

- 지원되는 파일 형식은 PNG 및 JPEG 이미지 형식입니다.
- 모델 버전의 최대 훈련 데이터 세트 수는 1개입니다.
- 데이터 세트 매니페스트 파일의 최대 크기는 1GB입니다.
- 객체, 장면, 개념(분류) 데이터 세트당 최소 고유 레이블 수는 2개입니다.
- 객체 위치 파악(탐지) 데이터 세트당 최소 고유 레이블 수는 1개입니다.
- 매니페스트당 최대 고유 레이블 수는 250개입니다.
- 레이블당 최소 이미지 수는 1개입니다.
- 객체 위치 파악(탐지) 데이터 세트당 최대 이미지 수는 250,000개입니다.

  아시아 태평양(뭄바이) 및 유럽(런던) AWS 리전의 이미지 한도는 28,000개입니다.
- 객체, 장면 및 개념(분류) 데이터 세트당 최대 이미지 수는 500,000개입니다. 기본값은 250,000입니다. 증가를 요청하려면 사례 생성을 참조하세요.

  아시아 태평양(뭄바이) 및 유럽(런던) AWS 리전의 이미지 한도는 28,000개입니다. 한도 증가는 요청할 수 없습니다.
- 이미지당 최대 레이블 수는 50개입니다.
- 이미지의 최소 경계 상자 수는 0개입니다.

- 이미지의 최대 경계 상자 수는 50개입니다.
- Amazon S3 버킷에 있는 이미지 파일의 최소 이미지 크기는 64픽셀 x 64픽셀입니다.
- Amazon S3 버킷에 있는 이미지 파일의 최대 이미지 크기는 4096픽셀 x 4096픽셀입니다.
- Amazon S3 버킷에 있는 이미지의 최대 파일 크기는 15MB입니다.
- 최대 이미지 종횡비는 20:1입니다.

테스트

- 한 모델 버전의 최대 테스트 데이터 세트 수는 1입니다.
- 데이터 세트 매니페스트 파일의 최대 크기는 1GB입니다.
- 객체, 장면, 개념(분류) 데이터 세트당 최소 고유 레이블 수는 2개입니다.
- 객체 위치 파악(탐지) 데이터 세트당 최소 고유 레이블 수는 1개입니다.
- 데이터 세트당 최대 고유 레이블 수는 250개입니다.
- 레이블당 최소 이미지 수는 0개입니다.
- 레이블당 최대 이미지 수는 1,000개입니다.
- 객체 위치 파악(탐지) 데이터 세트당 최대 이미지 수는 250,000개입니다.

  아시아 태평양(뭄바이) 및 유럽(런던) AWS 리전의 이미지 한도는 7,000개입니다.
- 객체, 장면 및 개념(분류) 데이터 세트당 최대 이미지 수는 500,000개입니다. 기본값은 250,000입니다. 증가를 요청하려면 사례 생성을 참조하세요.

  아시아 태평양(뭄바이) 및 유럽(런던) AWS 리전의 이미지 한도는 7,000개입니다. 한도 증가는 요청할 수 없습니다.
- 매니페스트당 이미지당 최소 레이블 수는 0입니다.
- 매니페스트당 이미지당 최소 레이블 수는 50입니다.
- 매니페스트당 이미지의 최소 경계 상자 수는 0개입니다.
- 매니페스트당 이미지의 최소 경계 상자 수는 50개입니다.
- Amazon S3 버킷에 있는 이미지 파일의 최소 이미지 크기는 64픽셀 x 64픽셀입니다.
- Amazon S3 버킷에 있는 이미지 파일의 최대 이미지 크기는 4096픽셀 x 4096픽셀입니다.
- Amazon S3 버킷에 있는 이미지의 최대 파일 크기는 15MB입니다.
- 지원되는 파일 형식은 PNG 및 JPEG 이미지 형식입니다.
- 최대 이미지 종횡비는 20:1입니다.

감지

- 원시 바이트로 전달되는 최대 이미지 크기는 4MB입니다.
- Amazon S3 버킷에 있는 이미지의 최대 파일 크기는 15MB입니다.
- 입력 이미지 파일(Amazon S3 버킷에 저장되거나 이미지 바이트로 제공)의 최소 이미지 크기는 64픽셀 x 64픽셀입니다.
- 입력 이미지 파일(Amazon S3에 저장되거나 이미지 바이트로 제공)의 최대 이미지 크기는 4096픽셀 x 4096픽셀입니다.
- 지원되는 파일 형식은 PNG 및 JPEG 이미지 형식입니다.
- 최대 이미지 종횡비는 20:1입니다.

모델 복사

- 프로젝트에 연결할 수 있는 최대 프로젝트 정책 수는 5개입니다.
- 대상의 최대 동시 복사 작업 수는 5입니다.

Amazon Rekognition Custom Labels API 참조

Amazon Rekognition Custom Labels API는 Amazon Rekognition API 참조 콘텐츠의 일부로 문서화되어 있습니다. 다음은 Amazon Rekognition Custom Labels API 작업 목록과 해당 Amazon Rekognition API 참조 주제에 대한 링크입니다. 또한 이 문서에 있는 API 참조 링크는 해당 Amazon Rekognition 개발자 가이드 API 참조 주제로 이동합니다. API 사용에 대한 자세한 내용은

이 섹션에서는 콘솔 및 SDK를 통해 Amazon Rekognition 사용자 지정 레이블 모델을 교육하고 사용하는 워크플로의 개요를 제공합니다. AWS

> **Note**
>
> Amazon Rekognition Custom Labels는 이제 프로젝트 내에서 데이터 세트를 관리합니다. 콘솔과 SDK를 사용하여 프로젝트용 데이터세트를 생성할 수 있습니다. AWS 이전에 Amazon

Rekognition Custom Labels를 사용한 적이 있는 경우 이전 데이터 세트를 새 프로젝트와 연결

해야 할 수 있습니다. 자세한 정보는 6단계: (선택 사항) 이전 데이터 세트를 새 프로젝트와 연결섹션을 참조하십시오.

주제

모델 유형 결정

먼저 비즈니스 목표에 따라 훈련할 모델 유형을 결정합니다. 예를 들어 소셜 미디어 게시물에서 로고를 찾거나, 매장 진열대에서 제품을 식별하거나, 조립 라인에서 기계 부품을 분류하도록 모델을 훈련할 수 있습니다.

Amazon Rekognition Custom Labels는 다음 유형의 모델을 훈련할 수 있습니다.

- 객체, 장면 및 개념 찾기

- 객체 위치 찾기

- 브랜드 위치 찾기

Amazon Rekognition Custom Labels는 훈련할 모델 유형을 결정하는 데 도움이 되도록 사용할 수 있는 예제 프로젝트를 제공합니다. 자세한 정보는 Amazon Rekognition Custom Labels 시작하기을 참조하세요.

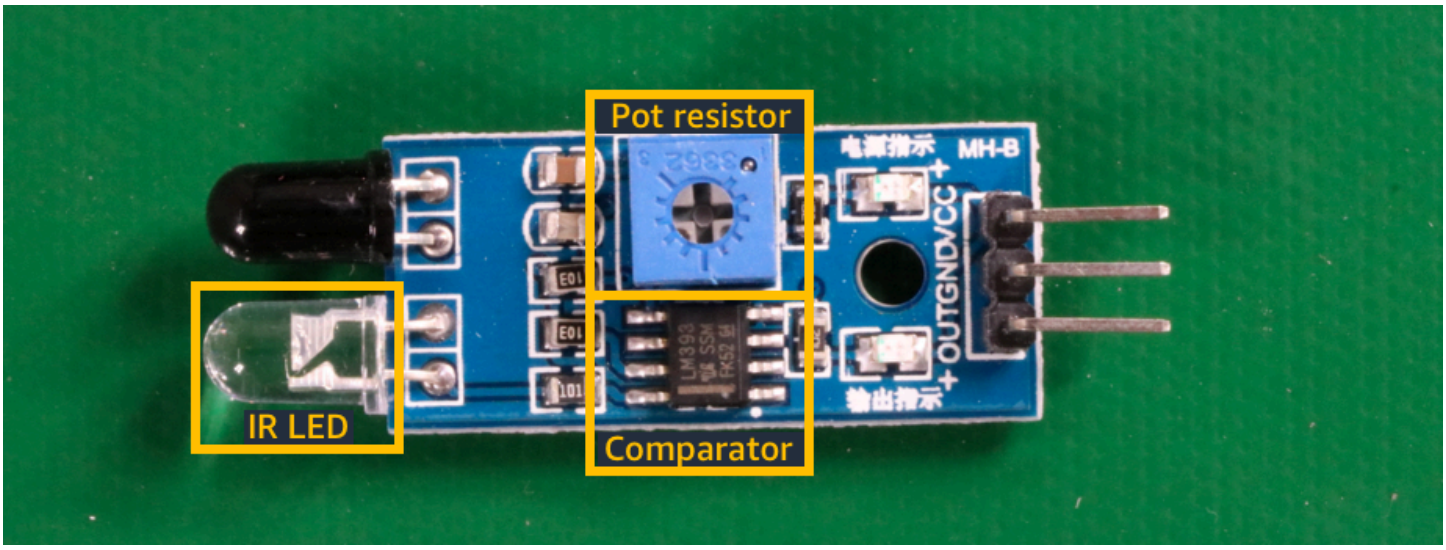
객체, 장면 및 개념 찾기

모델은 전체 이미지와 관련된 객체, 장면 및 개념의 분류를 예측합니다. 예를 들어 이미지에 관광 명소가 포함되어 있는지 여부를 판단하는 모델을 훈련할 수 있습니다. 예제 프로젝트에 관해서는 이미지 분류 항목을 참조하세요. 다음 호수 이미지는 사물, 풍경, 개념을 인식할 수 있는 이미지 종류의 예입니다.

또는 이미지를 여러 범주로 분류하는 모델을 훈련할 수도 있습니다. 예를 들어, 이전 이미지에는 하늘색, 반사 또는 호수와 같은 범주가 있을 수 있습니다. 예제 프로젝트에 관해서는 다중 레이블 이미지 분류 항목을 참조하세요.

객체 위치 찾기

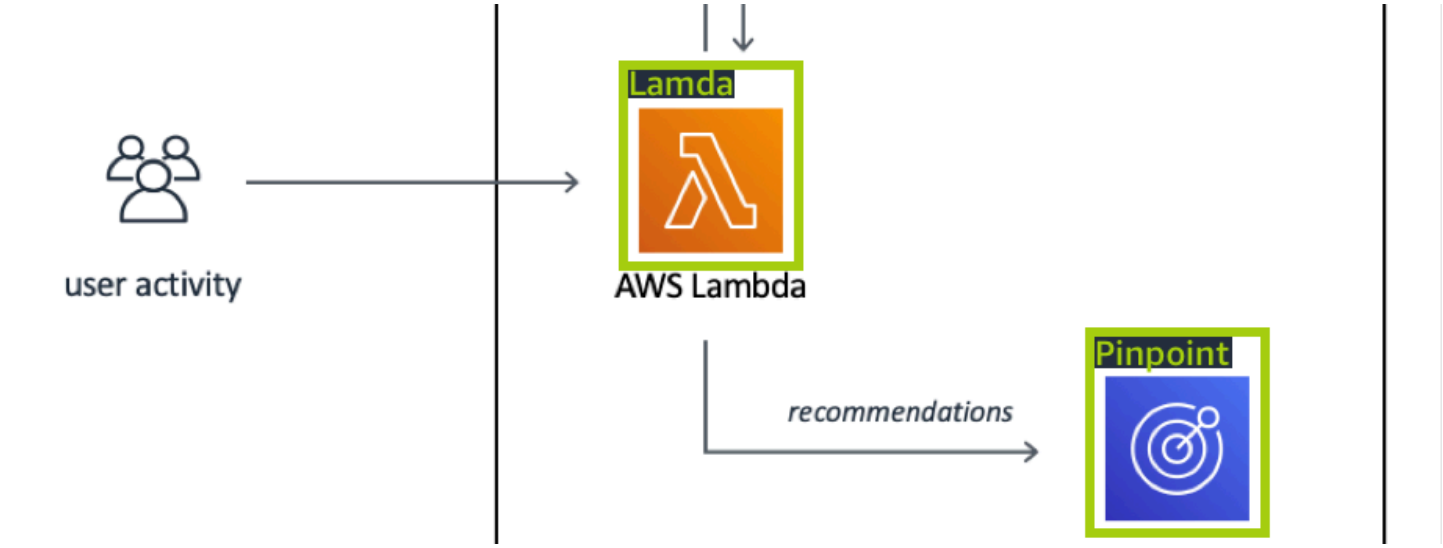
모델은 이미지 상의 객체 위치를 예측합니다. 예측에는 객체 위치에 대한 경계 상자 정보와 경계 상자 내의 객체를 식별하는 레이블이 포함됩니다. 예를 들어, 다음 이미지는 콤퍼레이터 또는 포트 저항기와 같은 회로 기판의 다양한 부분 주위에 있는 경계 상자를 보여줍니다.



객체 위치 파악 예제 프로젝트는 Amazon Rekognition Custom Labels가 레이블이 있는 경계 상자를 사용하여 객체 위치를 찾는 모델을 훈련하는 방법을 보여줍니다.

브랜드 위치 찾기

Amazon Rekognition Custom Labels는 이미지에서 브랜드 위치(예: 로고)를 찾는 모델을 훈련할 수 있습니다. 예측에는 브랜드 위치에 대한 경계 상자 정보와 경계 상자 내의 객체를 식별하는 레이블이 포함됩니다. 예제 프로젝트에 관해서는 브랜드 감지 항목을 참조하세요. 다음 이미지는 모델이 감지할 수 있는 일부 브랜드의 예입니다.



모델 생성

모델을 만드는 단계는 프로젝트 생성, 훈련 및 테스트 데이터 세트 생성, 모델 훈련입니다.

프로젝트 생성

Amazon Rekognition Custom Labels 프로젝트는 모델을 생성하고 관리하는 데 필요한 리소스의 모음입니다. 프로젝트는 다음을 관리합니다.

- 데이터 세트: 모델 훈련에 사용되는 이미지 및 이미지 레이블. 프로젝트에는 훈련 데이터 세트와 테스트 데이터 세트가 있습니다.

- 모델: 사용자의 비즈니스에 필요한 개념, 장면, 객체를 찾을 수 있게 훈련하는 소프트웨어입니다. 한 프로젝트에 여러 버전의 모델을 넣어 둘 수 있습니다.

하나의 프로젝트에는 하나의 용도만 지정해서 사용하시는 것을 권장합니다. 예를 들어 회로판에서 회로판 부품을 찾는 용도로만 사용하는 프로젝트처럼 말입니다.

Amazon Rekognition 사용자 지정 라벨 콘솔과 API를 사용하여 프로젝트를 생성할 수 있습니다. CreateProject 자세한 정보는 프로젝트 생성을 참조하세요.

훈련 및 테스트 데이터 세트 생성

데이터 세트는 해당 이미지를 설명하는 이미지와 레이블의 집합입니다. 프로젝트 내에서 Amazon Rekognition Custom Labels가 모델을 훈련하고 테스트하는 데 사용하는 교육 데이터 세트와 테스트 데이터 세트를 생성합니다.

레이블은 이미지에서 객체, 장면, 개념 또는 객체 주위의 경계 상자를 식별합니다. 레이블은 전체 이미지(이미지 수준)에 지정되거나 이미지에서 객체를 둘러싸는 경계 상자에 지정됩니다.

> **Important**
>
> 데이터 세트의 이미지에 레이블을 지정하는 방식에 따라 Amazon Rekognition Custom Labels가 생성하는 모델 유형이 결정됩니다. 예를 들어 객체, 장면 및 개념을 찾는 모델을 훈련하려면

훈련 및 테스트 데이터 세트의 이미지에 이미지 수준 레이블을 지정합니다. 자세한 정보는 데이터 세트 목적 설정을 참조하세요.

이미지는 PNG 및 JPEG 형식이어야 하며 입력 이미지 권장 사항을 따라야 합니다. 자세한 정보는 이미지 준비을 참조하세요.

훈련 및 테스트 데이터 세트 생성(콘솔)

단일 데이터 세트 또는 별도의 훈련 및 테스트 데이터 세트로 프로젝트를 시작할 수 있습니다. 단일 데이터 세트로 시작하는 경우 Amazon Rekognition Custom Labels는 훈련 중에 데이터 세트를 분할하여 프로젝트에 사용할 훈련 데이터 세트(80%)와 테스트 데이터 세트(20%)를 생성합니다. Amazon Rekognition Custom Labels가 훈련 및 테스트에 사용할 이미지를 결정하게 하려면 단일 데이터 세트로 시작하세요. 훈련, 테스트 및 성능 튜닝을 완벽하게 제어하려면 별도의 훈련 및 테스트 데이터 세트로 프로젝트를 시작하는 것이 좋습니다.

프로젝트의 데이터 세트를 만들려면 다음 방법 중 하나로 이미지를 가져옵니다.

- 로컬 컴퓨터에서 이미지 가져오기
- S3 버킷에서 이미지 가져오기 Amazon Rekognition Custom Labels는 이미지가 포함된 폴더 이름을 사용하여 이미지에 레이블을 지정할 수 있습니다.
- Amazon SageMaker Ground Truth 매니페스트 파일을 가져옵니다.
- 기존 Amazon Rekognition Custom Labels 데이터 세트를 복사합니다.

자세한 정보는 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성을 참조하세요.

이미지를 가져온 위치에 따라 이미지에 레이블이 지정되지 않을 수 있습니다. 예를 들어 로컬 컴퓨터에서 가져온 이미지에는 레이블이 지정되지 않습니다. Amazon SageMaker Ground Truth 매니페스트 파일에서 가져온 이미지에는 레이블이 지정됩니다. Amazon Rekognition Custom Labels 콘솔을 사용하여 레이블을 추가, 변경 및 할당할 수 있습니다. 자세한 정보는 이미지 레이블 지정을 참조하세요.

콘솔을 사용하여 훈련 및 테스트 데이터 세트를 생성하려면 이미지를 사용하여 훈련 및 테스트 데이터 세트 생성 항목을 참조하세요. 훈련 및 테스트 데이터 세트 생성이 포함된 튜토리얼에 관해서는 튜토리얼: 이미지 분류 항목을 참조하세요.

훈련 및 테스트 데이터 세트 생성(SDK)

훈련 및 테스트 데이터 세트를 만들려면 `CreateDataset` API를 사용합니다. Amazon Sagemaker 형식 매니페스트 파일을 사용하거나 기존 Amazon Rekognition Custom Labels 데이터세트를 복사하여 데이터 세트를 생성할 수 있습니다. 자세한 내용은 훈련 및 테스트 데이터 세트 생성(SDK) 섹션을 참조하세요. 필요한 경우 직접 매니페스트 파일을 생성할 수 있습니다. 자세한 정보는 the section called “매니페스트 파일 생성”을 참조하세요.

모델 훈련하기

훈련 데이터 세트를 사용하여 모델을 훈련하세요. 모델을 훈련할 때마다 새 버전의 모델이 생성됩니다. Amazon Rekognition Custom Labels는 훈련 중에 훈련된 모델의 성능을 테스트합니다. 그 결과를 사용하여 모델을 평가하고 개선할 수 있습니다. 훈련을 완료하는 데 시간이 걸립니다. 모델 훈련을 성공적으로 완료한 경우에만 비용이 청구됩니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 훈련을 참조하세요. 모델 훈련이 실패할 경우 Amazon Rekognition Custom Labels는 사용할 수 있는 디버깅 정보를 제공합니다. 자세한 정보는 실패한 모델 훈련 디버깅을 참조하세요.

모델 훈련(콘솔)

콘솔을 사용하여 모델을 훈련하려면 모델 훈련(콘솔) 항목을 참조하세요.

모델 훈련(SDK)

버전을 호출하여 Amazon Rekognition 사용자 지정 레이블 모델을 학습시킵니다. CreateProject 자세한 정보는 모델 훈련(SDK)을 참조하세요.

모델 개선

Amazon Rekognition Custom Labels는 테스트 중에 훈련된 모델을 개선하는 데 사용할 수 있는 평가 지표를 생성합니다.

모델 평가

테스트 중에 만든 성능 지표를 사용하여 모델의 성능을 평가하세요. F1, 정밀도, 재현율과 같은 성능 지표를 통해 훈련된 모델의 성능을 이해하고 프로덕션에 사용할 준비가 되었는지 결정할 수 있습니다. 자세한 정보는 모델 평가를 위한 지표을 참조하세요.

모델 평가(콘솔)

성능 지표를 보려면 평가 지표 액세스(콘솔) 항목을 참조하세요.

모델 평가(SDK)

성능 지표를 가져오려면 Versions를 호출하여 DescribeProject테스트 결과를 얻어야 합니다. 자세한 정보는 Amazon Rekognition Custom Labels 평가 지표에 액세스(SDK)을 참조하세요. 테스트 결과에는 콘솔에서는 이용할 수 없는 지표(예: 분류 결과에 대한 오차 행렬)가 포함됩니다. 테스트 결과는 다음과 같은 형식으로 반환됩니다.

- F1 점수: 모델의 정밀도 및 재현율의 전반적인 성능을 나타내는 단일 값입니다. 자세한 정보는 F1을 참조하세요.

- 요약 파일 위치: 테스트 개요에는 전체 테스트 데이터 세트에 대한 집계된 평가 지표와 각 개별 레이블에 대한 지표가 포함됩니다. `DescribeProjectVersions`는 개요 파일의 S3 버킷 및 폴더 위치를 반환합니다. 자세한 정보는 요약 파일을 참조하세요.
- 평가 매니페스트 스냅샷 위치: 스냅샷에는 신뢰도 등급 및 바이너리 분류 테스트 결과(예: 오탐지)를 비롯한 테스트 결과에 대한 세부 정보가 포함됩니다. `DescribeProjectVersions`는 스냅샷 파일의 S3 버킷 및 폴더 위치를 반환합니다. 자세한 정보는 평가 매니페스트 스냅샷을 참조하세요.

모델 개선

개선이 필요한 경우 훈련 이미지를 더 추가하거나 데이터 세트 레이블 지정을 개선할 수 있습니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 개선을 참조하세요. 또한 모델의 예측에 대한 피드백을 제공하고 이를 사용하여 모델을 개선할 수 있습니다. 자세한 정보는 모델 피드백 솔루션을 참조하세요.

모델 개선(콘솔)

데이터 세트에 이미지를 추가하려면 데이터 세트에 더 많은 이미지 추가 항목을 참조하세요. 레이블을 추가하거나 변경하려면 the section called “이미지 레이블 지정” 항목을 참조하세요.

모델을 재훈련하려면 모델 훈련(콘솔) 항목을 참조하세요.

모델 개선(SDK)

데이터 세트에 이미지를 추가하거나 이미지의 레이블을 변경하려면 `UpdateDatasetEntries` API를 사용하세요. `UpdateDatasetEntries`는 매니페스트 파일에 JSON 라인을 업데이트하거나 추가

합니다. 각 JSON 라인에는 지정된 레이블 또는 경계 상자 정보와 같은 단일 이미지에 대한 정보가 들어 있습니다. 자세한 정보는 더 많은 이미지 추가(SDK)을 참조하세요. 데이터 세트의 항목을 보려면 `ListDatasetEntries` API를 사용하세요.

모델을 재훈련하려면 모델 훈련(SDK) 항목을 참조하세요.

모델 시작

모델을 사용하려면 먼저 Amazon Rekognition Custom Labels 콘솔 또는 `StartProjectVersion` API를 사용하여 모델을 시작하세요. 모델을 실행하는 시간만큼 요금이 부과됩니다. 자세한 정보는 훈련된 모델 실행을 참조하세요.

모델 시작(콘솔)

콘솔을 사용하여 모델을 시작하려면 Amazon Rekognition Custom Labels 모델 시작(콘솔) 항목을 참조하세요.

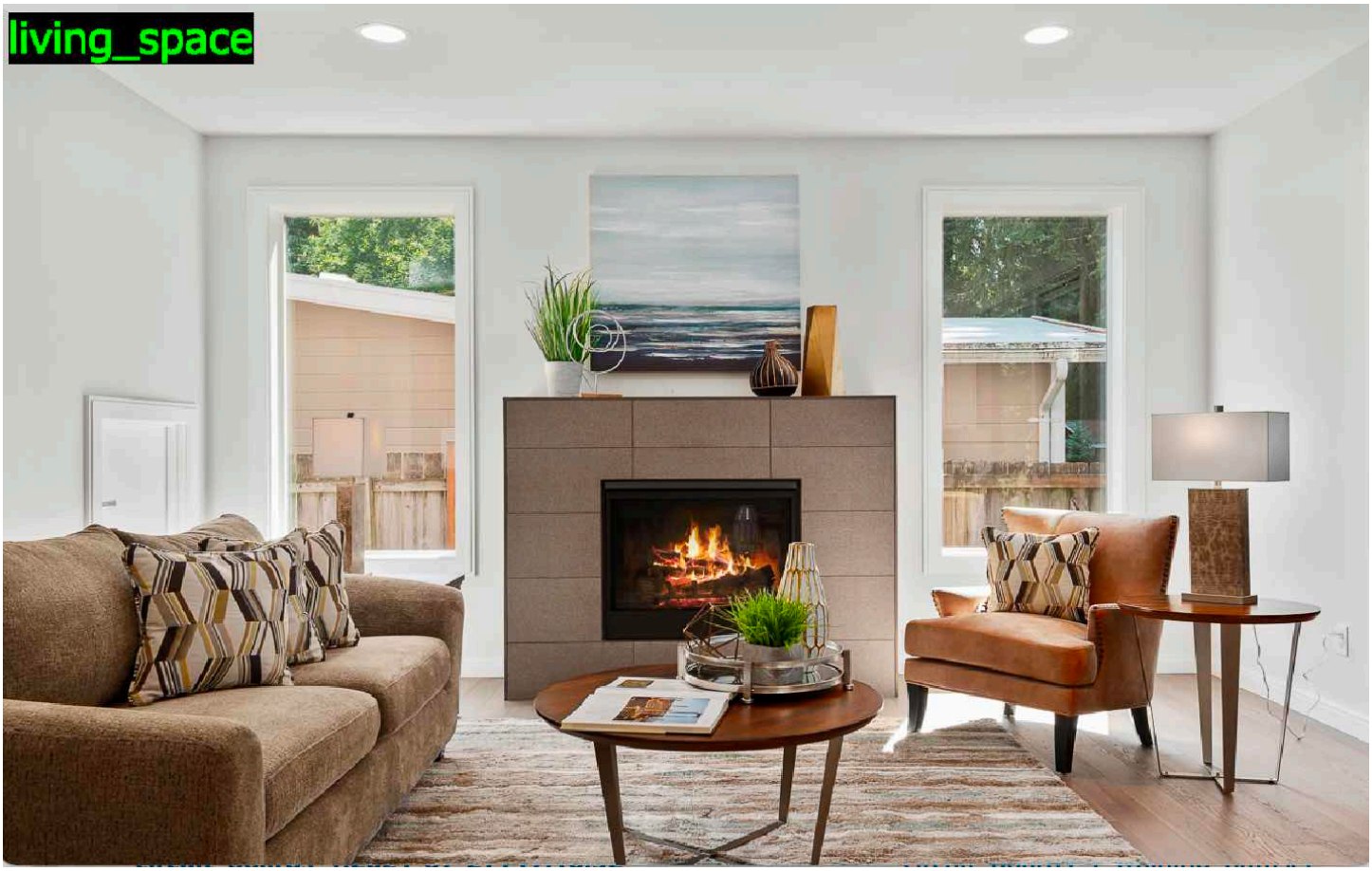
모델 시작

StartProjectVersion을 호출하여 모델을 시작합니다. 자세한 정보는 Amazon Rekognition Custom Labels 모델 시작(SDK)을 참조하세요.

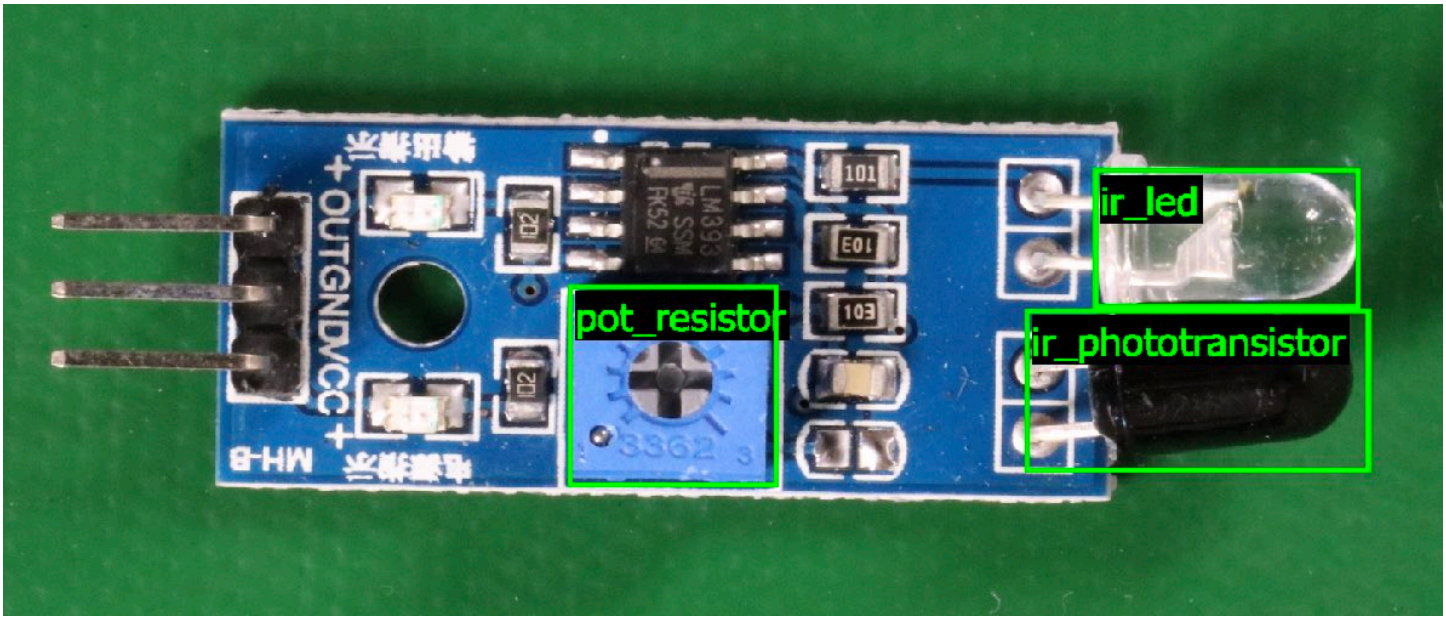
이미지 분석

모델로 이미지를 분석하려면 `DetectCustomLabels` API를 사용합니다. 로컬 이미지 또는 S3 버킷에 저장된 이미지를 지정할 수 있습니다. 작업을 수행하려면 사용하려는 모델의 Amazon 리소스 이름(ARN)도 필요합니다.

모델이 객체, 장면, 개념을 찾은 경우 응답에는 이미지에서 찾은 이미지 수준 레이블 목록이 포함됩니다. 예를 들어 다음 이미지는 방 예제 프로젝트를 사용하여 찾은 이미지 수준 레이블을 보여줍니다.



모델이 객체 위치를 찾은 경우 응답에는 이미지에서 찾은 레이블이 지정된 경계 상자 목록이 포함됩니다. 경계 상자는 이미지에서 객체의 위치를 나타냅니다. 테두리 상자 정보를 사용하여 객체 주위에 테두리 상자를 그릴 수 있습니다. 예를 들어, 다음 이미지는 회로판 예제 프로젝트를 사용하여 찾은 회로판 부품 주위의 경계 상자를 보여줍니다.



자세한 정보는 훈련된 모델을 사용한 이미지 분석을 참조하세요.

모델 중지

모델을 실행하는 시간만큼 요금이 부과됩니다. 모델을 더 이상 사용하지 않는 경우 Amazon Rekognition Custom Labels 콘솔을 사용하거나 `StopProjectVersion` API를 사용하여 모델을 중지하세요. 자세한 정보는 Amazon Rekognition Custom Labels 모델 중지을 참조하세요.

모델 중지(콘솔)

콘솔에서 실행 중인 모델을 중지하려면 Amazon Rekognition Custom Labels 모델 중지(콘솔) 항목을 참조하세요.

모델 중지(SDK)

실행 중인 모델을 중지하려면 Version을 호출하십시오. StopProject 자세한 내용은 Amazon Rekognition Custom Labels 모델 중지(SDK)을(를) 참조하세요.
항목을 참조하세요.

모델 훈련

프로젝트

- CreateProject: 리소스(이미지, 레이블, 모델)와 작업(훈련, 평가, 탐지)을 논리적으로 그룹화한 Amazon Rekognition Custom Labels 프로젝트를 생성합니다.
- 프로젝트 삭제: Amazon Rekognition Custom Labels 프로젝트를 삭제합니다.
- DescribeProjects: 모든 Amazon Rekognition Custom Labels 프로젝트의 목록을 반환합니다.

프로젝트 정책

- PutProjectPolicy: 신뢰할 수 있는 AWS 계정의 Amazon Rekognition Custom Labels 프로젝트에 프로젝트 정책을 연결합니다.
- ListProjectPolicies: 프로젝트에 연결된 프로젝트 정책 목록을 반환합니다.
- DeleteProjectPolicy: 기존 프로젝트 정책을 삭제합니다.

데이터 세트

- CreateDataset: Amazon Rekognition Custom Labels 데이터 세트를 생성합니다.
- DeleteDataset: Amazon Rekognition Custom Labels 데이터 세트를 삭제합니다.
- DescribeDataset: Amazon Rekognition Custom Labels 데이터 세트를 설명합니다.
- DistributeDatasetEntries: 훈련 데이터 세트의 항목(이미지)을 프로젝트의 훈련 데이터 세트 및 테스트 데이터 세트에 배포합니다.
- ListDatasetEntries: Amazon Rekognition Custom Labels 데이터 세트의 항목(이미지) 목록을 반환합니다.
- ListDatasetLabels: Amazon Rekognition Custom Labels 데이터 세트에 할당된 레이블 목록을 반환합니다.
- UpdateDatasetEntries: Amazon Rekognition Custom Labels 데이터 세트에 항목(이미지)을 추가하거나 업데이트합니다.

모델

- CreateProjectVersion: Amazon Rekognition Custom Labels 모델을 훈련합니다.

- CopyProjectVersion: Amazon Rekognition Custom Labels 모델을 복사합니다.
- DeleteProjectVersion: Amazon Rekognition Custom Labels 모델을 삭제합니다.
- DescribeProjectVersions: 특정 프로젝트 내의 모든 Amazon Rekognition Custom Labels 모델 목록을 반환합니다.

태그

- TagResource: Amazon Rekognition Custom Labels 모델에 하나 이상의 키 값 태그를 추가합니다.
- UntagResource: Amazon Rekognition Custom Labels 모델에서 하나 이상의 태그를 제거합니다.

모델 사용

- DetectCustomLabels: 사용자 지정 레이블 모델로 이미지를 분석합니다.
- StartProjectVersion: 사용자 지정 레이블 모델을 시작합니다.
- StopProjectVersion: 사용자 지정 레이블 모델을 중지합니다.

Amazon Rekognition Custom Labels의 문서 기록

다음 표는 Amazon Rekognition Custom Labels 개발자 가이드의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최근 문서 업데이트: 2023년 4월 19일

변경 사항	설명	날짜
모델 지속 시간 주제를 추가했습니다.	실행 시간과 모델이 사용한 추론 단위를 가져오는 방법을 보여줍니다. 자세한 내용은 실행 시간 및 사용된 추론 단위 보고를 참조하세요.	2023년 4월 19일
데이터 세트 콘텐츠가 재구성되었습니다.	매니페스트 파일 생성 콘텐츠를 매니페스트 파일로 이동했습니다. 데이터 세트 변환 주제를 다른 데이터세트 형식을 매니페스트 파일로 변환으로 옮겼습니다.	2023년 2월 20일
AWS WAF에 대한 IAM 지침이 업데이트되었습니다.	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하세요.	2023년 2월 15일
분류 모델의 오차 행렬 보기	Amazon Rekognition Custom Labels 콘솔에는 분류 모델에 대한 오차 행렬이 표시되지 않습니다. 대신 AWS SDK를 사용하여 오차 행렬을 구해서 표시할 수 있습니다. 자세한 내용은 모델의 오차 행렬 보기를 참조하세요.	2023년 1월 4일

Lambda 함수 예제가 업데이트 되었습니다.	Lambda 함수 예제는 이제 로컬 파일 또는 Amazon S3 버킷에서 전달된 이미지를 분석하는 방법을 보여줍니다. 자세한 내용은 AWS Lambda 함수를 사용한 이미지 분석을 참조하세요.	2022년 12월 2일
Amazon Rekognition Custom Labels는 이제 훈련된 모델을 복사할 수 있습니다.	이제 훈련된 모델을 한 AWS 계정에서 동일한 AWS 리전 내 다른 AWS 계정으로 복사할 수 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 복사(SDK)를 참조하세요.	2022년 8월 16일
Amazon Rekognition Custom Labels는 이제 추론 단위를 자동으로 조정할 수 있습니다.	수요 급증에 대응하기 위해 Amazon Rekognition Custom Labels는 이제 모델에서 사용하는 추론 단위 수를 조정할 수 있습니다. 자세한 내용은 훈련된 Amazon Rekognition Custom Labels 모델 실행을 참조하세요.	2022년 8월 16일
CSV 파일에서 매니페스트 파일을 생성	이제 CSV 파일에서 이미지 분류 정보를 읽는 스크립트를 사용하여 매니페스트 파일 생성을 간소화할 수 있습니다. 자세한 내용은 CSV 파일로 매니페스트 파일 생성을 참조하세요.	2022년 2월 2일

Amazon Rekognition Custom Labels는 이제 프로젝트와 함께 데이터 세트를 관리합니다.	프로젝트를 사용하여 모델을 생성하는 데 사용하는 훈련 및 테스트 데이터 세트를 관리할 수 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels의 이해를 참조하세요.	2021년 11월 1일
Amazon Rekognition Custom Labels가 AWS CloudFormation 항목과 통합되었습니다.	AWS CloudFormation 항목을 사용하여 Amazon Rekognition Custom Labels 프로젝트를 프로비저닝하고 구성할 수 있습니다. 자세한 내용은 AWS CloudFormation 항목으로 프로젝트 생성을 참조하세요.	2021년 10월 21일
시작하기 경험이 업데이트되었습니다.	Amazon Rekognition Custom Labels 콘솔에는 이제 튜토리얼 비디오와 예제 프로젝트가 포함되어 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 시작하기를 참조하세요.	2021년 7월 22일
임계값 및 지표 사용에 대한 정보가 업데이트되었습니다.	`MinConfidence` 입력 파라미터를 `DetectCustomLabels` 에 사용하여 원하는 임계값을 설정하는 방법에 대한 정보 자세한 내용은 훈련된 모델을 사용한 이미지 분석을 참조하세요.	2021년 6월 8일
AWS KMS key 지원이 추가되었습니다.	이제 자체 KMS 키를 사용하여 훈련 및 테스트 이미지를 암호화할 수 있습니다. 자세한 내용은 모델 훈련을 참조하세요.	2021년 5월 19일

태그 지정이 추가되었습니다.	Amazon Rekognition Custom Labels가 이제 태그를 지원합니다. 태그를 사용하여 Amazon Rekognition Custom Labels 모델을 식별, 구성, 검색 및 필터링할 수 있습니다. 자세한 내용은 모델 태그 지정을 참조하세요.	2021년 3월 25일
설정 주제가 업데이트되었습니다.	훈련 파일을 암호화하는 방법에 대한 설정 정보가 업데이트되었습니다. 자세한 내용은 5단계: (선택 사항) 훈련 파일 암호화를 참조하세요.	2021년 3월 18일
데이터 세트 복사 주제가 추가되었습니다.	데이터 세트를 다른 AWS 리전에 복사하는 방법에 대한 정보 자세한 내용은 데이터 세트를 다른 AWS 리전에 복사를 참조하세요.	2021년 3월 5일
Amazon SageMaker GroundTruth 다중 레이블 매니페스트 변환 주제를 추가했습니다.	Amazon SageMaker GroundTruth 다중 레이블 형식 매니페스트를 Amazon Rekognition Custom Labels 형식 매니페스트 파일로 변환하는 방법에 대한 정보 자세한 내용은 다중 레이블 SageMaker Ground Truth 매니페스트 파일 변환을 참조하세요.	2021년 2월 22일
모델 훈련을 위한 디버깅 정보를 추가했습니다.	이제 검증 결과 매니페스트를 사용하여 모델 훈련 오류에 대한 심층적인 디버깅 정보를 얻을 수 있습니다. 자세한 내용은 실패한 모델 훈련 디버깅을 참조하세요.	2020년 10월 8일

COCO 변환 정보 및 예제가 추가되었습니다.	COCO 객체 감지 형식 데이터 세트를 Amazon Rekognition Custom Labels 매니페스트 파일로 변환하는 방법에 대한 정보 자세한 내용은 COCO 데이터 세트 변환을 참조하세요.	2020년 9월 2일
Amazon Rekognition Custom Labels는 이제 단일 객체 훈련을 지원합니다.	단일 객체의 위치를 찾는 Amazon Rekognition Custom Labels 모델을 생성하기 위해 이제 레이블이 하나만 요구되는 데이터 세트를 생성할 수 있습니다. 자세한 내용은 경계 상자 그리기를 참조하세요.	2020년 6월 25일
프로젝트 및 모델 삭제 작업이 추가되었습니다.	이제 콘솔과 API를 사용하여 Amazon Rekognition Custom Labels 프로젝트 및 모델을 삭제할 수 있습니다. 자세한 내용은 Amazon Rekognition Custom Labels 모델 삭제 및 Amazon Rekognition Custom Labels 프로젝트 삭제를 참조하세요.	2020년 4월 1일
Java 예제가 추가되었습니다.	프로젝트 생성, 모델 훈련, 모델 실행, 이미지 분석을 다루는 Java 예제가 추가되었습니다.	2019년 12월 13일
새 특성 및 가이드	이것은 Amazon Rekognition Custom Labels 특성과 Amazon Rekognition Custom Labels 개발자 가이드의 첫 번째 릴리스입니다.	2019년 12월 3일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.