Custom Vision SDK

Image classification

In [6]:
```
from azure.cognitiveservices.vision.customvision.training import CustomVisionTrainingClient
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from azure.cognitiveservices.vision.customvision.training.models import ImageFileCreateBatch, ImageFileCreateEnt
from msrest.authentication import ApiKeyCredentials
import time
```

In [ ]:

In [7]:
```

ENDPOINT = "https://easgfdkjfk.cognitiveservices.azure.com/"
training_key = "f4a1f9a28d9442609c8931096ef39b05"
prediction_key = "5ce7589f7fcd4057bb01f7e62303eb58"
prediction_resource_id = "/subscriptions/f468ceaa-a610-4b88-9742-2b3e8f4ef76c/resourceGroups/Day2/providers/Micr
```

In [8]:
```
# Authenticate client

credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)
```

In [9]:
```
# Classifier

publish_iteration_name = "classifyModel"

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

# Create a new project
print ("Creating project...")
project = trainer.create_project("My New Project")
```

```
Creating project...
```

In [10]:
```
#Creating tags
cat_tag = trainer.create_tag(project.id, "cat")
dog_tag = trainer.create_tag(project.id, "dog")
lion_tag= trainer.create_tag(project.id, "lion")
```

In [18]:

```
import glob
import cv2
base_image_location = "C:/Users/Jaswanth Reddy/Desktop/Image dataset/api_cat_dog/"
base_image_location1 = "C:/Users/Jaswanth Reddy/Desktop/Image dataset/"


print("Adding images...")

image_list = []

for image_num in range(1, 20):
    file_name = "cat_{}.jpg".format(image_num)
    with open(base_image_location + "cat/" + file_name, "rb") as image_contents:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), tag_ids=[cat_tag.

for image_num in range(1, 20):
    file_name = "dog_{}.jpg".format(image_num)
    with open(base_image_location + "dog/" + file_name, "rb") as image_contents:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), tag_ids=[dog_tag.


for image_num in range(1, 20):
    file_name = "{}.jpg".format(image_num)
    with open(base_image_location1 + "lion/" + file_name, "rb") as image_contents:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), tag_ids=[lion_tag

upload_result = trainer.create_images_from_files(project.id, ImageFileCreateBatch(images=image_list))
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...
Image batch upload failed.
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  ErrorImageSize
Image status:  OK
Image status:  OK
Image status:  ErrorImageFormat
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
Image status:  OK
```

In [19]:

```python
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)
    time.sleep(1)

# The iteration is now trained. Publish it to the project endpoint
trainer.publish_iteration(project.id, iteration.id, publish_iteration_name, prediction_resource_id)
print ("Done!")
```

```
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
Training status: Training
```

Predicting image

In [20]:

```
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)

with open(r"C:\Users\Jaswanth Reddy\Desktop\Image dataset\cat_dog\train\cat\cat.2981.jpg", "rb") as image_conter
    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))
```

```
        cat: 99.99%
        lion: 0.12%
        dog: 0.08%
```

In [21]:

```
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)

with open(r"C:\Users\Jaswanth Reddy\Desktop\Image dataset\api_cat_dog\dog\dog_24.jpg", "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))
```

```
        dog: 99.95%
        cat: 0.36%
        lion: 0.15%
```

In [22]:

```
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)

with open(r"C:\Users\Jaswanth Reddy\Desktop\Image dataset\lion\lion0.jpg", "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))
```

```
        lion: 99.99%
        dog: 0.13%
        cat: 0.12%
```

In [ ]:

In [ ]: