

Name : Jagruti Mohanty

Assignment 7 - VERTEX AI

a)

<https://codelabs.developers.google.com/vertex-pipelines-intro#0>

This screenshot shows the Google Cloud Platform Vertex AI Workbench interface. The left sidebar navigation bar includes options like Dashboard, Datasets, Features, Labeling tasks, Workbench (which is selected), Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main content area is titled 'Notebooks' and displays a message: 'As of the M80 DLVM release, all environments will include JupyterLab 3.x by default. To continue using an existing environment's JupyterLab 1.x version, disable auto-upgrade (if enabled) and do not manually upgrade the environment to a new environment version. To create new Notebooks with JupyterLab 1.x installed, see creating specific versions of Notebooks.' Below this message, there is a table header for 'Notebooks' with columns: Notebook name, Zone, Auto-upgrade, Environment, and Machine type. A note below the table says 'Notebooks have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. Learn more'. At the bottom of the main area, it says 'You don't have any notebooks in this project yet'. On the right side, there is an 'Info panel' showing user information: Jagruti Mohanty (jagruti.mohanty44@gmail.com), Jagruti Mohanty (jagruti.mohanty@gmail.com), Jagruti Mohanty (jagruti.mohanty@sjtu.edu), and Chatanya Kumar (s.chatanya.kumar19@gmail.com). There are also 'Add account' and 'Sign out' buttons.

This screenshot shows the 'New notebook' dialog box overlaid on the Vertex AI Workbench interface. The dialog box has a title 'New notebook' and a form for entering notebook details. The 'Notebook name' field contains 'tensorflow-2-3-20211203-114247'. Below it, a note says: '6-32 character limit with lowercase letters, digits, or '-' only. Must start with a letter. Cannot end with a '-'.' The 'Region' dropdown is set to 'us-west1 (Oregon)' and the 'Zone' dropdown is set to 'us-west1-b'. The 'Notebook properties' section contains the following settings: Environment (TensorFlow Enterprise 2.3 (with LTS and Intel® MKL-DNN/MKL)), Machine type (4 vCPUs, 15 GB RAM), Boot disk (100 GB Standard persistent disk), Data disk (100 GB Standard persistent disk), Subnetwork (default(10.138.0.0/20)), External IP (Ephemeral(Automatic)), and Permission (Compute Engine default service account). At the bottom of the dialog box are 'ADVANCED OPTIONS', 'CANCEL', and 'CREATE' buttons. The background of the interface shows the same 'Notebooks' list as the previous screenshot, with the 'Workbench' option selected in the sidebar.

← → ⌂ https://console.cloud.google.com/vertex-ai/workbench/list/instances?authuser=2&project=vertex-ai-334019

Google Cloud Platform Vertex-AI Search products and resources

Vertex AI Notebooks NEW NOTEBOOK REFRESH START STOP RESET DELETE HIDE INFO PANEL

Dashboard MANAGED NOTEBOOKS PREVIEW USER-MANAGED NOTEBOOKS EXECUTIONS PREVIEW SCHEDULES PREVIEW

Datasets Features Labeling tasks

Workbench Pipelines Training Experiments Models Endpoints Batch predictions Metadata Marketplace

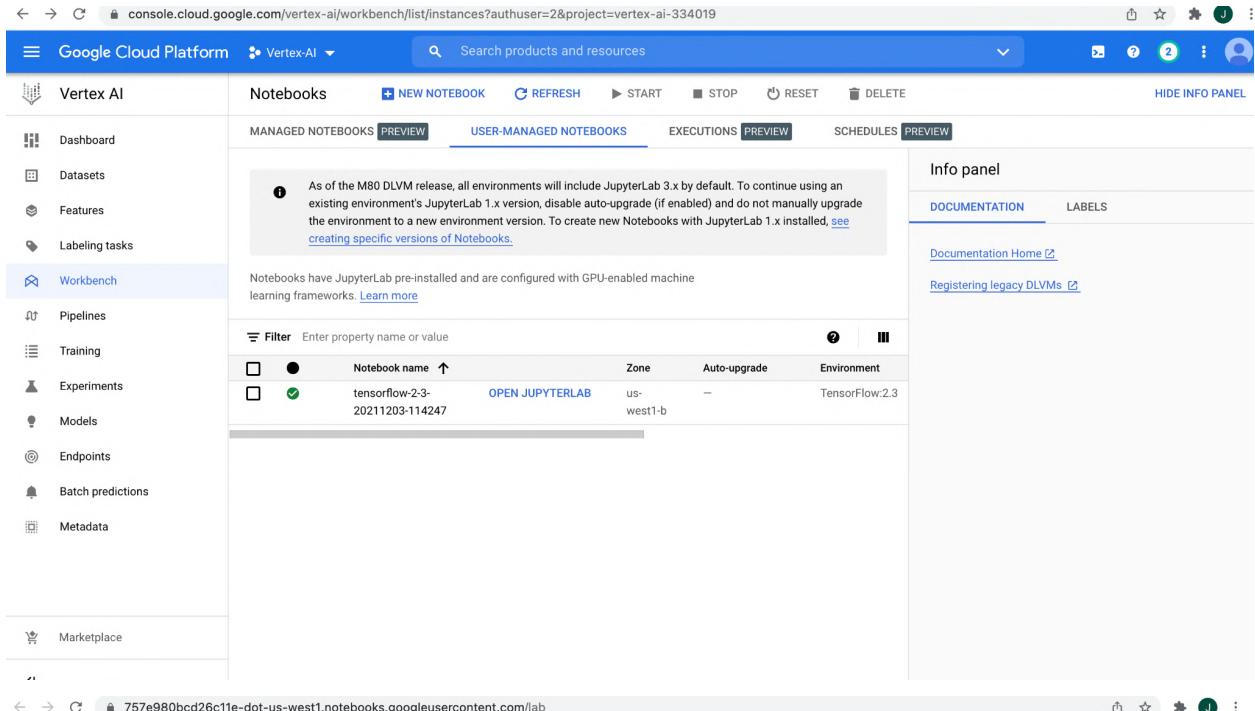
Notebooks have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. [Learn more](#)

Filter Enter property name or value

Notebook name ↑	Zone	Auto-upgrade	Environment
tensorflow-2-3-20211203-114247	OPEN JUPYTERLAB	us-west1-b	TensorFlow:2.3

Info panel DOCUMENTATION LABELS

Documentation Home [\[?\]](#)
Registering legacy DLVMs [\[?\]](#)



← → ⌂ https://757e980bcd26c11e-dot-us-west1.notebooks.googleusercontent.com/lab

File Edit View Run Kernel Git Tabs Settings Help

Launcher

Filter files by name

Name / Last Modified

- src 5 minutes ago
- tutorials 5 minutes ago

Notebook

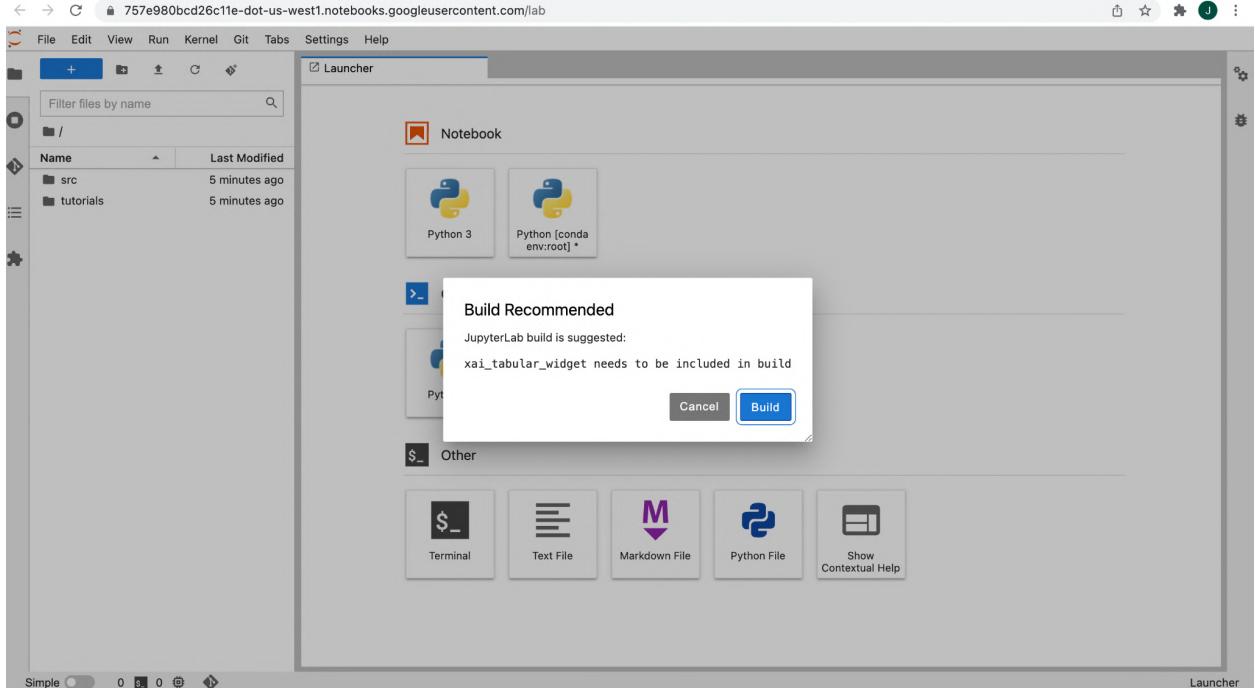
Python 3 Python [conda env:root] * [Build Recommended](#)

JupyterLab build is suggested:
xai_tabular_widget needs to be included in build

Cancel Build

Other

Terminal Text File Markdown File Python File Show Contextual Help



b)

<https://codelabs.developers.google.com/vertex-automl-tabular#0>

← → ⌂ console.cloud.google.com/vertex-ai/locations/us-central1/datasets/9199015655276281856/import?authuser=2&project=vertex-ai-334019

Google Cloud Platform Vertex-AI Search products and resources Google account: Jagruti Mohanty (jagruti.mohanty44@gmail.com)

Vertex AI fraud_detection

SOURCE ANALYZE

Dashboard Datasets

Features Labeling tasks Workbench Pipelines Training Experiments Models Endpoints Batch predictions Metadata Marketplace

Add data to your dataset

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)

Upload CSV files from your computer

Select CSV files from Cloud Storage

Select a table or view from BigQuery

Upload CSV files from your computer

Add up to 500 CSV files per upload. The files will be stored in a new Cloud Storage bucket ([charges apply](#)). Data from multiple files will be referenced as one dataset.

[SELECT FILES](#)

You can build two model types with tabular data. The model type is automatically chosen based on the data type of your target column.

- Regression models predict a numeric value. For example, predicting home prices or consumer spending.
- Classification models predict a category from a fixed number of categories. Examples include predicting whether an email is spam or not, or classes a student might be interested in attending.



\$625,000 \$975,000

← → C console.cloud.google.com/vertex-ai/locations/us-central1/datasets/9199015655276281856/import?authuser=2&project=vertex-ai-334019

Google Cloud Platform Vertex AI Search products and resources Google account: Jagruti Mohanty (jagruti.mohanty44@gmail.com)

Vertex AI fraud_detection

SOURCE ANALYZE

Add data to your dataset

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)

Upload CSV files from your computer

Select CSV files from Cloud Storage

Select a table or view from BigQuery

Upload CSV files from your computer

Add up to 500 CSV files per upload. The files will be stored in a new Cloud Storage bucket ([charges apply](#)). Data from multiple files will be referenced as one dataset.

SELECT FILES

SOURCE ANALYZE

Add data to your dataset

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)

Upload CSV files from your computer

Select CSV files from Cloud Storage

Select a table or view from BigQuery

Select a table or view from BigQuery

Use a BigQuery table or view as your data source. You'll need [permission to access the dataset](#) and get the [dataset ID and table ID](#). [Learn more](#)

BigQuery path *

bigquery-public-data.ml_datasets.ulb_fraud_detection **BROWSE**

Select path

No suggestions to display

The selected BigQuery table will be associated with your dataset. Making changes to the referenced BigQuery table will affect the dataset before training.

CONTINUE

← → C https://console.cloud.google.com/vertex-ai/locations/us-central1/datasets/9199015655276281856/analyze?authuser=2&project=vertex-ai-334019

Google Cloud Platform Vertex-AI Search products and resources

Vertex AI fraud_detection

Dashboard Datasets

Features Labeling tasks Workbench Pipelines Training Experiments Models Endpoints Batch predictions Metadata Marketplace

SOURCE ANALYZE

Dataset Info

Created: Dec 03, 2021 11:51 AM
Dataset format: BigQuery
Dataset location(s): bq://bigquery-public-data_fraud_detection

Summary

Total columns: 31
Total rows: -

FLOAT 30 (96.77%)
INTEGER 1 (3.23%)

GENERATE STATISTICS

Filter Enter property name or value

Column name	BigQuery type	BigQuery mode	Missing % (count)	Distinct values
Amount	FLOAT	NULLABLE	-	-
Class	INTEGER	NULLABLE	-	-
Time	FLOAT	NULLABLE	-	-
V1	FLOAT	NULLABLE	-	-
V10	FLOAT	NULLABLE	-	-
V11	FLOAT	NULLABLE	-	-

Training jobs and models

Use this dataset and annotation set to train a new machine learning model with AutoML or custom code

TRAIN NEW MODEL

← → C https://console.cloud.google.com/vertex-ai/locations/us-central1/datasets/9199015655276281856/analyze?authuser=2&project=vertex-ai-334019

Google Cloud Platform Vertex AI

Train new model

1 Training method
2 Model details
3 Training options
4 Compute and pricing

Model name * fraud_detection_2021123195477
Target column * Class (INTEGER)

Export test dataset to BigQuery

ADVANCED OPTIONS

CONTINUE

START TRAINING CANCEL

← → C https://console.cloud.google.com/vertex-ai/locations/us-central1/datasets/9199015655276281856/analyze?authuser=2&project=vertex-ai-334019

Google Cloud Platform Vertex AI

Train new model

1 Training method
2 Model details
3 Training options
4 Compute and pricing

START TRAINING CANCEL

Enter the maximum number of node hours you want to spend training your model.

You can train for as little as 1 node hour. You may also be eligible to train with free node hours. [Pricing guide](#)

Budget * 1 Maximum node hours

Estimated completion date: Dec 3, 2021 1 PM GMT-8

Enable early stopping

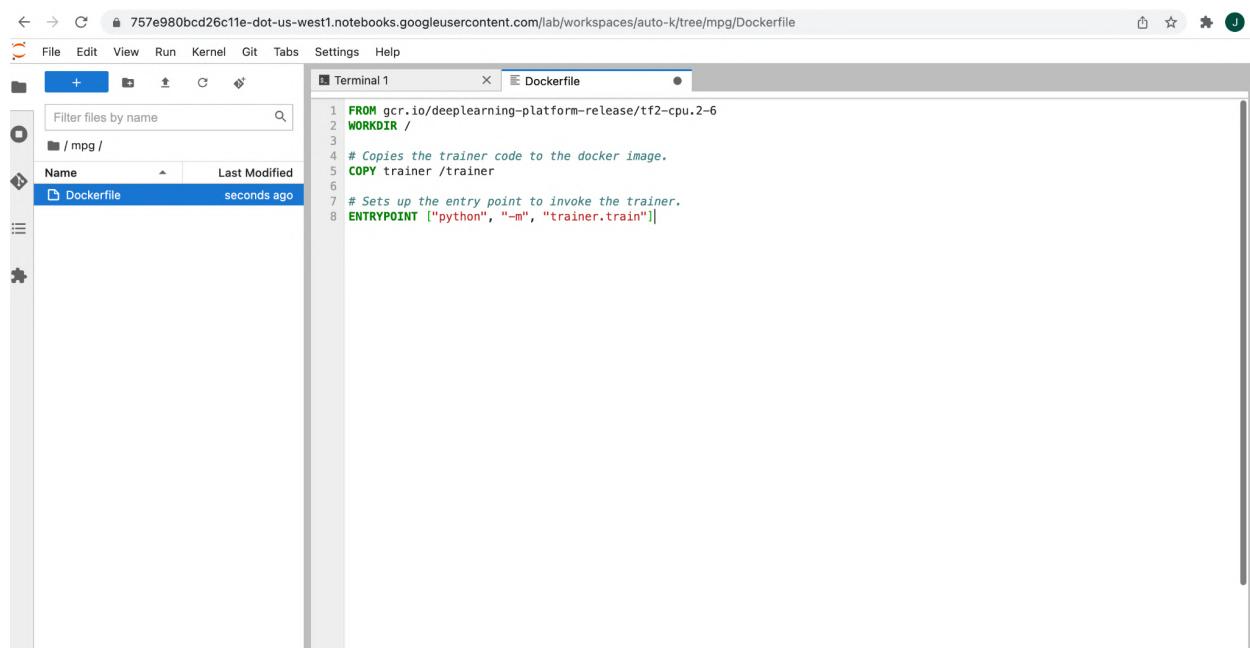
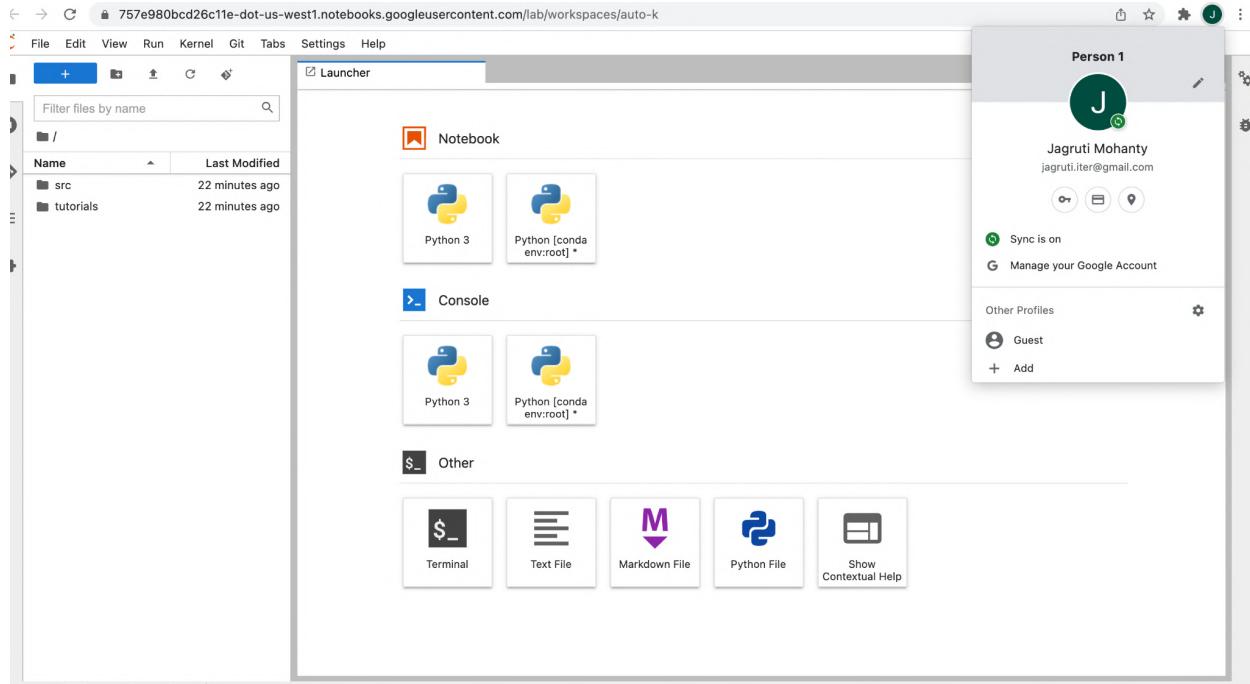
Ends model training when no more improvements can be made and refunds leftover training budget. If early stopping is disabled, training continues until the budget is exhausted.

The screenshot shows the Google Cloud Platform Vertex AI interface. On the left, there's a sidebar with various options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models (which is selected), Endpoints, Batch predictions, and Metadata. The main area has tabs for EVALUATE, DEPLOY & TEST (which is active), BATCH PREDICTIONS, and MODEL PROPERTIES. Under DEPLOY & TEST, it says "Model to run on a Docker container." Below that is a section titled "Deploy your model" with a note about endpoints. A "DEPLOY TO ENDPOINT" button is visible. A table lists an endpoint named "endpointfraud" with ID 141731446866837504, status "Deploying model", 0 models, and region "us-central1". The last updated time is Dec 3, 2021, 2:34:54 PM. At the bottom, there's a "Test your model" section with a "PREVIEW" button. It says "Your model must be successfully deployed to an endpoint before you can test it." There's a table for feature columns with "Time" as the column name, "Numerical" as the type, and "Required" as the requirement. The value is listed as "84883". To the right, it says "Predicted column not yet known" and "Prediction result".

c)

https://codelabs.developers.google.com/vertex_custom_training_prediction#0

(Links to an external site.)



The screenshot shows a Jupyter Notebook interface with a code editor and a terminal window.

Code Editor (train.py):

```

1 import numpy as np
2 import pandas as pd
3 import pathlib
4 import tensorflow as tf
5
6 from tensorflow import keras
7 from tensorflow.keras import layers
8
9 print(tf.__version__)
10
11 """## The Auto MPG dataset
12
13 The dataset is available from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/).
14
15 ### Get the data
16 First download the dataset.
17 """
18
19 dataset_path = keras.utils.get_file("auto-mpg.data", "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data")
20 dataset_path
21
22 """Import it using pandas"""
23
24 column_names = ['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight',
25                 'Acceleration', 'Model Year', 'Origin']
26 dataset = pd.read_csv(dataset_path, names=column_names,
27                       na_values = "?", comment="#",
28                       sep=",", skipinitialspace=True)
29
30 dataset.tail()
31
32 # TODO: replace 'your-gcs-bucket' with the name of the Storage bucket you created earlier
33 BUCKET = 'gs://ai-334019-bucket'
34
35 """## Clean the data
36
37 The dataset contains a few unknown values.
38 """

```

Terminal:

```
(base) jupyter@tensorflow-2-3-20211203-114247:~$ mkdir mpg
(base) jupyter@tensorflow-2-3-20211203-114247:~$ cd mpg
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ touch Dockerfile
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ gcloud config list --format 'value(core.project)'
vertex-ai-334019
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ PROJECT_ID=vertex-ai-334019
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ BUCKET_NAME="gs://${PROJECT_ID}-bucket"
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ BUCKET_NAME="gs://${vertex-ai-334019}-bucket"
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ gsutil mb -l us-central1 ${BUCKET_NAME}
Creating gs://ai-334019-bucket/...
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ mkdir trainer
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ touch trainer/train.py
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ IMAGE_URI="gcr.io/$vertex-ai-334019/mpg:v1"
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ docker build ./ -t ${IMAGE_URI}
invalid argument "gcr.io/ai-334019/mpg:v1" for "-t, --tag" flag: invalid reference format
See 'docker build --help'.
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ IMAGE_URI="gcr.io/$vertex-ai-334019/mpg:v1"

```

The screenshot shows a Jupyter Notebook interface with a code editor and a terminal window.

Code Editor (train.py):

```
(base) jupyter@tensorflow-2-3-20211203-114247:~$ mkdir mpg
(base) jupyter@tensorflow-2-3-20211203-114247:~$ cd mpg
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ touch Dockerfile
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ gcloud config list --format 'value(core.project)'
vertex-ai-334019
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ PROJECT_ID=vertex-ai-334019
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ BUCKET_NAME="gs://${PROJECT_ID}-bucket"
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ BUCKET_NAME="gs://${vertex-ai-334019}-bucket"
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ gsutil mb -l us-central1 ${BUCKET_NAME}
Creating gs://ai-334019-bucket/...
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ mkdir trainer
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ touch trainer/train.py
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ IMAGE_URI="gcr.io/$vertex-ai-334019/mpg:v1"
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ docker build ./ -t ${IMAGE_URI}
invalid argument "gcr.io/ai-334019/mpg:v1" for "-t, --tag" flag: invalid reference format
See 'docker build --help'.
(base) jupyter@tensorflow-2-3-20211203-114247:~/mpg$ IMAGE_URI="gcr.io/$vertex-ai-334019/mpg:v1"
```

d) Big query :

<https://codelabs.developers.google.com/vertex-workbench-intro#0>

← → C https://console.cloud.google.com/vertex-ai/workbench/list/instances?authuser=4&project=vertex-ai-334021

Notebook name	Zone	Auto-upgrade	Environment
london-bikes-codelabs	us-central1-a	—	TensorFlow:2.3
tensorflow-2-3-20211203-133926	us-west1-b	—	TensorFlow:2.3

← → C https://console.cloud.google.com/vertex-ai/workbench/create-managed?authuser=4&project=vertex-ai-334021

Notebook name *
managed-notebook-1638572301
Name must be 63 characters or less, must start with a letter and include only lowercase letters, digits, or '-'.

Region *
us-central1 (Iowa)

Advanced settings

CREATE **CANCEL**

Quota exceeded for quota metric 'Create Runtime API requests' and limit 'Create Runtime API requests per minute' of service 'notebooks.googleapis.com' for consumer 'project_number:310606389043'.

f) https://codelabs.developers.google.com/vertex_hyperparameter_tuning#0

File Edit View Run Kernel Git Tabs Settings Help

Terminal 2 task.py Dockerfile

```
1 FROM gcr.io/deeplearning-platform-release/tf2-gpu.2-5
2
3 WORKDIR /
4
5 # Installs hypertune library
6 RUN pip install cloudml-hypertune
7
8 # Copies the trainer code to the docker image.
9 COPY trainer /trainer
10
11 # Sets up the entry point to invoke the trainer.
12 ENTRYPOINT ["python", "-m", "trainer.task"]
```

File Edit View Run Kernel Git Tabs Settings Help

Terminal 2 task.py Dockerfile

```
1 import tensorflow as tf
2 import tensorflow_datasets as tfds
3 import argparse
4 import hypertune
5
6 NUM_EPOCHS = 10
7
8 def get_args():
9     """Parses args. Must include all hyperparameters you want to tune."""
10    parser = argparse.ArgumentParser()
11    parser.add_argument(
12        '--learning_rate',
13        required=True,
14        type=float,
15        help='learning rate')
16    parser.add_argument(
17        '--momentum',
18        required=True,
19        type=float,
20        help='SGD momentum value')
21    parser.add_argument(
22        '--num_neurons',
23        required=True,
24        type=int,
25        help='number of units in last hidden layer')
26    args = parser.parse_args()
27    return args
28
29
30
31 def preprocess_data(image, label):
32     """Resizes and scales images."""
33
34     image = tf.image.resize(image, (150,150))
35     return tf.cast(image, tf.float32) / 255., label
36
37
38
39 def create_dataset():
```

The screenshot shows a Jupyter Notebook environment with two tabs open: 'task.py' and 'Dockerfile'. The 'task.py' tab contains Python code for a neural network model and hyperparameter tuning. The 'Dockerfile' tab shows a Dockerfile for the project. Below the notebook is a terminal window displaying the command-line output of a Docker build process.

```

task.py content (partial):
59     inputs = tf.keras.Input(shape=(150, 150, 3))
60     x = tf.keras.layers.Conv2D(16, (3, 3), activation='relu')(inputs)
61     x = tf.keras.layers.MaxPooling2D((2, 2))(x)
62     x = tf.keras.layers.Conv2D(32, (3, 3), activation='relu')(x)
63     x = tf.keras.layers.MaxPooling2D((2, 2))(x)
64     x = tf.keras.layers.Conv2D(64, (3, 3), activation='relu')(x)
65     x = tf.keras.layers.MaxPooling2D((2, 2))(x)
66     x = tf.keras.layers.Flatten()(x)
67     x = tf.keras.layers.Dense(num_neurons, activation='relu')(x)
68     outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)
69     model = tf.keras.Model(inputs, outputs)
70     model.compile(
71         loss='binary_crossentropy',
72         optimizer=tf.keras.optimizers.SGD(learning_rate=learning_rate, momentum=momentum),
73         metrics=['accuracy'])
74
75
76
77 def main():
78     args = get_args()
79     train_data, validation_data = create_dataset()
80     model = create_model(args.num_neurons, args.learning_rate, args.momentum)
81     history = model.fit(train_data, epochs=NUM_EPOCHS, validation_data=validation_data)
82
83     # DEFINE METRIC
84     hp_metric = history.history['val_accuracy'][-1]
85
86     hpt = hypertune.HyperTune()
87     hpt.report_hyperparameter_tuning_metric(
88         hyperparameter_metric_tag='accuracy',
89         metric_value=hp_metric,
90         global_step=NUM_EPOCHS)
91
92
93 if __name__ == "__main__":
94     main()

```

```

Dockerfile content (partial):
FROM tensorflow/tensorflow:2.3.0
WORKDIR /horses_or_humans
COPY . .
RUN pip install --upgrade pip
RUN pip install --upgrade tensorflow-hypertune
RUN python task.py > /dev/null

```

```

Terminal 2 output (partial):
(base) jupyter@tensorflow-2-3-20211203-133926:~$ mkdir horses_or_humans
(base) jupyter@tensorflow-2-3-20211203-133926:~$ cd horses_or_humans
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ touch Dockerfile
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ touch trainer/task.py
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ touch trainer/main.py
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ PROJECT_ID='vertex-ai-334021'
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ IMAGE_URI='gcr.io/$PROJECT_ID/horse-human:hypertune'
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ docker build ./ -t $IMAGE_URI
invalid argument "gcr.io/-ai-334021/horse-human:hypertune" for "-t, --tag" flag: invalid reference format
See 'docker build --help'.
(base) jupyter@tensorflow-2-3-20211203-133926:~/horses_or_humans$ docker push $IMAGE_URI

```

g) <https://codelabs.developers.google.com/vertex-mlmd-pipelines#0>

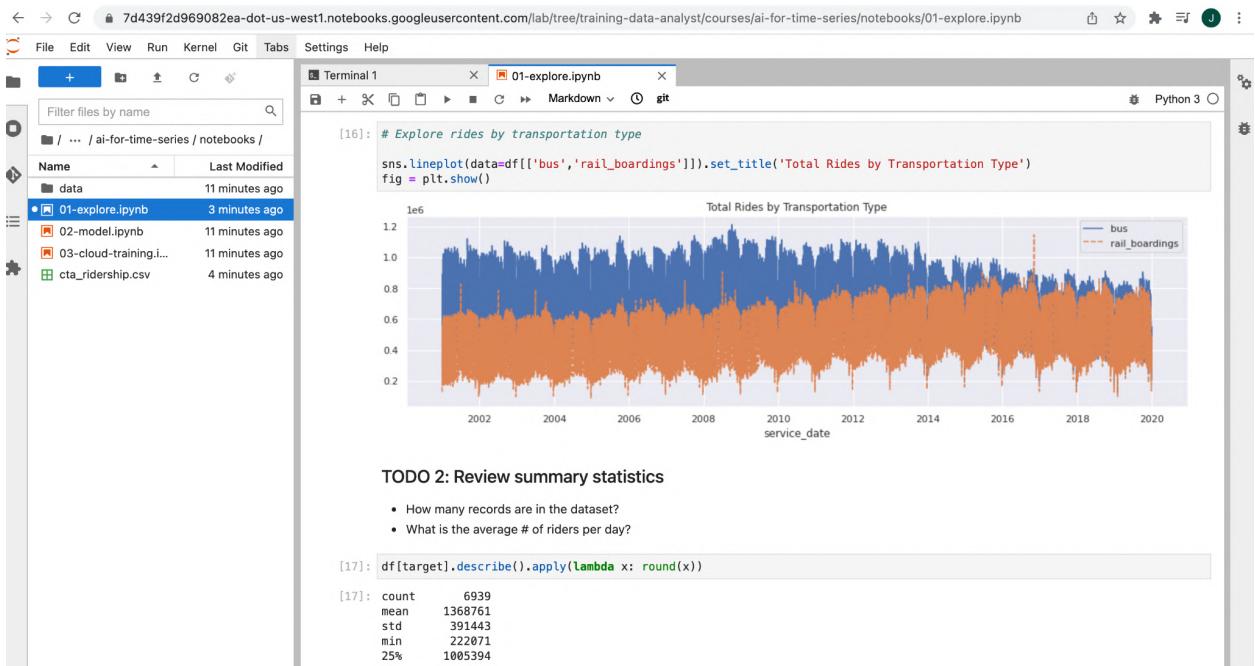
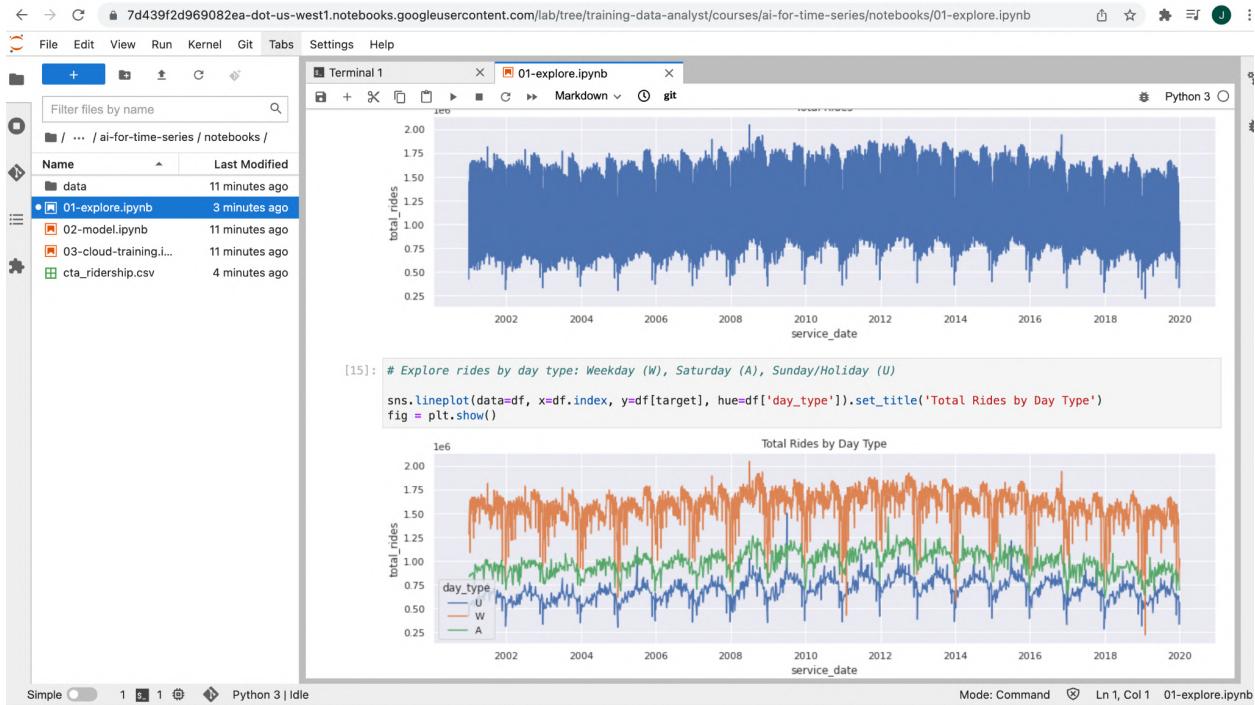
← → C https://console.cloud.google.com/vertex-ai/workbench/create-managed?authuser=4&project=vertex-ai-334021

The screenshot shows the Google Cloud Platform Vertex AI Workbench interface. On the left, there's a sidebar with options like Dashboard, Datasets, Features, Labeling tasks, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The 'Workbench' section is currently selected. In the main area, there's a form to 'Create a managed notebook'. The 'Notebook name' field contains 'managed-notebook-1638572301'. Below it, a note says: 'Name must be 63 characters or less, must start with a letter and include only lowercase letters, digits, or _.' The 'Region' dropdown is set to 'us-central1 (Iowa)'. Under 'Advanced settings', there are 'CREATE' and 'CANCEL' buttons. A modal window in the center displays an error message: 'Quota exceeded for quota metric 'Create Runtime API requests' and limit 'Create Runtime API requests per minute' of service 'notebooks.googleapis.com' for consumer 'project_number:310606389043'.'

← → C https://console.cloud.google.com/vertex-ai/locations/us-central1/models/5366254460290990080/deploy?authuser=2&project=vertex-ai-334019

The screenshot shows the Google Cloud Platform Vertex AI Model interface. The sidebar includes options like Dashboard, Datasets, Features, Labeling tasks, Pipelines, Training, Experiments, Models (which is selected), Endpoints, Batch predictions, and Metadata. The main area shows a model named 'fraud_detection_2021123195457'. It has tabs for EVALUATE, DEPLOY & TEST (which is selected), BATCH PREDICTIONS, and MODEL PROPERTIES. Under 'DEPLOY your model', it says: 'Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.' There's a 'DEPLOY TO ENDPOINT' button. Below it, a table lists a single endpoint: 'endpointfraud' (ID: 141731446866837504, Status: Active, Models: 0, Region: us-central1, Monitoring: Disabled). The 'Test your model' section has a 'PREVIEW' tab. It shows a table with columns: Feature column name, Type, Required or optional, Value, and Local feature. The rows are: Time (Numerical, Required, Value: 84883, Local feature: --), V1 (Numerical, Required, Value: 0.0218339018594994, Local feature: --), V2 (Numerical, Required, Value: 0.0672124249198848, Local feature: --), and V3 (Numerical, Required, Value: --, Local feature: --). To the right, there are sections for 'Predicted column not yet known' and 'Prediction result'.

h) <https://codelabs.developers.google.com/codelabs/automl-forecasting-with-vertex-ai#0>



File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 01-explore.ipynb

```
min      222071
25%    1005394
50%    1548343
75%    1660947
max     2049519
Name: total_rides, dtype: int64
```

TODO 3: Explore seasonality

- Is there much difference between months?
- Can you extract the trend and seasonal pattern from the data?

```
[18]: # Show the distribution of values for each day of the week in a boxplot:
# Min, 25th percentile, median, 75th percentile, max

daysofweek = df.index.to_series().dt.dayofweek

fig = sns.boxplot(x=daysofweek, y=df[target])
```

File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 01-explore.ipynb

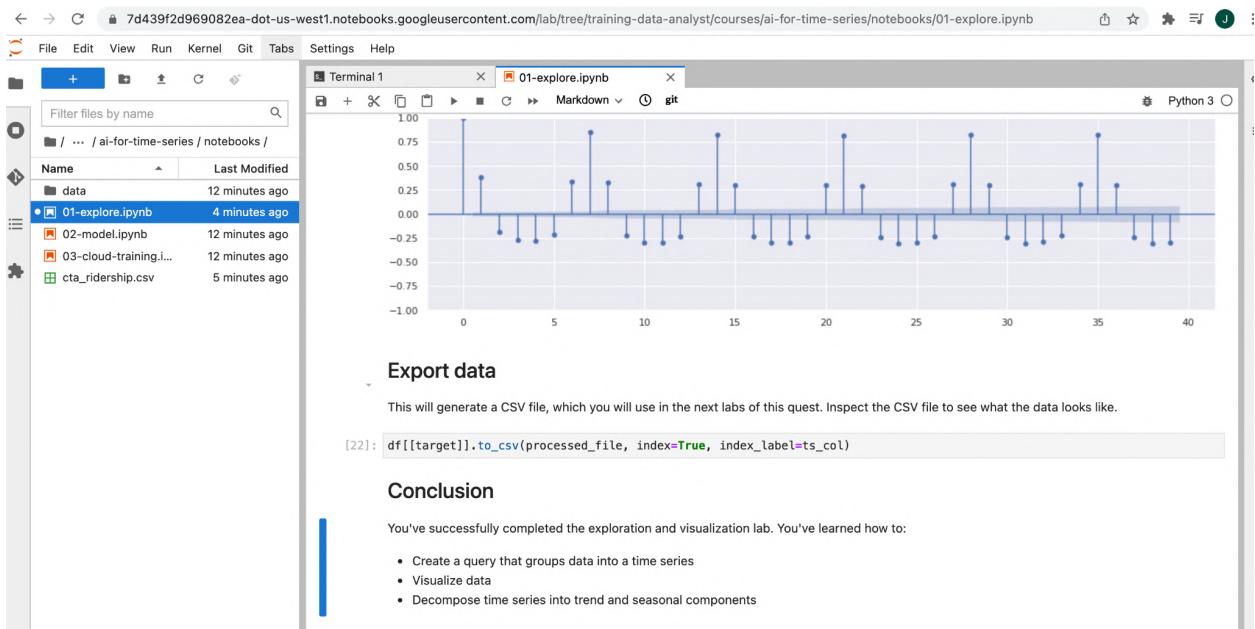
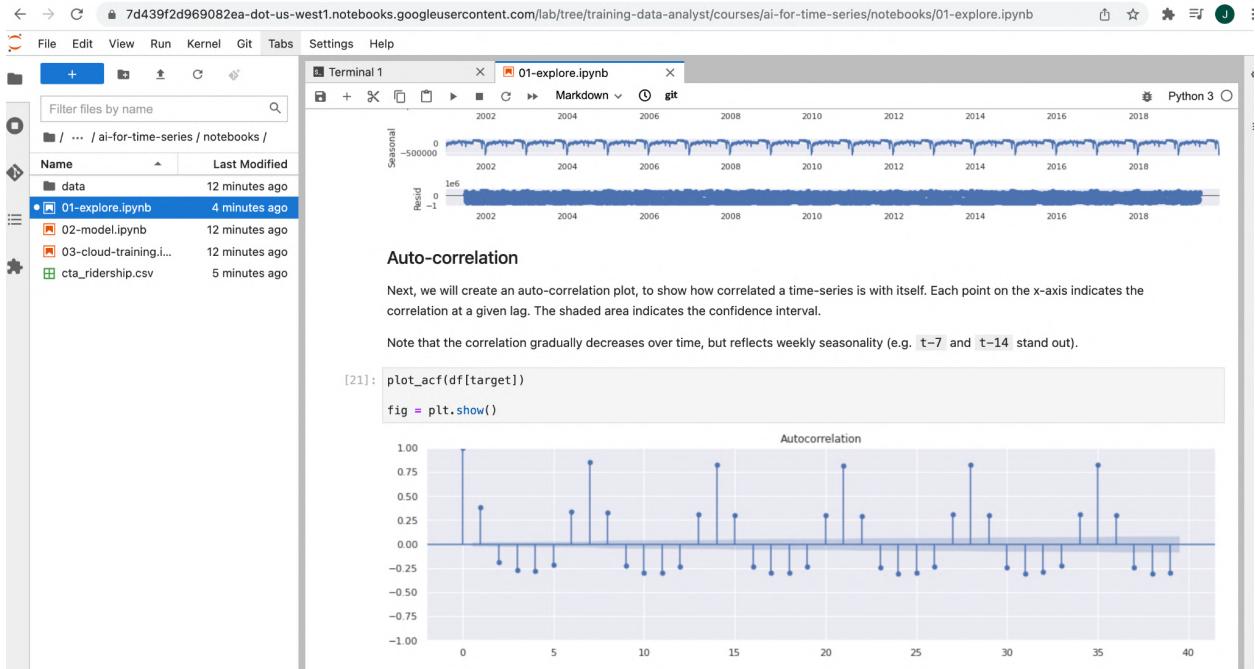
```
[19]: # Show the distribution of values for each month in a boxplot:

months = df.index.to_series().dt.month

fig = sns.boxplot(x=months, y=df[target])
```

```
[20]: # Decompose the data into trend and seasonal components

result = seasonal_decompose(df[target], period=365)
fig = result.plot()
```



- i) https://codelabs.developers.google.com/vertex_multiworker_training#0

File Edit View Run Kernel Git Tabs Settings Help

Terminal 2 task.py Dockerfile

```
1 FROM gcr.io/deeplearning-platform-release/tf2-gpu.2-5
2
3 WORKDIR /
4
5 # Installs hypertune library
6 RUN pip install cloudml-hypertune
7
8 # Copies the trainer code to the docker image.
9 COPY trainer /trainer
10
11 # Sets up the entry point to invoke the trainer.
12 ENTRYPOINT ["python", "-m", "trainer.task"]
```

File Edit View Run Kernel Git Tabs Settings Help

Terminal 2 task.py Dockerfile

```
1 import tensorflow as tf
2 import tensorflow_datasets as tfds
3 import argparse
4 import hypertune
5
6 NUM_EPOCHS = 10
7
8 def get_args():
9     """Parses args. Must include all hyperparameters you want to tune."""
10    parser = argparse.ArgumentParser()
11    parser.add_argument(
12        '--learning_rate',
13        required=True,
14        type=float,
15        help='learning rate')
16    parser.add_argument(
17        '--momentum',
18        required=True,
19        type=float,
20        help='SGD momentum value')
21    parser.add_argument(
22        '--num_neurons',
23        required=True,
24        type=int,
25        help='number of units in last hidden layer')
26    args = parser.parse_args()
27    return args
28
29
30
31 def preprocess_data(image, label):
32     """Resizes and scales images."""
33
34     image = tf.image.resize(image, (150,150))
35     return tf.cast(image, tf.float32) / 255., label
36
37
38
39 def create_dataset():
```

Training pipelines are the primary model training workflow in Vertex AI. You can use training pipelines to create an AutoML-trained model or a custom-trained model. For custom-trained models, training pipelines orchestrate custom training jobs and hyperparameter tuning with additional steps like adding a data set or uploading the model to Vertex AI for prediction serving. [Learn more](#)

Region
us-central1 (Iowa) ▾ ⚙

Filter Enter a property name

Name	ID	Job type	Model type
multiworker-cassava	2119053575840595968	Training pipeline	Custom

The training job will take about 30-35 minutes to complete.

Congratulations! 🎉

You've learned how to use Vertex AI to:

- Launch a multi-worker training job for training code provided in a custom container. You used a TensorFlow model in this example, but you can train a model built with any framework using custom or built-in containers.

j) <https://codelabs.developers.google.com/codelabs/bqml-vertex-prediction#0>

Training pipelines are the primary model training workflow in Vertex AI. You can use training pipelines to create an AutoML-trained model or a custom-trained model. For custom-trained models, training pipelines orchestrate custom training jobs and hyperparameter tuning with additional steps like adding a data set or uploading the model to Vertex AI for prediction serving. [Learn more](#)

Region
us-central1 (Iowa) ▾ ⚙

Filter Enter a property name

Name	ID	Job type	Model type
multiworker-cassava	2119053575840595968	Training pipeline	Custom

The training job will take about 30-35 minutes to complete.

Congratulations! 🎉

You've learned how to use Vertex AI to:

- Launch a multi-worker training job for training code provided in a custom container. You used a TensorFlow model in this example, but you can train a model built with any framework using custom or built-in containers.

File Edit View Run Kernel Git Tabs Settings Help

Terminal 2 task.py Untitled.ipynb Dockerfile Python 3

```
[ ]: %%writefile default-pred.json
{
    "instances": [
        {"age": 39,
         "bill_amt_1": 47174,
         "bill_amt_2": 47974,
         "bill_amt_3": 48630,
         "bill_amt_4": 50803,
         "bill_amt_5": 30789,
         "bill_amt_6": 15874,
         "education_level": "1",
         "limit_balance": 50000,
         "marital_status": "2",
         "pay_0": 0,
         "pay_2": 0,
         "pay_3": 0,
         "pay_4": 0,
         "pay_5": "0",
         "pay_6": "0",
         "pay_amt_1": 1800,
         "pay_amt_2": 2000,
         "pay_amt_3": 3000,
         "pay_amt_4": 2000,
         "pay_amt_5": 2000,
         "pay_amt_6": 2000,
         "sex": "1"}
    ]
}

[ ]: REGION="us-central1" # either us-central1, europe-west4, or asia-east1
```

The screenshot shows a Jupyter Notebook interface with the following components:

- File Browser:** On the left, it shows a directory structure under "/horses_or_humans/". The "Untitled.ipynb" file is selected.
- Code Editor:** The main area contains Python code. It includes a JSON object definition and a curl command for making a POST request to a Google Cloud endpoint.
- Terminal:** A terminal window at the bottom shows the command `curl` being run with specific headers and a JSON payload.

```

{
    "instances": [
        {"age": 39,
         "bill_amt_1": 47174,
         "bill_amt_2": 47974,
         "bill_amt_3": 48630,
         "bill_amt_4": 50803,
         "bill_amt_5": 30789,
         "bill_amt_6": 15874,
         "education_level": "1",
         "limit_balance": 50000,
         "marital_status": "2",
         "pay_0": 0,
         "pay_2": 0,
         "pay_3": 0,
         "pay_4": 0,
         "pay_5": 0,
         "pay_6": 0,
         "pay_ant_1": 1800,
         "pay_ant_2": 2000,
         "pay_ant_3": 3000,
         "pay_ant_4": 2000,
         "pay_ant_5": 2000,
         "pay_ant_6": 2000,
         "sex": "1"
    ]
}

[ ]: REGION="us-central1" # either us-central1, europe-west4, or asia-east1

[ ]: !curl \
-X POST \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-H "Content-Type: application/json" \
https://us-central1-prediction-aiplatform.googleapis.com/v1alpha1/projects/$PROJECT_ID/locations/$REGION/endpoints/$ENDPOINT \
-d "@default-pred.json"

```

k) <https://codelabs.developers.google.com/vertexx-xgb-wit#0>

The screenshot shows a Jupyter Notebook interface with the following components:

- File Browser:** On the left, it shows a directory structure under "/". The "src" and "tutorials" files are selected.
- Terminal:** The main area shows the output of the command `pip3 install xgboost==1.2`, which installs the xgboost package.

```

(base) jupyter@london-bikes-codelabs:~$ pip3 install xgboost==1.2
Collecting xgboost==1.2
  Downloading xgboost-1.2.0-py3-none-manylinux2010_x86_64.whl (148.9 MB)
    ██████████ 148.9 MB 24 kB/s
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from xgboost==1.2) (1.7.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from xgboost==1.2) (1.19.5)
Installing collected packages: xgboost
Successfully installed xgboost-1.2.0
(base) jupyter@london-bikes-codelabs:~$ 

```

[Untitled.ipynb](#)

```
[1]: import pandas as pd
import xgboost as xgb
import numpy as np
import collections
import wikitext

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder

[2]: !gsutil cp 'gs://mortgage_dataset_files/mortgage-small.csv' .
Copying gs://mortgage_dataset_files/mortgage-small.csv...
| [1 files] [330.8 MiB/330.8 MiB]
Operation completed over 1 objects/330.8 MiB.

[3]: COLUMN_NAMES = collections.OrderedDict({
    'as_of_year': np.int16,
    'agency_code': 'category',
    'loan_type': 'category',
    'property_type': 'category',
    'loan_purpose': 'category',
    'occupancy': np.int8,
    'loan_amt_thousands': np.float64,
    'preapproval': 'category',
    'county_code': np.float64,
    'applicant_income_thousands': np.float64,
    'purchaser_type': 'category',
    'hoepa_status': 'category',
    'lien_status': 'category',
    'population': np.float64,
    'population': np.float64,
})
```

[Untitled.ipynb](#)

```
[4]: COLUMN_NAMES = collections.OrderedDict({
    'loan_type': 'category',
    'property_type': 'category',
    'loan_purpose': 'category',
    'occupancy': np.int8,
    'loan_amt_thousands': np.float64,
    'preapproval': 'category',
    'county_code': np.float64,
    'applicant_income_thousands': np.float64,
    'purchaser_type': 'category',
    'hoepa_status': 'category',
    'lien_status': 'category',
    'population': np.float64,
    'ffiec_median_fair_income': np.float64,
    'tract_to_msa_income_pct': np.float64,
    'num_owner_occupied_units': np.float64,
    'num_1_to_4_family_units': np.float64,
    'approved': np.int8
})

[5]: data = pd.read_csv(
    'mortgage-small.csv',
    index_col=False,
    dtype=COLUMN_NAMES
)
data = data.dropna()
data = shuffle(data, random_state=2)
data.head()
```

	as_of_year	agency_code	loan_type	property_type	loan_purpose	occupancy	loan_amt_thousands	preapproval	county_code	applicant
310650	2016	Consumer Financial Protection Bureau (CFPB)	Conventional (any loan other than FHA, VA, FSA...)	One to four-family (other than manufactured ho...	Refinancing	1	110.0	Not applicable	119.0	
630129	2016	Department of Housing and Urban Development	Conventional (any loan other than FHA, VA, ...)	One to four-family (other than manufactured ho...	Home purchase	1	480.0	Not applicable	33.0	

← → C c27f2704da25242-dot-us-central1.notebooks.googleusercontent.com/lab/tree/Untitled.ipynb

File Edit View Run Kernel Git Tabs Settings Help

SEARCH

WARNING

The JupyterLab development team is excited to have a robust third-party extension community. However, we do not review third-party extensions, and some extensions may introduce security risks or contain malicious code that runs on your machine.

Enable

INSTALLED

DISCOVER

Untitled.ipynb

Code git

Administration (NCUA) FHA, VA, manufactured FSA... applicable Python 3

```
[6]: # Class labels - 0: denied, 1: approved
print(data['approved'].value_counts())

labels = data['approved'].values
data = data.drop(columns=['approved'])

1    665389
0    334610
Name: approved, dtype: int64
```

```
[7]: dummy_columns = list(data.dtypes[data.dtypes == 'category'].index)
data = pd.get_dummies(data, columns=dummy_columns)

data.head()
```

```
[7]:
```

	as_of_year	occupancy	loan_amt_thousands	county_code	applicant_income_thousands	population	ffiec_median_fam_income	tract_to_n
310650	2016	1	110.0	119.0	55.0	5930.0	64100.0	
630129	2016	1	480.0	33.0	270.0	4791.0	90300.0	
715484	2016	2	240.0	59.0	96.0	3439.0	105700.0	
887708	2016	1	76.0	65.0	85.0	3952.0	61300.0	
719598	2016	1	100.0	127.0	70.0	2422.0	46400.0	

5 rows × 44 columns

```
[9]: x,y = data.values,labels
x_train,x_test,y_train,y_test = train_test_split(x,y)
```

Untitled.ipynb

```
[9]: x,y = data.values,labels
x_train,x_test,y_train,y_test = train_test_split(x,y)

[10]: model = xgb.XGBClassifier(
    objective='reg:logistic'
)

[11]: model.fit(x_train, y_train)

[12]: y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')
0.874616

[13]: model.save_model('model.bst')

[14]: num_wit_examples = 500
test_examples = np.hstack((x_test[:num_wit_examples],y_test[:num_wit_examples].reshape(-1,1)))

[16]: config_builder = (WitConfigBuilder(test_examples.tolist(), data.columns.tolist() + ['mortgage_status'])
.set_custom_predict_fn(model.predict_proba)
.set_target_feature('mortgage_status')
.set_label_vocab(['denied', 'approved']))
WitWidget(config_builder, height=800)
```

Untitled.ipynb

```
[11]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bytree=1,
colsample_bynode=1, colsample_bylevel=1, gamma=0, gpu_id=-1,
importance_type='gain', interaction_constraints='',
learning_rate=0.300000012, max_delta_step=0, max_depth=6,
min_child_weight=1, missing=nan, monotone_constraints=''),
n_estimators=100, n_jobs=0, num_parallel_tree=1,
objective='reg:logistic', random_state=0, reg_alpha=0,
reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='exact', validate_parameters=1, verbosity=None)

[12]: y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')
0.874616

[13]: model.save_model('model.bst')

[14]: num_wit_examples = 500
test_examples = np.hstack((x_test[:num_wit_examples],y_test[:num_wit_examples].reshape(-1,1)))

[16]: config_builder = (WitConfigBuilder(test_examples.tolist(), data.columns.tolist() + ['mortgage_status'])
.set_custom_predict_fn(model.predict_proba)
.set_target_feature('mortgage_status')
.set_label_vocab(['denied', 'approved']))
WitWidget(config_builder, height=800)
```

Datapoint editor Performance & Fairness Features No datapoints loaded yet

Visualize Binning I ... Binning I ... Color By Label By Scatter I ... Scatter I ...
 Datapoints Partial dependence plots Nearest counterfactual

Datapoints and their inference results will be displayed here.

I) https://codelabs.developers.google.com/vertex_notebook_executor#0

```
[1]: import tensorflow as tf
import tensorflow_datasets as tfds
import tensorflow_hub as hub

[ ]: data, info = tfds.load(name='deep_weeds', as_supervised=True, with_info=True)
NUM_CLASSES = info.features['label'].num_classes
DATASET_SIZE = info.splits['train'].num_examples

[ ]: def preprocess_data(image, label):
    image = tf.image.resize(image, (300,300))
    return tf.cast(image, tf.float32) / 255., label

[ ]: # Create train/validation splits

# Shuffle dataset
dataset = data['train'].shuffle(1000)

train_split = 0.8
val_split = 0.2
train_size = int(train_split * DATASET_SIZE)
val_size = int(val_split * DATASET_SIZE)

train_data = dataset.take(train_size)
train_data = train_data.map(preprocess_data)
train_data = train_data.batch(64)

validation_data = dataset.skip(train_size)
validation_data = validation_data.map(preprocess_data)
validation_data = validation_data.batch(64)

[ ]: feature_extractor_model = "inception_v3"

[ ]: tf_hub_uri = f"https://tfhub.dev/google/imagenet/{feature_extractor_model}/feature_vector/5"
```

```
[ ]: # Shuffle dataset
dataset = data['train'].shuffle(1000)

train_split = 0.8
val_split = 0.2
train_size = int(train_split * DATASET_SIZE)
val_size = int(val_split * DATASET_SIZE)

train_data = dataset.take(train_size)
train_data = train_data.map(preprocess_data)
train_data = train_data.batch(64)

validation_data = dataset.skip(train_size)
validation_data = validation_data.map(preprocess_data)
validation_data = validation_data.batch(64)

[ ]: feature_extractor_model = "inception_v3"

[ ]: tf_hub_uri = f"https://tfhub.dev/google/imagenet/{feature_extractor_model}/feature_vector/5"

[ ]: feature_extractor_layer = hub.KerasLayer(
    tf_hub_uri,
    trainable=False)

[ ]: model = tf.keras.Sequential([
    feature_extractor_layer,
    tf.keras.layers.Dense(units=NUM_CLASSES)
])

[ ]: model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['acc'])

model.fit(train_data, validation_data=validation_data, epochs=20)
```

m)<https://codelabs.developers.google.com/codelabs/time-series-forecasting-with-cloud-ai-platform#0>

The screenshot shows a Jupyter Notebook interface with a file browser on the left and a terminal window on the right.

File Browser:

- Address bar: 7d439f2d969082ea-dot-us-west1.notebooks.googleusercontent.com/lab
- File menu: File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help
- Toolbar: Back, Forward, Refresh, New, Save, Run, Kernel, Git, Tabs, Settings, Help
- File list:
 - /
 - Name Last Modified
 - src 26 minutes ago
 - training-data-analyst seconds ago
 - tutorials 26 minutes ago
- Search bar: Filter files by name

Terminal Window:

```
(base) jupyter@tensorflow-2-3-20211203-133926:~$ git clone https://github.com/GoogleCloudPlatform/training-data-analyst
Cloning into 'training-data-analyst'...
remote: Enumerating objects: 54506, done.
remote: Counting objects: 100% (375/375), done.
remote: Compressing objects: 100% (246/246), done.
remote: Total 54506 (delta 147), reused 266 (delta 83), pack-reused 54131
Receiving objects: 100% (54506/54506), 627.40 MiB | 22.17 MiB/s, done.
Resolving deltas: 100% (34518/34518), done.
Checking out files: 100% (12334/12334), done.
(base) jupyter@tensorflow-2-3-20211203-133926:~$
```

File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 01-explore.ipynb Python 3

```
Restarting the kernel may be required to use new packages.

[1]: %pip install -U statsmodels scikit-learn --user

Requirement already satisfied: statsmodels in /opt/conda/lib/python3.7/site-packages (0.13.1)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-packages (1.0.1)
Requirement already satisfied: numpy>=1.17 in /opt/conda/lib/python3.7/site-packages (from statsmodels) (1.19.5)
Requirement already satisfied: pandas>=0.25 in /opt/conda/lib/python3.7/site-packages (from statsmodels) (1.3.4)
Requirement already satisfied: scipy>=1.3 in /opt/conda/lib/python3.7/site-packages (from statsmodels) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn) (3.0.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.25>statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.25>statsmodels) (2.021.3)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (from patsy>=0.5.2>statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.

Note: To restart the Kernel, navigate to Kernel > Restart Kernel... on the Jupyter menu.
```

Import libraries and define constants

```
[3]: from pandas.plotting import register_matplotlib_converters
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import grangercausalitytests

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

[5]: # Enter your project and region. Then run the cell to make sure the
# Cloud SDK uses the right project for all the commands in this notebook.
```

File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 01-explore.ipynb Python 3

```
[5]: # Enter your project and region. Then run the cell to make sure the
# Cloud SDK uses the right project for all the commands in this notebook.

PROJECT = 'Vertex-AI' # REPLACE WITH YOUR PROJECT NAME
REGION = 'us-central-1' # REPLACE WITH YOUR REGION e.g. us-central1

#Don't change the following command - this is to check if you have changed the project name above.
#assert PROJECT != 'Vertex-AI', 'Don\'t forget to change the project variables!'

[7]: target = 'total_rides' # The variable you are predicting
target_description = 'Total Rides' # A description of the target variable
features = {'day_type': 'Day Type'} # Weekday = W, Saturday = A, Sunday/Holiday = U
ts_col = 'service_date' # The name of the column with the date field

raw_data_file = 'https://data.cityofchicago.org/api/views/6ily-9597/rows.csv?accessType=DOWNLOAD'
processed_file = 'cta_ridership.csv' # Which file to save the results to
```

Load data

```
[8]: # Import CSV file
df = pd.read_csv(raw_data_file, index_col=[ts_col], parse_dates=[ts_col])

[9]: # Model data prior to 2020
df = df[df.index < '2020-01-01']

[10]: # Drop duplicates
df = df.drop_duplicates()

[11]: # Sort by date
```

File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 01-explore.ipynb

Explore data

```
[12]: # Print the top 5 rows
df.head()
```

	day_type	bus	rail_boardings	total_rides
2001-01-01	U	297192	126455	423647
2001-01-02	W	780827	501952	1282779
2001-01-03	W	824923	536432	1361355
2001-01-04	W	870021	550011	1420032
2001-01-05	W	890426	557917	1448343

TODO 1: Analyze the patterns

- Is ridership changing much over time?
- Is there a difference in ridership between the weekday and weekends?
- Is the mix of bus vs rail ridership changing over time?

```
[13]: # Initialize plotting
register_matplotlib_converters() # Addresses a warning
sns.set(rc={'figure.figsize':(16,4)})
```

```
[14]: # Explore total rides over time
sns.lineplot(data=df, x=df.index, y=df[target]).set_title('Total Rides')
fig = plt.show()
```



