Preparing for Training and Serving on Cloud MLE

In this contrived example, we'll use the famous Iris dataset to train and serve a TensorFlow model using the Estimator API on Cloud MLE. To begin, let's walk through the following steps:

1. Create a bucket on GCS by running the gsutil mb command on the cloud terminal. Replace it with unique bucket name.

   ```
   export bucket_name=iris-dataset'
   gsutil mb gs://$bucket_name
   ```

2. Transfer training and test data from the code repository to the GCP bucket.

3. Move the train data.

   ```
   gsutil cp train_data.csv gs://$bucket_name
   ```

4. Move the train data.

   ```
   gsutil cp test_data.csv gs://$bucket_name
   ```

5. Move the hold-out data for batch predictions.

   ```
   gsutil cp hold_out_test.csv gs://$bucket_name
   ```

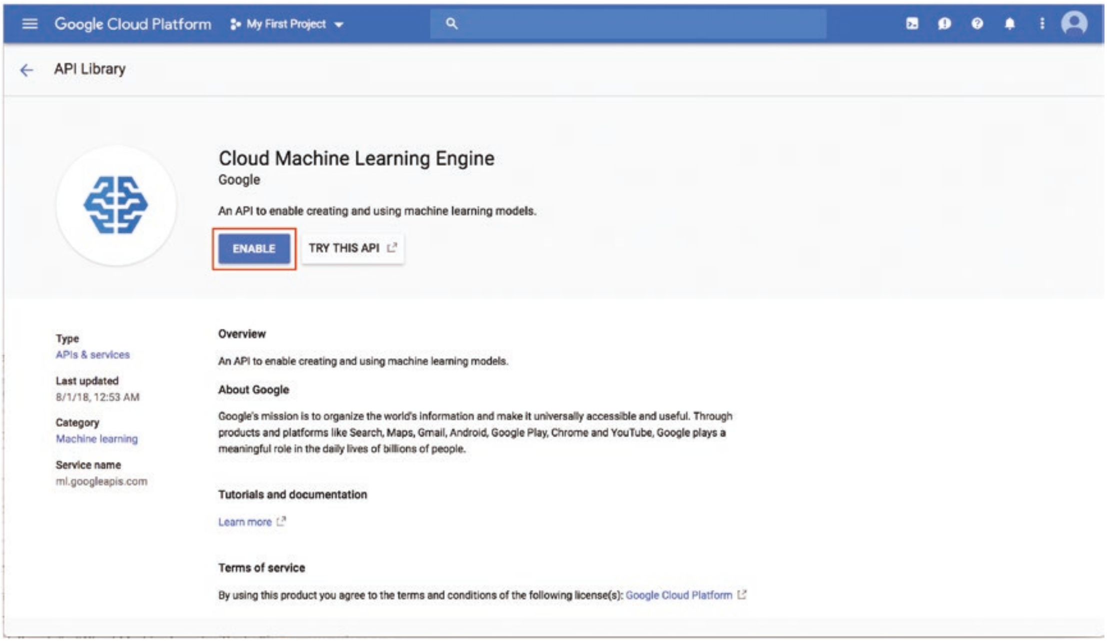
6. Enable the Cloud Machine Learning API to be able to create and use machine learning models on GCP Cloud MLE:

   a. Go to APIs & Services.

   b. Click "Enable APIs & Services".

   c. Search for "Cloud Machine Learning Engine".

   d. Click ENABLE API as shown in Figure 41-2.



Cloud Machine Learning Engine

Google

An API to enable creating and using machine learning models.

Type
APIs & services

Last updated
8/1/18, 12:53 AM

Category
Machine learning

Service name
ml.googleapis.com

Overview

An API to enable creating and using machine learning models.

About Google

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

Tutorials and documentation

Learn more

Terms of service

By using this product you agree to the terms and conditions of the following license(s): Google Cloud Platform

Figure 41-2. *Enable Cloud Machine Learning APIs*

Packaging the Code for Training on Cloud MLE

The code for training on Cloud MLE must be prepared as a python package. The recommended project structure is explained as follows:

IrisCloudML: [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: [script contains the logic of the model written in TensorFlow, Keras, etc.]
  - task.py: [script contains the application that orchestrates or manages the training job]
- scripts: [folder containing scripts to execute jobs on Cloud MLE]
  - distributed-training.sh: [script to run a distributed training job on Cloud MLE]