Image classification with the Azure Custom Vision

Command to install required libraries:

pip install azure-cognitiveservices-vision-customvision

In [2]:
```
from azure.cognitiveservices.vision.customvision.training import CustomVisionTrai
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPr
from azure.cognitiveservices.vision.customvision.training.models import ImageFile
from msrest.authentication import ApiKeyCredentials
import os, time, uuid
```

In [3]:
```
# Replace with valid values
ENDPOINT = "https://sdai.cognitiveservices.azure.com/"
PredictionENDPOINT = "https://sdai-prediction.cognitiveservices.azure.com/"
training_key = "56e05f26d6d84d42a99c877e392c6170"
prediction_key = "e9eb4c1a158c4bfc8da8397bb161ef2c"
prediction_resource_id = "/subscriptions/5cf92398-d8a0-400a-9d32-05fcb62b046b/res
```

Authenticate the client

Instantiate a training and prediction client with your endpoint and keys. Create ApiKeyServiceClientCredentials objects with your keys, and use them with your endpoint to create a CustomVisionTrainingClient and CustomVisionPredictionClient object.

In [4]:
```
credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": predicti
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentia
```

Create a new Custom Vision project

Add the following code to your script to create a new Custom Vision service project.

See the create_project method to specify other options when you create your project (explained in the Build a classifier web portal guide).

In [5]:
```
publish_iteration_name = "classifyModel"

credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)

# Create a new project
print ("Creating project...")
project_name = uuid.uuid4()
project = trainer.create_project(project_name)
print('d')
```

```
Creating project...
d
```

In [6]:
```
# Make two tags in the new project
mango_tag = trainer.create_tag(project.id, "Mango")
apple_tag = trainer.create_tag(project.id, "Apple")
```

Upload and tag images

In [7]:
```
#__file__ : Put location where your "images" folder is located in your system

base_image_location = os.path.join (os.path.dirname("C:/Users/badda/Downloads/Dat

print("Adding images...")

image_list = []

for image_num in range(1, 11):
    file_name = "mango_{}.jpg".format(image_num)
    with open(os.path.join (base_image_location, "Mango", file_name), "rb") as in
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_cor

for image_num in range(1, 11):
    file_name = "apple_{}.jpg".format(image_num)
    with open(os.path.join (base_image_location, "Apple", file_name), "rb") as in
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_cor

upload_result = trainer.create_images_from_files(project.id, ImageFileCreateBatch
if not upload_result.is_batch_successful:
    print("Image batch upload failed.")
    for image in upload_result.images:
        print("Image status: ", image.status)
    exit(-1)
```

```
Adding images...
```

In [8]:
```
print ("Training...")
iteration = trainer.train_project(project.id)
while (iteration.status != "Completed"):
    iteration = trainer.get_iteration(project.id, iteration.id)
    print ("Training status: " + iteration.status)
    print ("Waiting 10 seconds...")
    time.sleep(10)
```

```
Training...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Training
```

```
Waiting 10 seconds...
Training status: Training
Waiting 10 seconds...
Training status: Completed
Waiting 10 seconds...
```

Publish the current iteration

In [9]:
```
# The iteration is now trained. Publish it to the project endpoint
trainer.publish_iteration(project.id, iteration.id, publish_iteration_name, predi
print ("Done!")
```

```
Done!
```

Test the prediction endpoint

In [10]:
```
# Now there is a trained endpoint that can be used to make a prediction
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": predicti
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentia

with open(os.path.join (base_image_location, "Test/test_image.jpg"), "rb") as ima
    results = predictor.classify_image(
        project.id, publish_iteration_name, image_contents.read())

    # Display the results.
    for prediction in results.predictions:
        print("\t" + prediction.tag_name +
              ": {0:.2f}%".format(prediction.probability * 100))
```

```
        Mango: 100.00%
        Apple: 0.00%
```

In [11]:
```
# Now there is a trained endpoint that can be used to make a prediction
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": predicti
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentia

with open(os.path.join (base_image_location, "Test/test.jpg"), "rb") as image_cor
    results = predictor.classify_image(
        project.id, publish_iteration_name, image_contents.read())

    # Display the results.
    for prediction in results.predictions:
        print("\t" + prediction.tag_name +
              ": {0:.2f}%".format(prediction.probability * 100))
```

```
        Apple: 100.00%
        Mango: 0.00%
```

In [ ]:
```
```

Task 4: Make a small code toolkit where you upload the image in runtime and it performs classification.You have to use same ipynb file to perform the task. (40%)

In [12]:
```
# below is a way to dynamically upload the image by giving image directory and im
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": predicti
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentia

import os
base_image_location1 = input("Enter test image directory path ")
base_image_location1 = str(base_image_location1).replace(os.path.sep, '/') + '/'
image = input("Enter jpg image filename without extention ") + '.jpg'

with open(os.path.join (base_image_location1, image), "rb") as image_contents:
    results = predictor.classify_image(
        project.id, publish_iteration_name, image_contents.read())

    # Display the results.
    for prediction in results.predictions:
        print("\t" + prediction.tag_name +
              ": {0:.2f}%".format(prediction.probability * 100))
```

```
Enter test image directory path C:\Users\badda\Downloads\Data1\Images\Test
Enter jpg image filename without extention test
        Apple: 100.00%
        Mango: 0.00%
```

In [13]:
```
#Below is one more way to uplaod images at runtime to perform classification

import os
from IPython.display import Image
import ipywidgets as widgets
from ipywidgets import interact, interact_manual

#download any image to the below directory
filedir = "C:/Users/badda/Downloads/Data1/Images/Test/"

#below function is used to select the image at runtime
@interact
def display_images(filename=os.listdir(filedir)):
    display(Image(filedir+filename))
    with open(os.path.join (filedir, filename), "rb") as image_contents:
        results = predictor.classify_image(
        project.id, publish_iteration_name, image_contents.read())

    # Display the results.
    for prediction in results.predictions:
        print("\t" + prediction.tag_name +
              ": {0:.2f}%".format(prediction.probability * 100))
```

filename apple.jpg

```
Apple: 100.00%
Mango: 0.00%
```

In [ ]:

Task 1: Execute the code properly with given sample data and solve any issues that may arise in the code.(30%)

Task 2: Explain what you analyzed in the code. Make a detailed report. (10%)

Task 3: Use any other image dataset to run the tasks above again.(20%)