2. Run the following commands on the terminal to move the training and testing datasets to the buckets:

   Train set features.

   ```
   gsutil cp X_train.csv gs://iris-sklearn
   ```

   Train set targets.

   ```
   gsutil cp y_train.csv gs://iris-sklearn
   ```

   Test sample for online prediction.

   ```
   gsutil cp test-sample.json gs://iris-sklearn
   ```

Prepare the Training Scripts

The code for training a Scikit-learn model on Cloud MLE is also prepared as a python package. The project structure is as follows:

Iris_SklearnCloudML: [project name as parent folder]

- Trainer: [folder containing the model and execution code]
  - __init__.py: [an empty special python file indicating that the containing folder is a Python package]
  - model.py: [file contains the logic of the model written in Scikit-learn]
- scripts: [folder containing scripts to execute jobs on Cloud MLE]
  - single-instance-training.sh: [script to run a single instance training job on Cloud MLE]
  - online-prediction.sh: [script to execute an online prediction job on Cloud MLE]
  - create-prediction-service.sh: [script to create a prediction service on Cloud MLE]
- config.yaml: [configuration file for specifying model version]

The model code for training on Cloud MLE with Scikit-learn (shown in the following) is stored in the file ‘model.py’. The machine learning algorithm used in this model is the Random forest Classifier.

```
# [START setup]
import datetime
import os
import subprocess
import sys
import pandas as pd

from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
from tensorflow.python.lib.io import file_io

# Fill in your Cloud Storage bucket name
BUCKET_ID = 'iris-sklearn'
# [END setup]

# [START download-and-load-into-pandas]
iris_data_filename = 'gs://iris-sklearn/X_train.csv'
iris_target_filename = 'gs://iris-sklearn/y_train.csv'

# Load data into pandas
with file_io.FileIO(iris_data_filename, 'r') as iris_data_f:
    iris_data = pd.read_csv(filepath_or_buffer=iris_data_f,
                        header=None, sep=',').values

with file_io.FileIO(iris_target_filename, 'r') as iris_target_f:
    iris_target = pd.read_csv(filepath_or_buffer=iris_target_f,
                        header=None, sep=',').values

iris_target = iris_target.reshape((iris_target.size,))
# [END download-and-load-into-pandas]

# [START train-and-save-model]
# Train the model
classifier = RandomForestClassifier()
classifier.fit(iris_data, iris_target)
```

```
# Export the classifier to a file
model = 'model.joblib'
joblib.dump(classifier, model)
# [END train-and-save-model]

# [START upload-model]
# Upload the saved model file to Cloud Storage
model_path = os.path.join('gs://', BUCKET_ID, 'model', datetime.datetime.
now().strftime(
    'iris_%Y%m%d_%H%M%S'), model)
subprocess.check_call(['gsutil', 'cp', model, model_path], stderr=sys.
stdout)
# [END upload-model]
```

Take note of the following points in the preceding code block:

- The code uses the 'file.io' module from the package 'tensorflow.python.lib.io' to stream a file stored on Cloud Storage.
- The rest of the code runs the classifier to build the model and exports the model to a bucket location on GCS. Cloud MLE will read from this bucket when building a prediction service for online predictions.

Execute a Scikit-learn Training Job on Cloud MLE

The bash code for executing a training job for the Scikit-learn model is presented in the following and is saved in the file 'single-instance-training.sh'.

```
export SCALE_TIER=BASIC # BASIC | BASIC_GPU | STANDARD_1 | PREMIUM_1 |
BASIC_TPU
DATE=`date '+%Y%m%d_%H%M%S'`
export JOB_NAME=iris_sklearn_$DATE
export GCS_JOB_DIR=gs://iris-sklearn/jobs/$JOB_NAME

echo $GCS_JOB_DIR

gcloud ml-engine jobs submit training $JOB_NAME \
                                    --stream-logs \
                                    --scale-tier $SCALE_TIER \
```