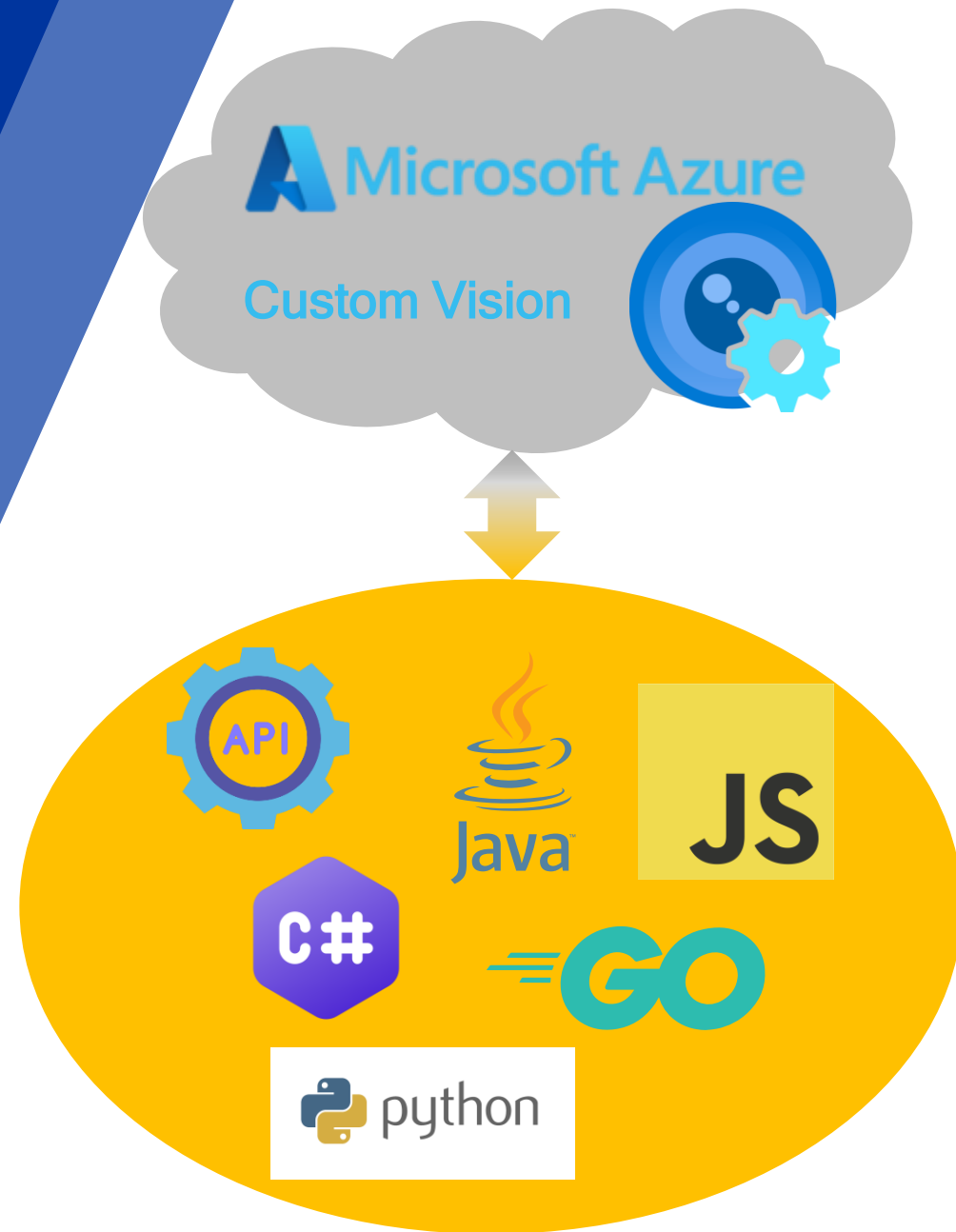


중급과정

커스텀 비전

6차시: 외부 애플리케이션에서 호출을
통한 커스텀 비전 모델 활용 방안



수업 일정

전체 수업은 7회로 구성된다.



- 클라우드와 Azure
- 커스텀 비전



- 개체 감지 AI 모델 – (1)
- 개체 감지의 원리와 이미지 수집



- 개체 감지 AI 모델 – (2)
- 오버더문의 번지 캐릭터 찾기



- 이미지 분류 AI 모델 – (1)
- 암석식별머신을 만들기 위한 문제정의



- 이미지 분류 AI 모델 – (2)
- 암석식별머신 만들기



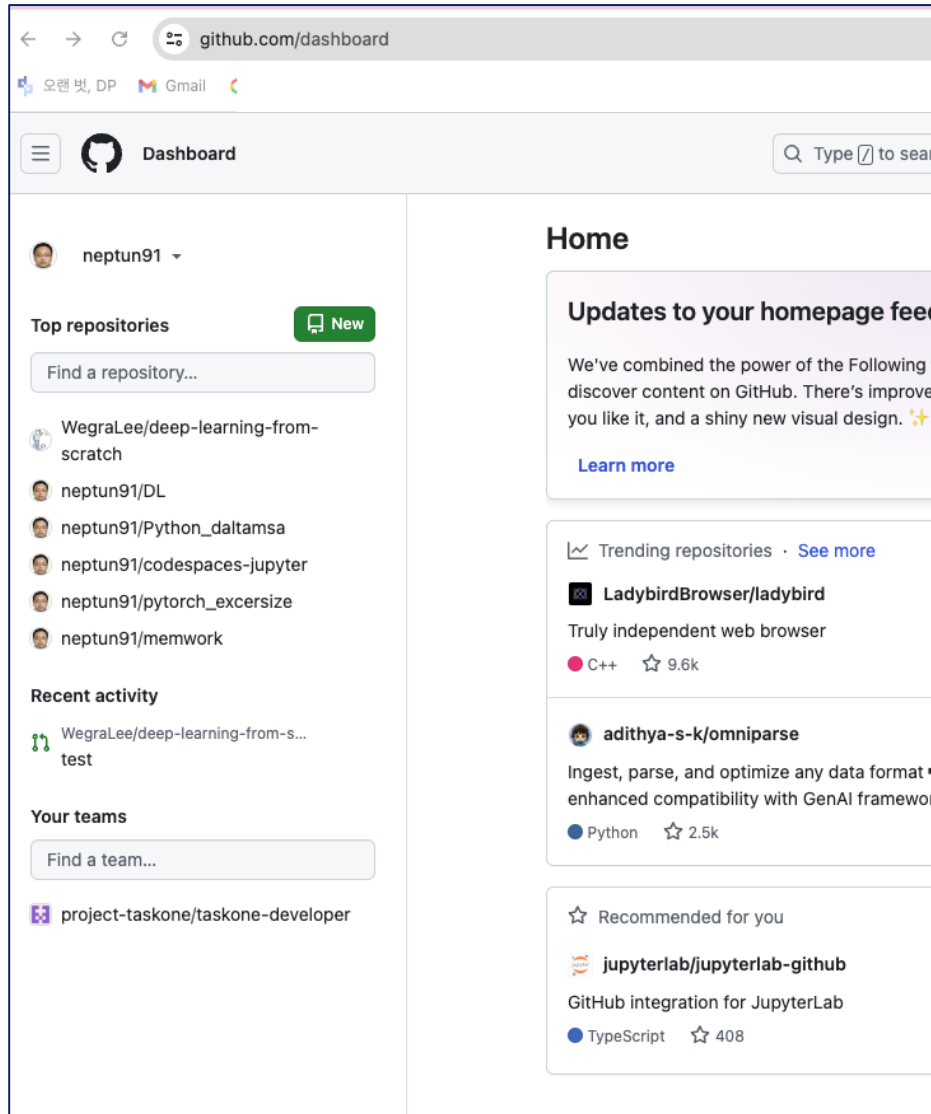
- 외부 애플리케이션에서 호출을 통한 커스텀 비전 모델 활용 방안



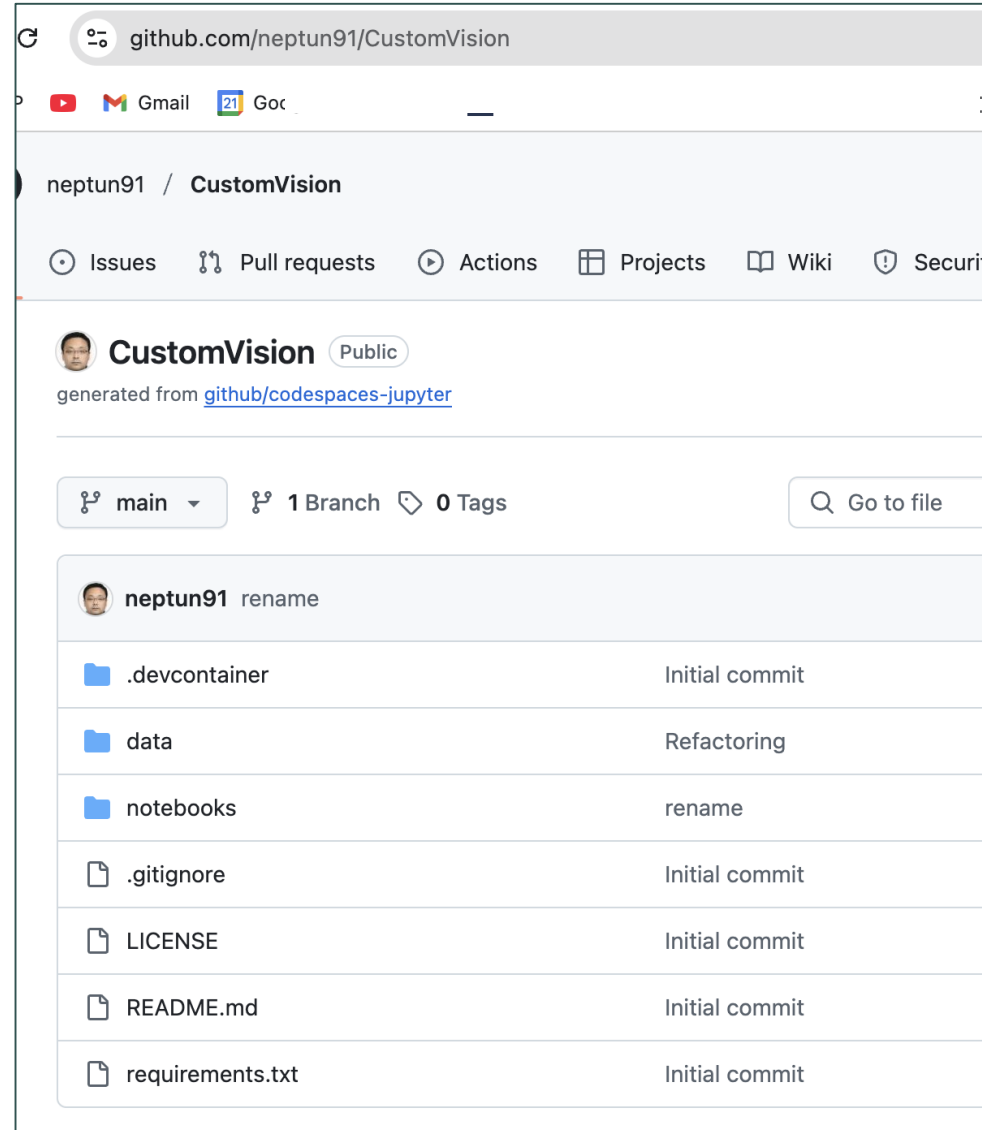
- Gradio를 활용한 커스텀 비전 모델 활용 방안

GitHub codespace 생성(1)

Step1. GitHub 로그인 상태

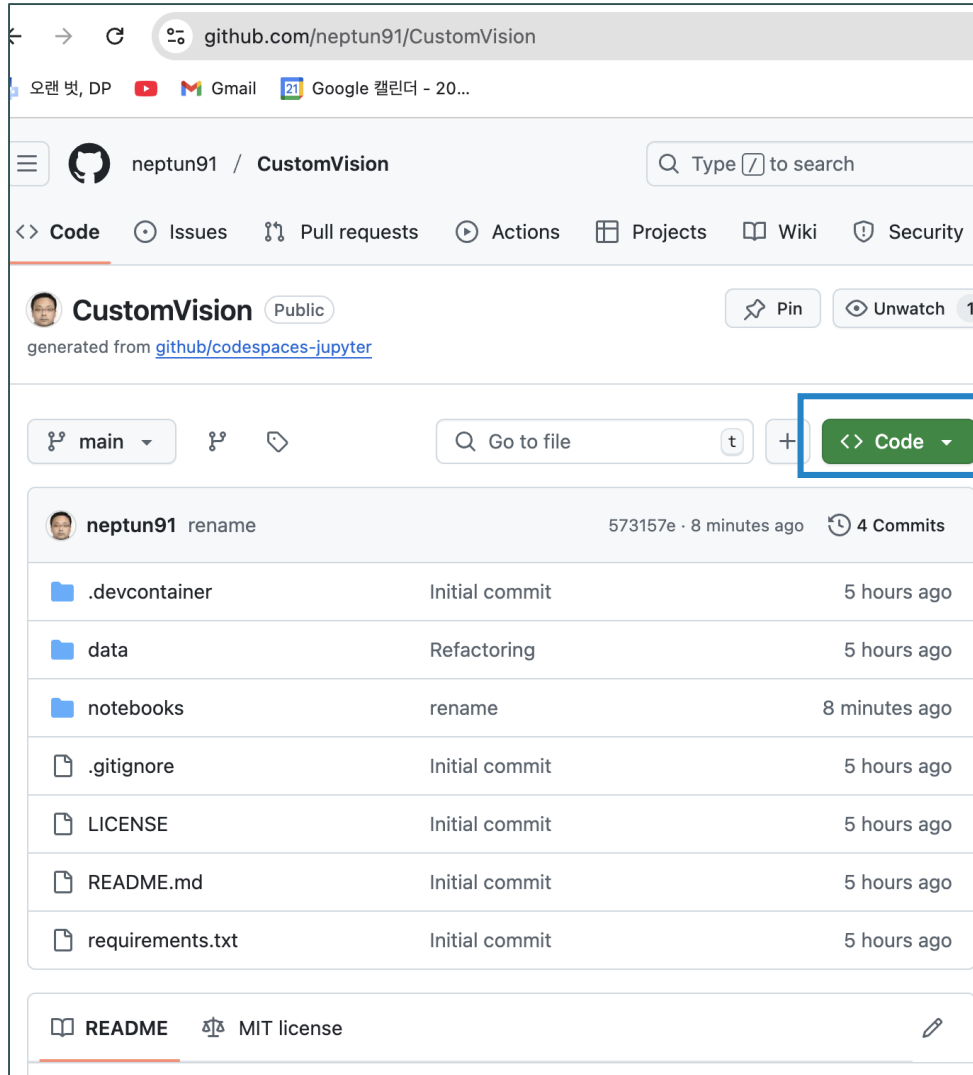


Step2 주소창입력 : <https://github.com/neptun92/CustomVision>



GitHub codespace 생성(2)

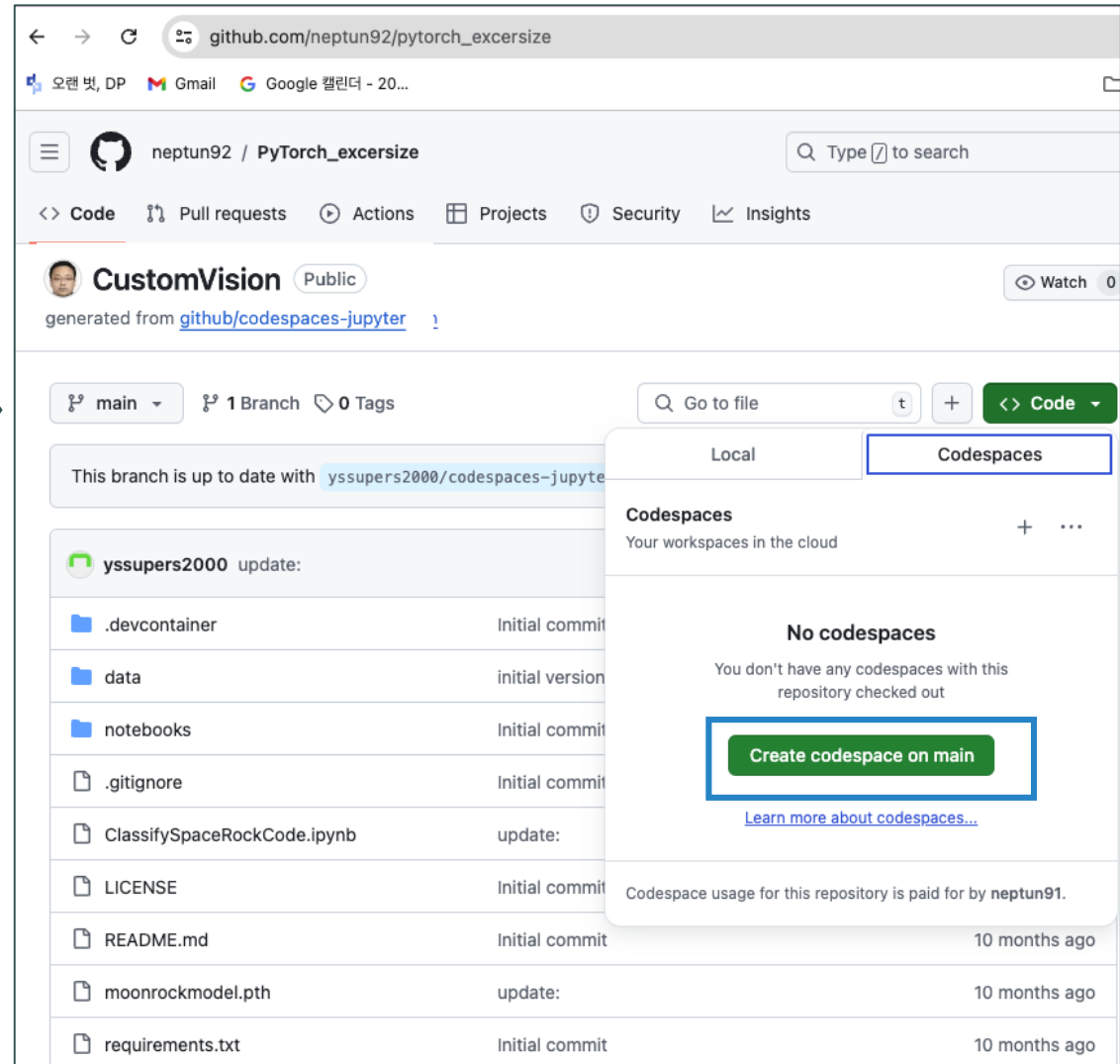
Step3. CustomVision 소스 화면 → "<> Code" 버튼 클릭



The screenshot shows the GitHub repository page for 'neptun91 / CustomVision'. The repository is public and generated from 'github/codespaces-jupyter'. The file list shows several files and folders, including '.devcontainer', 'data', 'notebooks', '.gitignore', 'LICENSE', 'README.md', and 'requirements.txt'. The '<> Code' button is highlighted with a blue box.



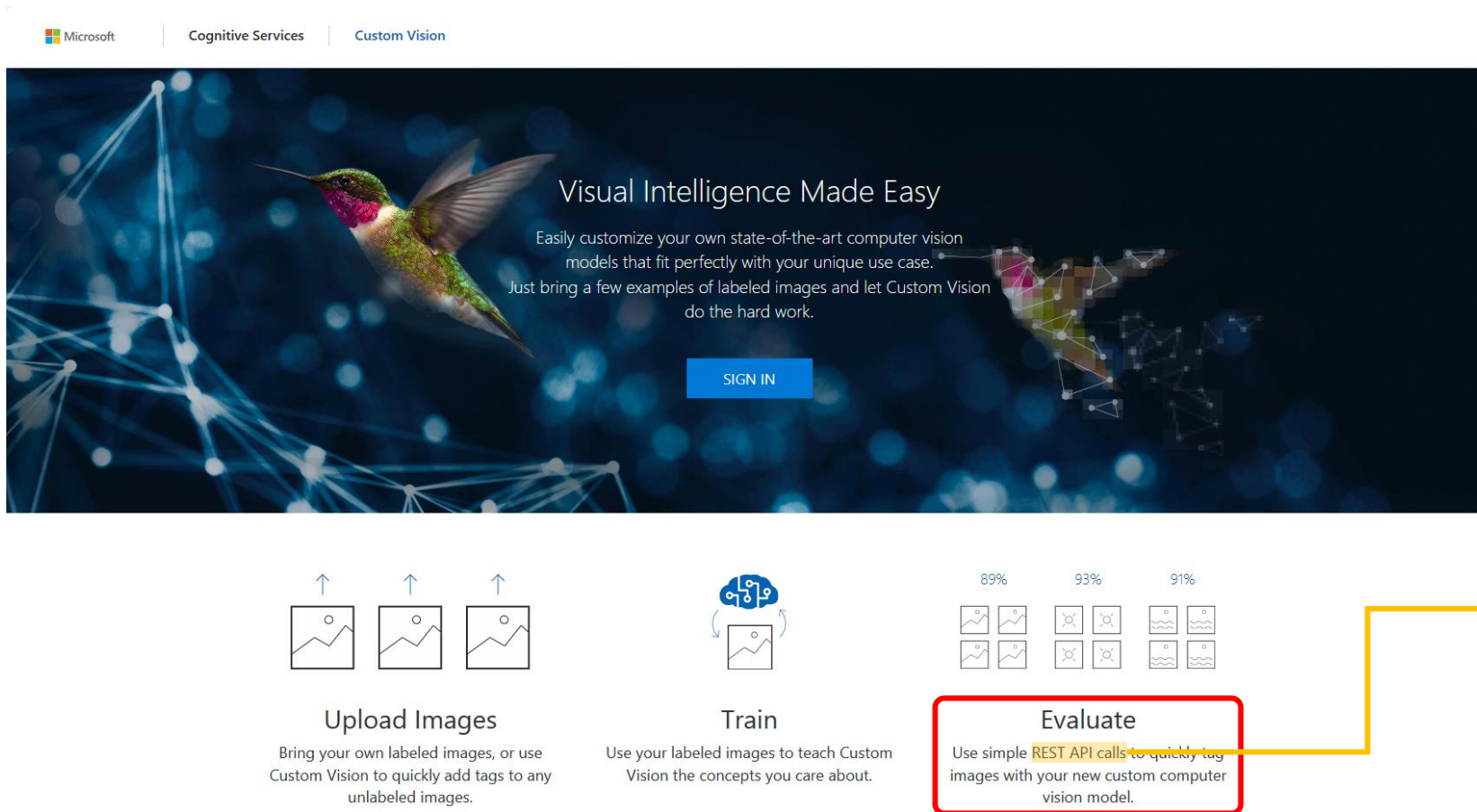
Step4 'Create codespace on main' 클릭



The screenshot shows the GitHub repository page for 'neptun92 / PyTorch_excercise'. The repository is public and generated from 'github/codespaces-jupyter'. The file list shows several files and folders, including '.devcontainer', 'data', 'notebooks', '.gitignore', 'ClassifySpaceRockCode.ipynb', 'LICENSE', 'README.md', 'moonrockmodel.pth', and 'requirements.txt'. The 'Create codespace on main' button is highlighted with a blue box.

Custom Vision을 외부에서 활용하기

Azure의 Custom Vision은 URL(<https://www.customvision.ai>)에서 알 수 있듯이 Azure Portal(<https://portal.azure.com>)과는 독립된 별도의 사이트에서 제공되는 서비스이며 사용자가 만든 컴퓨터 비전 모델을 바로 확인할 수 있는 UI를 제공합니다. 그런데, 만약 다른 프로그램에서도 활용하고 싶다면?



The screenshot shows the Azure Custom Vision website interface. At the top, there's a navigation bar with 'Microsoft', 'Cognitive Services', and 'Custom Vision'. Below this is a large hero section with a hummingbird and the text 'Visual Intelligence Made Easy'. A 'SIGN IN' button is visible. Below the hero section, there are three main steps in the workflow:

- Upload Images**: Bring your own labeled images, or use Custom Vision to quickly add tags to any unlabeled images. (Icon: three image uploads)
- Train**: Use your labeled images to teach Custom Vision the concepts you care about. (Icon: a cloud with a plus sign and a circular arrow)
- Evaluate**: Use simple **REST API calls** to quickly tag images with your new custom computer vision model. (Icon: three sets of image classification results with accuracy percentages: 89%, 93%, and 91%). This step is highlighted with a red box.

An orange arrow points from the 'Evaluate' step to the text on the right.

REST API 호출을 통해서
사용자가 만든 커스텀 비
전 모델을 외부 프로그램
에서 활용할 수 있음

Custom Vision Resource 복습

앞에서 배운 Custom Vision의 Resource 생성 과정을 다시 살펴보겠습니다. 앞에서는 Custom Vision 포털로 이동하여 서비스를 이용했지만, 외부 애플리케이션과 연동을 위해서는 API 서비스를 활용해야 합니다.



Custom Vision을 시작 및 실행하기 위한 빠른 시작 지침을 살펴봅니다.

1

API 키를 가져와 애플리케이션을 인증하고 서비스에 대한 호출을 보내기 시작합니다.
모든 Custom Vision 호출에는 키가 필요합니다. 키는 왼쪽 창의 키 및 엔드포인트 섹션에서 찾을 수 있습니다. 요청 헤더(웹 API) 또는 Custom Vision 클라이언트(SDK)에서 키를 지정하세요.
[API 키](#)

2

Custom Vision 포털에서 서비스 이해해 보기 - API 키 필요
코드를 작성하지 않고 API를 빠르게 사용해 보려면 Custom Vision 포털을 사용합니다. 애플리케이션을 빌드할 준비가 되었으면 Custom Vision SDK를 사용하여 이 모든 기능을 프로그래밍 방식으로 구현할 수 있습니다.
[Custom Vision 포털](#)

3

웹 API 호출 - API 키 및 위치가 필요합니다.
해당 빠른 시작의 샘플 코드를 사용하여 Custom Vision 서비스를 애플리케이션에 통합하여 이미지에서 관심 있는 범주를 인식합니다. API 키와 위치는 왼쪽 탭의 키 및 엔드포인트 섹션에서 찾을 수 있습니다.
[C# 빠른 시작](#)
[Python 빠른 시작](#)
[NodeJS 빠른 시작](#)
[Java 빠른 시작](#)
[빠른 시작으로 이동](#)

4

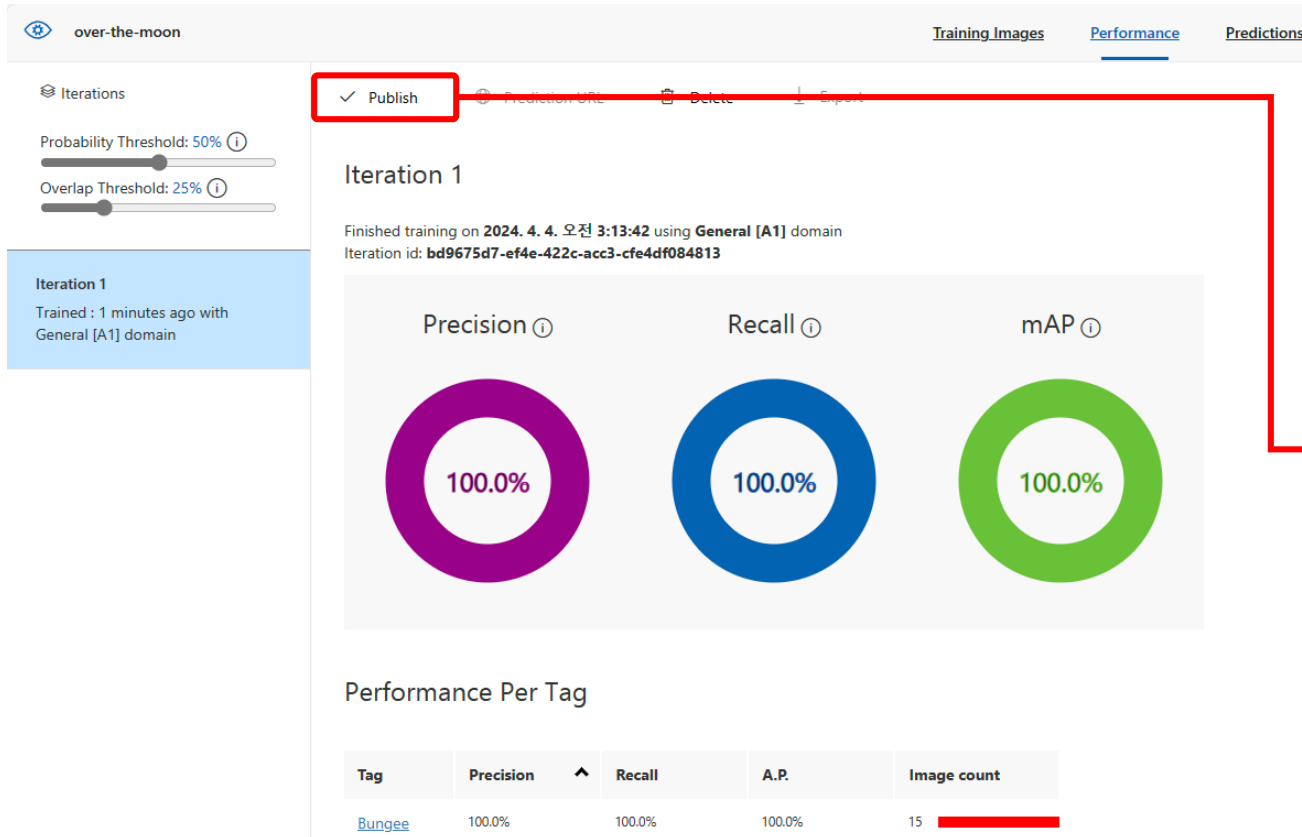
애플리케이션 빌드해 보기
지원되는 언어 목록을 가져옵니다. API의 각 속성 및 메서드에 대한 자세한 정보를 가져옵니다. API를 실행하고 나면 API '개요'에서 Azure Portal의 API 상태와 사용량을 확인할 수 있습니다. Flow를 통해 코드
[설명서](#)
[Custom Vision 및 IoT 카메라로 시각적 경고 생성](#)
[로컬 디바이스에서 실행할 모델 내보내기](#)

- 앞에서 배운 방법으로 Custom Vision Portal에서 GUI를 활용하여 서비스를 이용
- 웹 API를 활용하여 외부 애플리케이션과 연결. C#, Python, NodeJS, Java 등 다양한 언어 지원
- Microsoft Learn에서 제공하는 예제를 참고

개체 감지

개체 감지 서비스와 연동하기

앞에서 배운 '번지 찾기' 실습에서 이어서 진행해보도록 하겠습니다. 우선, 앞에서 만든 모델을 publish하여 외부에서 사용할 수 있도록 합니다.



Publish 버튼을 누르고 나오는 UI에서 Model name 과 Prediction resource를 지정합니다. 여기서 지정한 값이 뒤에서 API 호출하는 부분에서 사용됩니다.

The 'Publish Model' dialog box contains the following text and fields:

We only support publishing to a prediction resource in the same region as the training resource the project resides in.

Please check if you have a prediction resource and if the prediction resource is in the same region as the training resource.

Model name
Iteration1

Prediction resource
customtest-Prediction

Buttons: Publish, Cancel

외부 연동에서 필요한 정보들 확인하기 1/2

Publish가 완료되면 아래와 같이 'PUBLISHED'라는 표시가 나오고, Prediction URL 버튼이 활성화 됩니다. 이 때 Prediction URL을 눌러서 나오는 값들이 뒤에서 사용됩니다.

The screenshot shows the 'over-the-moon' web interface. On the left, under 'Iterations', 'Iteration 1' is listed with a 'PUBLISHED' status. The main area shows 'Iteration 1' details: 'Finished training on 2024. 4. 4. 오전 3:13:42 using General [A1] domain', 'Iteration id: bd9675d7-ef4e-422c-acc3-cfe4df084813', and 'Published as: Iteration1'. Performance metrics are shown as donut charts: Precision at 100.0%, Recall at approximately 80%, and mAP at approximately 70%. A 'Prediction URL' button is highlighted with a red box. A red arrow points from this button to a modal window titled 'How to use the Prediction API'. The modal provides instructions for using the API with an image URL or file, with the Prediction-Key and Content-Type headers highlighted in red boxes.

over-the-moon

Training Images Performance Predictions

Train Quick Test

Iterations

Probability Threshold: 50%
Overlap Threshold: 25%

Iteration 1 **PUBLISHED**
Trained : 5 hours ago with General [A1] domain

Unpublish Prediction URL Delete Export

Iteration 1

Finished training on 2024. 4. 4. 오전 3:13:42 using General [A1] domain
Iteration id: bd9675d7-ef4e-422c-acc3-cfe4df084813
Published as: Iteration1

Precision 100.0% Recall mAP

Performance Per Tag

Tag	Precision
Bungee	100.0%

How to use the Prediction API

If you have an image URL:

`https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic`

Set Prediction-Key Header to : 700799768f3a4f708adb9591616e4932
Set Content-Type Header to : application/json
Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

`https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic`

Set Prediction-Key Header to : 700799768f3a4f708adb9591616e4932
Set Content-Type Header to : application/octet-stream
Set Body to : <image file>

Got it!

외부 연동에서 필요한 정보들 확인하기 2/2

화면에 나오는 첫번째 URL을 스크롤해서 파란색으로 선택된 부분을 복사해놓습니다. 이 값은 프로젝트 ID 입니다.

How to use the Prediction API

If you have an image URL:

```
https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic
```

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

```
https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic
```

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/octet-stream

Set Body to : <image file>

Got it!

How to use the Prediction API

If you have an image URL:

```
ion/v3.0/Prediction/4831a078-352c-47d5-86a0-c8654dabc2dc/detect/iterations/ite
```

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

```
https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic
```

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/octet-stream

Set Body to : <image file>

Got it!

Training Images Performance Predictions **Train** Quick Test ?



over-the-moon

Project Settings

General

Project Name*

over-the-moon

Project Id

4831a078-352c-47d5-86a0-c8654dabc2dc

Description

GitHub Codespaces 설정하기

Azure Custom Vision API를 사용하기 위해서는 Python 환경에 azure-cognitiveservices-vision-customvision 라이브러리를 추가해야 하며, 아래와 같이 2가지 방법이 있습니다.

1 Codespace 생성시에 추가하는 방법

- Codespace Jupyter Notebook template에 있는 requirements.txt에 필요한 python library를 등록합니다.
- 이를 위해서는 GitHub에서 제공하는 기본 repository에서 branch 해서 requirements.txt를 수정한 후 commit해야 합니다.
- 기본 template에 있는 내용을 바꿔야 하므로 Git에 대한 이해가 필요하지만, 일단 적용해 놓으면 계속 사용 가능합니다.

requirements.txt

```
ipywidgets==7.7.1
matplotlib==3.7.0
numpy==1.24.2
pandas==1.5.3
torch==1.12.1
torchvision==0.13.1
tqdm==4.64.0
azure-cognitiveservices-vision-customvision==3.1.0
```

2 Codespace 생성 후에 터미널에서 추가하는 방법

- Workspace 하단에 보이는 터미널 창에서 pip 명령을 통해 추가로 설치합니다.



The screenshot shows a JupyterLab interface with a terminal window open. The terminal title bar includes tabs for '문제', '출력', '디버그 콘솔', '터미널', '포트', and '2 JUPYTER'. The terminal content shows the user '@yssupers2000' in the directory '/workspaces/codespaces-jupyter' on the 'main' branch, typing the command 'pip install azure-cognitiveservices-vision-customvision'. The command is highlighted with a red rectangular box.

Azure Custom Vision 라이브러리 등록

우선, Azure Custom Vision 라이브러리를 사용할 수 있도록 import

```
# Azure의 Custom Vision 라이브러리를 추가. 예측을 위하여 prediction을 포함
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient

# OpenAPI 스펙에 맞춰서 Authentication을 처리할 수 있도록 해주는 코드
from msrest.authentication import ApiKeyCredentials

# Matplotlib의 pyplot을 포함하여 예측 결과를 그리기
from matplotlib import pyplot as plt

# Python Image 라이브러리로 이미지 그리기
from PIL import Image, ImageDraw, ImageFont

# Python Numpy (수학 및 과학 연산 패키지) 포함
import numpy as np

# 파일 처리 작업을 위해 os 라이브러리 포함
import os
```

CustomVisionPredictionClient Class

Reference

[Feedback](#)

In this article

- [Constructor](#) ▾
- [Variables](#)
- [Methods](#) ▾

Inheritance `azure.cognitiveservices.vision.customvision.prediction.operations._custom_vision_prediction_client_operations.CustomVisionPredictionClientOperationsMixin` → `CustomVisionPredictionClient`
`msrest.service_client.SDKClient` → `CustomVisionPredictionClient`

Constructor

```
Python Copy  
CustomVisionPredictionClient(endpoint, credentials)
```

Parameters

[Expand table](#)

Name	Description
<div>✓ <code>endpoint</code> Required*</div>	<code>str</code> Supported Cognitive Services endpoints.
<div>✓ <code>credentials</code> Required*</div>	<code>None</code> Subscription credentials which uniquely identify client subscription.

참고

Variables

[Expand table](#)

Name	Description
<code>config</code>	<code>CustomVisionPredictionClientConfiguration</code> Configuration for client.

Methods

[Expand table](#)

✓ <code>classify_image</code>	Classify an image and saves the result.
<code>classify_image_url</code>	Classify an image url and saves the result.
<code>classify_image_url_with_no_store</code>	Classify an image url without saving the result.
<code>classify_image_with_no_store</code>	Classify an image without saving the result.
<code>close</code>	Close the client if keep_alive is True.
✓ <code><u>detect_image</u></code>	Detect objects in an image and saves the result.
<code>detect_image_url</code>	Detect objects in an image url and saves the result.
<code>detect_image_url_with_no_store</code>	Detect objects in an image url without saving the result.
<code>detect_image_with_no_store</code>	Detect objects in an image without saving the result.

Custom Vision Web에 있는 값을 설정하여 클라이언트 인증

사용자가 만든 AI 모델의 endpoint 및 key 값 등을 설정합니다. Custom Vision Web에서 확인했던 값들을 설정하는 코드입니다.

사용자가 만든 AI 모델의 예측 기능을 사용하기 위한 endpoint 지정

`prediction_endpoint = https://customtest-prediction.cognitiveservices.azure.com`

KEY 값 지정

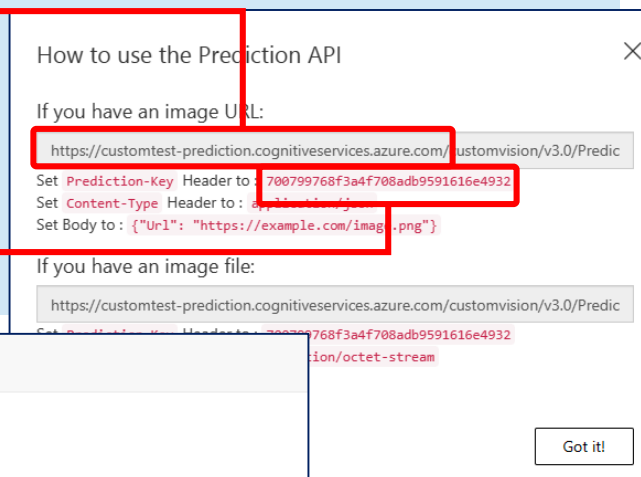
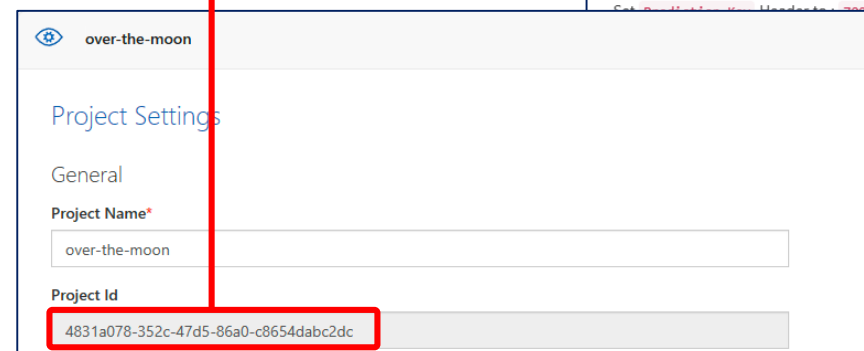
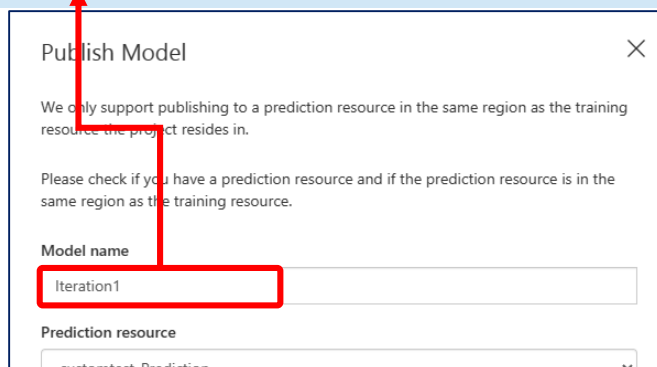
`prediction_key = "700799768f3a4f708adb9591616e4932"`

프로젝트 ID 지정

`project_id = "4831a078-352c-47d5-86a0-c8654dabc2dc"`

모델명 지정

`model_name = "Iteration1"`



앞에서 지정한 API KEY를 써서 커스텀 비전 모델을 사용할 클라이언트를 인증

`credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})`

endpoint를 써서 클라이언트 등록

`predictor = CustomVisionPredictionClient(endpoint=prediction_endpoint, credentials=credentials)`

AI 모델을 사용해서 예측에 사용할 이미지 지정

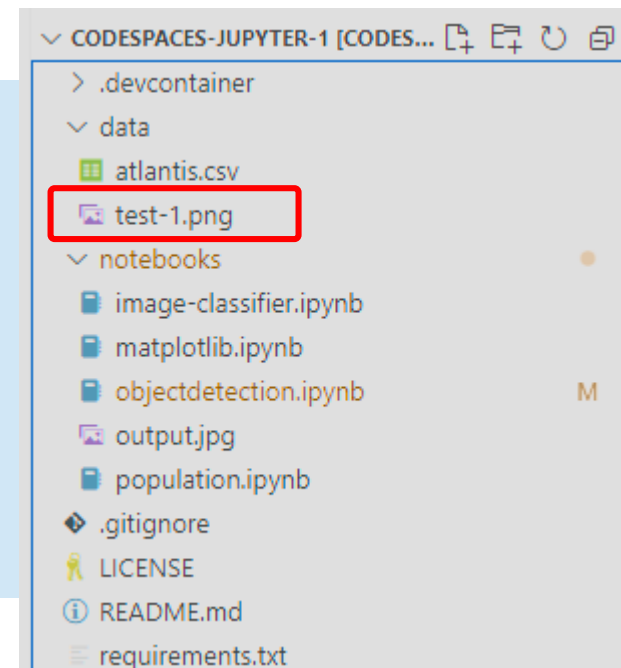
테스트 이미지를 읽기

```
# 테스트 이미지를 Codespace workspace에 추가한 후 image_file 변수로 지정
image_file = "../data/test-1.png"

# 이미지 파일 등록되었음을 출력
Print(' Detecting objects in ', image_file)

# Python Imaging Library의 image open함수를 써서 테스트 이미지 파일 오픈
image = Image.open(image_file)

# Numpy에서 이미지의 shape을 높이, 폭, 채널 읽기
h, w, ch = np.array(image).shape
```



테스트 이미지를 적용하고 결과를 텍스트로 출력

인공지능 모델에 테스트 이미지 적용하고 결과를 표시합니다. 우선 text로 결과를 표시해봅니다.

```
# 테스트 이미지를 열고 모델에 적용해서 결과를 저장
with open(image_file, mode="rb") as image_data:
    results = predictor.detect_image(project_id, model_name, image_data)

# 예측한 결과를 모두 출력 (텍스트로 표시됨)
for prediction in results.predictions:
    print( "\t " + prediction.tag_name + " : {0:.2f}% bbox.left = {1:.2f}, bbox.top = {2:.2f}, bbox.width = {3:.2f}, bbox.height = {4:.2f} ".format(prediction.probability * 100, prediction.bounding_box.left, prediction.bounding_box.top, prediction.bounding_box.width, prediction.bounding_box.height))

# 그래프 크기 지정하고 축 비활성화
fig = plt.figure(figsize=(8,8))
plt.axis('off')

# 테스트 이미지를 그리기
# 개체 인식 박스를 magenta로 지정
draw = ImageDraw.Draw(image)
lineWidth = int(w/100)
color = 'magenta'
```

결과

Bungee: 99.28% bbox.left = 0.00, bbox.top = 0.35, bbox.width = 0.75, bbox.height = 0.58
Bungee: 40.26% bbox.left = 0.01, bbox.top = 0.06, bbox.width = 0.11, bbox.height = 0.43
Bungee: 35.52% bbox.left = 0.60, bbox.top = 0.07, bbox.width = 0.40, bbox.height = 0.93
Bungee: 8.67% bbox.left = 0.01, bbox.top = 0.19, bbox.width = 0.67, bbox.height = 0.81
Bungee: 5.49% bbox.left = 0.00, bbox.top = 0.02, bbox.width = 0.16, bbox.height = 0.63
Bungee: 1.84% bbox.left = 0.00, bbox.top = 0.39, bbox.width = 0.38, bbox.height = 0.61
Bungee: 1.39% bbox.left = 0.45, bbox.top = 0.47, bbox.width = 0.55, bbox.height = 0.53

인식한 개체 영역을 bounding box로 표시

인식 결과를 bounding box로 이미지에 표시하고 파일로 저장하기

```
# 개체 인식된 모든 결과에 대해서
for prediction in results.predictions:

    # 확률이 50%이 이상인 경우 bounding box 값을 읽음
    if (prediction.probability*100) > 50:
        left = prediction.bounding_box.left * w
        top = prediction.bounding_box.top * h
        width = prediction.bounding_box.width * w
        height = prediction.bounding_box.height * h

        # bounding box 값을 magenta색으로 표시
        points = ((left,top), (left+width,top), (left+width,top+height), (left,top+height),(left,top))
        draw.line(points, fill=color, width=lineWidth)
        plt.annotate(prediction.tag_name + ' {0:.2f}%'.format(prediction.probability * 100), (left, top),
        color=color)

# bounding box 표시된 이미지를 output.jpg로 저장
plt.imshow(image)
outputfile = 'output.jpg'
fig.savefig(outputfile)
print('Results saved in', outputfile)
```



이미지 분류

이미지 분류 서비스와 연동하기

앞에서 배운 '암석 식별 머신' 실습에서 이어서 진행해보도록 하겠습니다. 우선, 앞에서 만든 모델을 publish하여 외부에서 사용할 수 있도록 합니다.

The screenshot displays the Elixirr AI platform interface for a project named 'find-basalt-highland-prj'. The 'Performance' tab is active, showing training results for 'Iteration 1'. The model was trained on 2024. 4. 4. 오후 11:24:14 using the General [A2] domain. The iteration ID is 88336083-a98f-4533-8789-bf9e003c2a9a, and the classification type is Multiclass (Single tag per image). The performance metrics are displayed as donut charts: Precision at 75.0%, Recall at 75.0%, and AP at 95.0%. Below these charts, a table titled 'Performance Per Tag' shows the performance for 'highland' and 'basalt' tags. The 'highland' tag has a precision of 100.0%, recall of 50.0%, and A.P. of 100.0% with 8 images. The 'basalt' tag has a precision of 66.7%, recall of 100.0%, and A.P. of 100.0% with 10 images. A red box highlights the 'Publish' button in the top navigation bar. An arrow points from this button to a 'Publish Model' dialog box. The dialog box contains instructions about publishing to a prediction resource in the same region as the training resource. It includes a 'Model name' field with the value 'Iteration1' and a 'Prediction resource' dropdown menu with the value 'customtest-Prediction'. The dialog box has 'Publish' and 'Cancel' buttons at the bottom.

find-basalt-highland-prj

Training Images Performance Predictions Train Quick Test

Iterations

Probability Threshold: 50%

Iteration 1

Trained : moments ago with General [A2] domain

Iteration 1

Finished training on 2024. 4. 4. 오후 11:24:14 using General [A2] domain
Iteration id: 88336083-a98f-4533-8789-bf9e003c2a9a
Classification type: Multiclass (Single tag per image)

Precision 75.0% Recall 75.0% AP 95.0%

Performance Per Tag

Tag	Precision	Recall	A.P.	Image count
highland	100.0%	50.0%	100.0%	8
basalt	66.7%	100.0%	100.0%	10

Publish Model

We only support publishing to a prediction resource in the same region as the training resource the project resides in.

Please check if you have a prediction resource and if the prediction resource is in the same region as the training resource.

Model name

Iteration1

Prediction resource

customtest-Prediction

Publish Cancel

외부 연동에서 필요한 정보들 확인하기 1/2

Publish가 완료되면 아래와 같이 'PUBLISHED'라는 표시가 나오고, Prediction URL 버튼이 활성화 됩니다. 이 때 Prediction URL을 눌러서 나오는 값들이 뒤에서 사용됩니다.

The screenshot shows the Azure Custom Vision interface for a project named 'find-basalt-highland-prj'. The 'Performance' tab is selected. On the left, 'Iteration 1' is listed with a 'PUBLISHED' status. The main area shows 'Iteration 1' details: 'Finished training on 2024. 4. 4. 오후 11:24:14 using General [A2] domain', 'Iteration id: 88336083-a98f-4533-8789-bf9e003c2a9a', 'Classification type: Multiclass (single tag per image)', and 'Published as: Iteration1'. A red box highlights the 'Prediction URL' button. A red arrow points from this button to a modal window titled 'How to use the Prediction API'. The modal provides instructions for using the API with an image URL or file, including the URL, headers, and body. A red box highlights the 'Prediction-Key' header value in the modal.

Iteration 1

Trained : 5 minutes ago with General [A2] domain

Published : 5 minutes ago with General [A2] domain

Finished training on 2024. 4. 4. 오후 11:24:14 using General [A2] domain
Iteration id: 88336083-a98f-4533-8789-bf9e003c2a9a
Classification type: Multiclass (single tag per image)
Published as: Iteration1

Precision 75.0%

Recall

AP

Performance Per Tag

Tag	Precision
highland	100.0%
basalt	66.7%

How to use the Prediction API

If you have an image URL:

https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic

Set Prediction-Key Header to : 700799768f3a4f708adb9591616e4932

Set Content-Type Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic

Set Prediction-Key Header to : 700799768f3a4f708adb9591616e4932

Set Content-Type Header to : application/octet-stream

Set Body to : <image file>

Got it!

외부 연동에서 필요한 정보들 확인하기 2/2

화면에 나오는 첫번째 URL을 스크롤해서 파란색으로 선택된 부분을 복사해놓습니다. 이 값은 프로젝트 ID 입니다.

How to use the Prediction API

×

If you have an image URL:

`https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/2a6e1202-f478-46da-bd97-a0e00651d015/classify/iteration:1`

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

`https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/2a6e1202-f478-46da-bd97-a0e00651d015/classify/iteration:1`

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/octet-stream

Set Body to : <image file>

Got it!

How to use the Prediction API

×

If you have an image URL:

`nvision/v3.0/Prediction/2a6e1202-f478-46da-bd97-a0e00651d015/classify/iteration:1`

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}

If you have an image file:

`https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/2a6e1202-f478-46da-bd97-a0e00651d015/classify/iteration:1`

Set **Prediction-Key** Header to : 700799768f3a4f708adb9591616e4932

Set **Content-Type** Header to : application/octet-stream

Set Body to : <image file>

Got it!


Training Images

Performance

Predictions

Train

Quick Test



?

find-basalt-highland-prj

Project Settings

General

Project Name*

find-basalt-highland-prj

Project Id

2a6e1202-f478-46da-bd97-a0e00651d015

Description

Azure Custom Vision 라이브러리 등록

GitHub Codespace 설정 과정은 개체감지와 동일합니다. 이제 Python 코딩 부분을 살펴보겠습니다. 우선, Azure Custom Vision 라이브러리를 사용할 수 있도록 import합니다.

```
# Azure의 Custom Vision 라이브러리를 추가. 예측을 위하여 prediction을 포함
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient

# OpenAPI 스펙에 맞춰서 Authentication을 처리할 수 있도록 해주는 코드
from msrest.authentication import ApiKeyCredentials
```

Custom Vision Web에 있는 값을 설정하여 클라이언트 인증

사용자가 만든 AI 모델의 endpoint 및 key 값 등을 설정합니다. Custom Vision Web에서 확인했던 값들을 설정하는 코드입니다.

사용자가 만든 AI 모델의 예측 기능을 사용하기 위한 endpoint 지정

prediction_endpoint = <https://customtest-prediction.cognitiveservices.azure.com>

KEY 값 지정

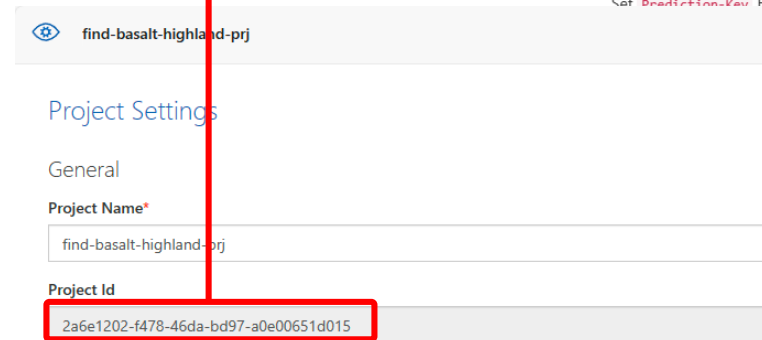
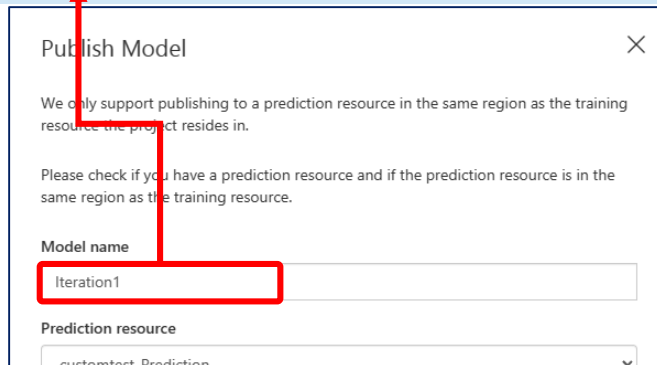
prediction_key = "700799768f3a4f708adb9591616e4932"

프로젝트 ID 지정

project_id = "2a6e1202-f478-46da-bd97-a0e00651d015"

모델명 지정

model_name = "Iteration1"



How to use the Prediction API

If you have an image URL:

<https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic>

Set Prediction-Key Header to: 700799768f3a4f708adb9591616e4932

Set Content-Type Header to: application/json

Set Body to: {"Url": "https://example.com/image.png"}

If you have an image file:

<https://customtest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predic>

Set Prediction-Key Header to: 700799768f3a4f708adb9591616e4932

/v1/octet-stream

Got it!

앞에서 지정한 API KEY를 써서 커스텀 비전 모델을 사용할 클라이언트를 인증

credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})

endpoint를 써서 클라이언트 등록

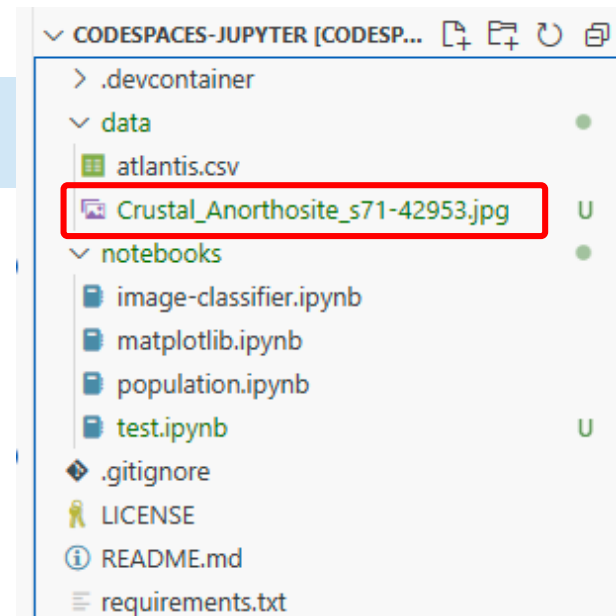
predictor = CustomVisionPredictionClient(endpoint=prediction_endpoint, credentials=credentials)

참고: 앞에서 진행한 개체감지와 Endpoint와 key는 동일하고 project ID만 다른 것을 알 수 있습니다.

AI 모델을 사용해서 예측에 사용할 이미지 지정

테스트 이미지를 읽기

```
# 테스트 이미지를 Codespace workspace에 추가한 후 image_file 변수로 지정  
image_file = "../data/Crustal_Anorthosite_s71-42953.jpg"
```



테스트 이미지를 적용하고 결과를 텍스트로 출력

인공지능 모델에 테스트 이미지 적용하고 결과를 표시합니다.

```
# 테스트 이미지를 열고 모델에 적용해서 결과를 저장
with open(image_file, mode="rb") as image_data:
    results = predictor.classify_image(project_id, model_name, image_data)

# 예측한 결과를 출력
for prediction in results.predictions:
    print(f"Tag: {prediction.tag_name}, Probability: {prediction.probability:.2f}")
```



Crustal_Anorthosite_s71-42953.jpg - 고지대암석 (Highland)

결과

Tag: highland, Probability: 0.98 Tag: basalt, Probability: 0.02

학습하는 것도 Python coding으로 가능한가요?

이상으로 학습한 모델을 Python 코드에서 사용하는 방법을 알아보았습니다. 또한, Training 부터 Custom Vision Web을 쓰지 않고 Python 코드 혹은 C#, Go lang, Java 등 다양한 언어를 써서 외부 애플리케이션에서 서비스를 활용할 수 있습니다.

Q: Custom Vision Web UI 말고 직접 코딩을 해야 하는 경우가 있나요?

A: 우리가 직접 이미지 데이터를 수집해서 바운딩 박스를 그리는 작업을 하는 경우가 아니고 이미 만들어진 이미지 데이터를 사용 하는 경우에 해당 데이터 셋에는 이미지 데이터와 바운딩 박스 정보가 text 정보로 존재합니다. 이때는 코딩을 써서 처리하는 것이 더 간단한 방법입니다.

<https://learn.microsoft.com/ko-kr/azure/ai-services/custom-vision-service/quickstarts/object-detection?tabs=windows%2Cvisual-studio&pivots=programming-language-python>

