Structured data prediction using Cloud AI Platform

This notebook illustrates:

1. Create a BigQuery Dataset and Google Cloud Storage Bucket
2. Export from BigQuery to CSVs in GCS
3. Training on Cloud AI Platform
4. Deploy trained model

In [1]:
```
!sudo chown -R jupyter:jupyter /home/jupyter/training-data-analyst
```

In [2]:
```
!pip install --user google-cloud-bigquery==1.25.0
```

```
Collecting google-cloud-bigquery==1.25.0
  Downloading google_cloud_bigquery-1.25.0-py2.py3-none-any.whl (169 kB)
     |████████████████████████████████| 169 kB 7.4 MB/s eta 0:00:01
Requirement already satisfied: google-auth<2.0dev,>=1.9.0 in /opt/conda/lib/pyth
on3.7/site-packages (from google-cloud-bigquery==1.25.0) (1.35.0)
Requirement already satisfied: six<2.0.0dev,>=1.13.0 in /opt/conda/lib/python3.
7/site-packages (from google-cloud-bigquery==1.25.0) (1.16.0)
Collecting google-resumable-media<0.6dev,>=0.5.0
  Downloading google_resumable_media-0.5.1-py2.py3-none-any.whl (38 kB)
Collecting google-cloud-core<2.0dev,>=1.1.0
  Downloading google_cloud_core-1.7.2-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: google-api-core<2.0dev,>=1.15.0 in /opt/conda/li
b/python3.7/site-packages (from google-cloud-bigquery==1.25.0) (1.31.2)
Requirement already satisfied: protobuf>=3.6.0 in /opt/conda/lib/python3.7/site-
packages (from google-cloud-bigquery==1.25.0) (3.16.0)
Requirement already satisfied: packaging>=14.3 in /opt/conda/lib/python3.7/site-
packages (from google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery==1.25.0)
(21.0)
Requirement already satisfied: setuptools>=40.3.0 in /opt/conda/lib/python3.7/si
te-packages (from google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery==1.25.
0) (57.4.0)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in /opt/c
onda/lib/python3.7/site-packages (from google-api-core<2.0dev,>=1.15.0->google-c
loud-bigquery==1.25.0) (1.53.0)
Requirement already satisfied: pytz in /opt/conda/lib/python3.7/site-packages (f
rom google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery==1.25.0) (2021.1)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /opt/conda/lib/pyth
on3.7/site-packages (from google-api-core<2.0dev,>=1.15.0->google-cloud-bigquery
==1.25.0) (2.25.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.7/site-pa
ckages (from google-auth<2.0dev,>=1.9.0->google-cloud-bigquery==1.25.0) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /opt/conda/lib/python3.
7/site-packages (from google-auth<2.0dev,>=1.9.0->google-cloud-bigquery==1.25.0)
(4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.
7/site-packages (from google-auth<2.0dev,>=1.9.0->google-cloud-bigquery==1.25.0)
(0.2.7)
Requirement already satisfied: pyparsing>=2.0.2 in /opt/conda/lib/python3.7/site
-packages (from packaging>=14.3->google-api-core<2.0dev,>=1.15.0->google-cloud-b
```

```
igquery==1.25.0) (2.4.7)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /opt/conda/lib/python3.7/
site-packages (from pyasn1-modules>=0.2.1->google-auth<2.0dev,>=1.9.0->google-cl
oud-bigquery==1.25.0) (0.4.8)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-pac
kages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.0->google-
cloud-bigquery==1.25.0) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/si
te-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.0->g
oogle-cloud-bigquery==1.25.0) (2021.5.30)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.
7/site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.
0->google-cloud-bigquery==1.25.0) (1.26.6)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.7/sit
e-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2.0dev,>=1.15.0->go
ogle-cloud-bigquery==1.25.0) (4.0.0)
Installing collected packages: google-resumable-media, google-cloud-core, google
-cloud-bigquery
ERROR: pip's dependency resolver does not currently take into account all the pa
ckages that are installed. This behaviour is the source of the following depende
ncy conflicts.
google-cloud-storage 1.42.0 requires google-resumable-media<3.0dev,>=1.3.0; pyth
on_version >= "3.6", but you have google-resumable-media 0.5.1 which is incompat
ible.
Successfully installed google-cloud-bigquery-1.25.0 google-cloud-core-1.7.2 goog
le-resumable-media-0.5.1
```

Note: Restart your kernel to use updated packages.

Kindly ignore the deprecation warnings and incompatibility errors related to google-cloud-storage.

Set up environment variables and load necessary libraries

Set environment variables so that we can use them throughout the entire notebook. We will be using our project name for our bucket, so you only need to change your project and region.

In [3]:
```
# change these to try this notebook out
BUCKET = 'qwiklabs-gcp-01-44a3211a105a' # Replace with the your bucket name
PROJECT = 'qwiklabs-gcp-01-44a3211a105a' # Replace with your project-id
REGION = 'us-central1'
```

In [4]:
```
import os

from google.cloud import bigquery
```

In [5]:
```
os.environ["PROJECT"] = PROJECT
os.environ["BUCKET"] = BUCKET
os.environ["REGION"] = REGION
os.environ["TFVERSION"] = "2.3"
os.environ["PYTHONVERSION"] = "3.7"
```

In [6]:
```
%%bash
export PROJECT=$(gcloud config list project --format "value(core.project)")
echo "Your current GCP Project Name is: "$PROJECT
```

```
Your current GCP Project Name is: qwiklabs-gcp-01-44a3211a105a
```

The source dataset

Our dataset is hosted in BigQuery. The CDC's Natality data has details on US births from 1969 to 2008 and is a publically available dataset, meaning anyone with a GCP account has access. Click here to access the dataset.

The natality dataset is relatively large at almost 138 million rows and 31 columns, but simple to understand. `weight_pounds` is the target, the continuous value we'll train a model to predict.

Create a BigQuery Dataset and Google Cloud Storage Bucket

A BigQuery dataset is a container for tables, views, and models built with BigQuery ML. Let's create one called **babyweight**. We'll do the same for a GCS bucket for our project too.

In [7]:

```
%%bash

# Create a BigQuery dataset for babyweight if it doesn't exist
datasetexists=$(bq ls -d | grep -w babyweight)

if [ -n "$datasetexists" ]; then
    echo -e "BigQuery dataset already exists, let's not recreate it."

else
    echo "Creating BigQuery dataset titled: babyweight"

    bq --location=US mk --dataset \
        --description "Babyweight" \
        $PROJECT:babyweight
    echo "Here are your current datasets:"
    bq ls
fi

## Create GCS bucket if it doesn't exist already...
exists=$(gsutil ls -d | grep -w gs://${BUCKET}/)

if [ -n "$exists" ]; then
    echo -e "Bucket exists, let's not recreate it."

else
    echo "Creating a new GCS bucket."
    gsutil mb -l ${REGION} gs://${BUCKET}
    echo "Here are your current buckets:"
    gsutil ls
fi
```

```
Creating BigQuery dataset titled: babyweight
Dataset 'qwiklabs-gcp-01-44a3211a105a:babyweight' successfully created.
Here are your current datasets:
  datasetId
 ------------
```

```
  babyweight
Bucket exists, let's not recreate it.
```

Create the training and evaluation data tables

Since there is already a publicly available dataset, we can simply create the training and evaluation data tables using this raw input data. First we are going to create a subset of the data limiting our columns to `weight_pounds`, `is_male`, `mother_age`, `plurality`, and `gestation_weeks` as well as some simple filtering and a column to hash on for repeatable splitting.

- Note: The dataset in the create table code below is the one created previously, e.g. "babyweight".

Preprocess and filter dataset

We have some preprocessing and filtering we would like to do to get our data in the right format for training.

Preprocessing:

- Cast `is_male` from `BOOL` to `STRING`
- Cast `plurality` from `INTEGER` to `STRING` where `[1, 2, 3, 4, 5]` becomes `["Single(1)", "Twins(2)", "Triplets(3)", "Quadruplets(4)", "Quintuplets(5)"]`
- Add `hashcolumn` hashing on `year` and `month`

Filtering:

- Only want data for years later than `2000`
- Only want baby weights greater than `0`
- Only want mothers whose age is greater than `0`
- Only want plurality to be greater than `0`
- Only want the number of weeks of gestation to be greater than `0`

In [8]:

```
%%bigquery
CREATE OR REPLACE TABLE
    babyweight.babyweight_data AS
SELECT
    weight_pounds,
    CAST(is_male AS STRING) AS is_male,
    mother_age,
    CASE
        WHEN plurality = 1 THEN "Single(1)"
        WHEN plurality = 2 THEN "Twins(2)"
        WHEN plurality = 3 THEN "Triplets(3)"
        WHEN plurality = 4 THEN "Quadruplets(4)"
        WHEN plurality = 5 THEN "Quintuplets(5)"
    END AS plurality,
    gestation_weeks,
    FARM_FINGERPRINT(
        CONCAT(
```

```
            CAST(year AS STRING),
            CAST(month AS STRING)
        )
    ) AS hashmonth
FROM
    publicdata.samples.natality
WHERE
    year > 2000
    AND weight_pounds > 0
    AND mother_age > 0
    AND plurality > 0
    AND gestation_weeks > 0
```

```
Query complete after 0.00s: 100%|██████████| 3/3 [00:00<00:00, 1822.55query/s]
```

Out[8]: —

Augment dataset to simulate missing data

Now we want to augment our dataset with our simulated babyweight data by setting all gender information to `Unknown` and setting plurality of all non-single births to `Multiple(2+)` .

In [9]:

```
%%bigquery
CREATE OR REPLACE TABLE
    babyweight.babyweight_augmented_data AS
SELECT
    weight_pounds,
    is_male,
    mother_age,
    plurality,
    gestation_weeks,
    hashmonth
FROM
    babyweight.babyweight_data
UNION ALL
SELECT
    weight_pounds,
    "Unknown" AS is_male,
    mother_age,
    CASE
        WHEN plurality = "Single(1)" THEN plurality
        ELSE "Multiple(2+)"
    END AS plurality,
    gestation_weeks,
    hashmonth
FROM
    babyweight.babyweight_data
```

```
Query complete after 0.01s: 100%|██████████| 3/3 [00:00<00:00, 320.25query/s]
```

Out[9]: —

Split augmented dataset into train and eval sets

Using `hashmonth` , apply a module to get approximately a 75/25 train-eval split.

Split augmented dataset into train dataset

In [10]:
```
%%bigquery
CREATE OR REPLACE TABLE
    babyweight.babyweight_data_train AS
SELECT
    weight_pounds,
    is_male,
    mother_age,
    plurality,
    gestation_weeks
FROM
    babyweight.babyweight_augmented_data
WHERE
    ABS(MOD(hashmonth, 4)) < 3
```

```
Query complete after 0.00s: 100%|██████████| 3/3 [00:00<00:00, 1651.73query/s]
```

Out[10]: —

Split augmented dataset into eval dataset

In [11]:
```
%%bigquery
CREATE OR REPLACE TABLE
    babyweight.babyweight_data_eval AS
SELECT
    weight_pounds,
    is_male,
    mother_age,
    plurality,
    gestation_weeks
FROM
    babyweight.babyweight_augmented_data
WHERE
    ABS(MOD(hashmonth, 4)) = 3
```

```
Query complete after 0.00s: 100%|██████████| 3/3 [00:00<00:00, 1622.55query/s]
```

Out[11]: —

Verify table creation

Verify that you created the dataset and training data table.

In [12]:
```
%%bigquery
-- LIMIT 0 is a free query; this allows us to check that the table exists.
SELECT * FROM babyweight.babyweight_data_train
LIMIT 0
```

```
Query complete after 0.00s: 100%|██████████| 1/1 [00:00<00:00, 555.61query/s]
Downloading: 0rows [00:00, ?rows/s]
```

Out[12]:

weight_pounds	is_male	mother_age	plurality	gestation_weeks

In [13]:
```
%%bigquery
-- LIMIT 0 is a free query; this allows us to check that the table exists.
SELECT * FROM babyweight.babyweight_data_eval
LIMIT 0
```

```
Query complete after 0.00s: 100%|██████████| 1/1 [00:00<00:00, 446.30query/s]
Downloading: 0rows [00:00, ?rows/s]
```

Out[13]:

	weight_pounds	is_male	mother_age	plurality	gestation_weeks

Export from BigQuery to CSVs in GCS

Use BigQuery Python API to export our train and eval tables to Google Cloud Storage in the CSV format to be used later for TensorFlow/Keras training. We'll want to use the dataset we've been using above as well as repeat the process for both training and evaluation data.

In [14]:

```
# Construct a BigQuery client object.
client = bigquery.Client()

dataset_name = "babyweight"

# Create dataset reference object
dataset_ref = client.dataset(
    dataset_id=dataset_name, project=client.project)

# Export both train and eval tables
for step in ["train", "eval"]:
    destination_uri = os.path.join(
        "gs://", BUCKET, dataset_name, "data", "{}*.csv".format(step))
    table_name = "babyweight_data_{}".format(step)
    table_ref = dataset_ref.table(table_name)
    extract_job = client.extract_table(
        table_ref,
        destination_uri,
        # Location must match that of the source table.
        location="US",
    )  # API request
    extract_job.result()  # Waits for job to complete.

    print("Exported {}:{}.{} to {}".format(
        client.project, dataset_name, table_name, destination_uri))
```

```
Exported qwiklabs-gcp-01-44a3211a105a:babyweight.babyweight_data_train to gs://q
wiklabs-gcp-01-44a3211a105a/babyweight/data/train*.csv
Exported qwiklabs-gcp-01-44a3211a105a:babyweight.babyweight_data_eval to gs://qw
iklabs-gcp-01-44a3211a105a/babyweight/data/eval*.csv
```

Verify CSV creation

Verify that we correctly created the CSV files in our bucket.

In [15]:

```
%%bash
gsutil ls gs://${BUCKET}/babyweight/data/*.csv
```

```
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/eval000000000000.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/eval000000000001.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000000.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000001.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000002.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000003.csv
```

```
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000004.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000005.csv
```

Check data exists

Verify that you previously created CSV files we'll be using for training and evaluation.

In [16]:

```
%%bash
gsutil ls gs://${BUCKET}/babyweight/data/*000000000000.csv
```

```
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/eval000000000000.csv
gs://qwiklabs-gcp-01-44a3211a105a/babyweight/data/train000000000000.csv
```

Training on Cloud AI Platform

Now that we see everything is working locally, it's time to train on the cloud!

To submit to the Cloud we use `gcloud ai-platform jobs submit training [jobname]` and simply specify some additional parameters for AI Platform Training Service:

- jobname: A unique identifier for the Cloud job. We usually append system time to ensure uniqueness
- job-dir: A GCS location to upload the Python package to
- runtime-version: Version of TF to use.
- python-version: Version of Python to use. Currently only Python 3.7 is supported for TF 2.3.
- region: Cloud region to train in. See here for supported AI Platform Training Service regions

Below the `-- \` we add in the arguments for our `task.py` file.

In [17]:

```
%%bash

OUTDIR=gs://${BUCKET}/babyweight/trained_model
JOBID=babyweight_$(date -u +%y%m%d_%H%M%S)

gcloud ai-platform jobs submit training ${JOBID} \
    --region=${REGION} \
    --module-name=trainer.task \
    --package-path=$(pwd)/babyweight/trainer \
    --job-dir=${OUTDIR} \
    --staging-bucket=gs://${BUCKET} \
    --master-machine-type=n1-standard-8 \
    --scale-tier=CUSTOM \
    --runtime-version=${TFVERSION} \
    --python-version=${PYTHONVERSION} \
    -- \
    --train_data_path=gs://${BUCKET}/babyweight/data/train*.csv \
    --eval_data_path=gs://${BUCKET}/babyweight/data/eval*.csv \
    --output_dir=${OUTDIR} \
    --num_epochs=10 \
    --train_examples=10000 \
    --eval_steps=100 \
    --batch_size=32 \
    --nembeds=8
```

```
ERROR: (gcloud.ai-platform.jobs.submit.training) FAILED_PRECONDITION: Field: job
_dir Error: The provided GCS path gs://qwiklabs-gcp-01-44a3211a105a/babyweight/t
rained_model cannot be written by service account service-204959009116@cloud-ml.
google.com.iam.gserviceaccount.com.
- '@type': type.googleapis.com/google.rpc.BadRequest
  fieldViolations:
  - description: The provided GCS path gs://qwiklabs-gcp-01-44a3211a105a/babywei
ght/trained_model
      cannot be written by service account service-204959009116@cloud-ml.google.
com.iam.gserviceaccount.com.
    field: job_dir

---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
/tmp/ipykernel_19723/2586862429.py in <module>
----> 1 get_ipython().run_cell_magic('bash', '', '\nOUTDIR=gs://${BUCKET}/babywe
ight/trained_model\nJOBID=babyweight_$(date -u +%y%m%d_%H%M%S)\n\ngcloud ai-plat
form jobs submit training ${JOBID} \\\n    --region=${REGION} \\\n    --module-n
ame=trainer.task \\\n    --package-path=$(pwd)/babyweight/trainer \\\n    --job-
dir=${OUTDIR} \\\n    --staging-bucket=gs://${BUCKET} \\\n    --master-machine-t
ype=n1-standard-8 \\\n    --scale-tier=CUSTOM \\\n    --runtime-version=${TFVERS
ION} \\\n    --python-version=${PYTHONVERSION} \\\n    -- \\\n    --train_data_p
ath=gs://${BUCKET}/babyweight/data/train*.csv \\\n    --eval_data_path=gs://${BU
CKET}/babyweight/data/eval*.csv \\\n    --output_dir=${OUTDIR} \\\n    --num_epo
chs=10 \\\n    --train_examples=10000 \\\n    --eval_steps=100 \\\n    --batch_s
ize=32 \\\n    --nembeds=8\n')

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_c
ell_magic(self, magic_name, line, cell)
   2401             with self.builtin_trap:
   2402                 args = (magic_arg_s, cell)
-> 2403                 result = fn(*args, **kwargs)
   2404             return result
   2405

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in named_sc
ript_magic(line, cell)
    140             else:
    141                 line = script
--> 142             return self.shebang(line, cell)
    143
    144         # write a basic docstring:

/opt/conda/lib/python3.7/site-packages/decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a,
 **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in shebang
(self, line, cell)
    243                 sys.stderr.flush()
    244         if args.raise_error and p.returncode!=0:
```

```
--> 245             raise CalledProcessError(p.returncode, cell, output=out, std
err=err)
    246
    247     def _run_script(self, p, cell, to_close):

CalledProcessError: Command 'b'\nOUTDIR=gs://${BUCKET}/babyweight/trained_model
\nJOBID=babyweight_$(date -u +%y%m%d_%H%M%S)\n\ngcloud ai-platform jobs submit t
raining ${JOBID} \\\n    --region=${REGION} \\\n    --module-name=trainer.task
 \\\n    --package-path=$(pwd)/babyweight/trainer \\\n    --job-dir=${OUTDIR}
 \\\n    --staging-bucket=gs://${BUCKET} \\\n    --master-machine-type=n1-standa
rd-8 \\\n    --scale-tier=CUSTOM \\\n    --runtime-version=${TFVERSION} \\\n
 --python-version=${PYTHONVERSION} \\\n    -- \\\n    --train_data_path=gs://${B
UCKET}/babyweight/data/train*.csv \\\n    --eval_data_path=gs://${BUCKET}/babywe
ight/data/eval*.csv \\\n    --output_dir=${OUTDIR} \\\n    --num_epochs=10 \\\n
--train_examples=10000 \\\n    --eval_steps=100 \\\n    --batch_size=32 \\\n
 --nembeds=8\n'' returned non-zero exit status 1.
```

The training job should complete within 15 to 20 minutes. You do not need to wait for this training job to finish before moving forward in the notebook, but will need a trained model.

Check our trained model files

Let's check the directory structure of our outputs of our trained model in folder we exported. We'll want to deploy the saved_model.pb within the timestamped directory as well as the variable values in the variables folder. Therefore, we need the path of the timestamped directory so that everything within it can be found by Cloud AI Platform's model deployment service.

In [18]:

```
%%bash
gsutil ls gs://${BUCKET}/babyweight/trained_model
```

```
CommandException: One or more URLs matched no objects.
```

```
---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
/tmp/ipykernel_19723/3083723745.py in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'gsutil ls gs://${BUCKET}/babyw
eight/trained_model\n')

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_c
ell_magic(self, magic_name, line, cell)
   2401             with self.builtin_trap:
   2402                 args = (magic_arg_s, cell)
-> 2403                 result = fn(*args, **kwargs)
   2404             return result
   2405

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in named_sc
ript_magic(line, cell)
    140             else:
    141                 line = script
--> 142             return self.shebang(line, cell)
    143
    144         # write a basic docstring:

/opt/conda/lib/python3.7/site-packages/decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
```

```
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a,
 **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in shebang
(self, line, cell)
    243             sys.stderr.flush()
    244         if args.raise_error and p.returncode!=0:
--> 245             raise CalledProcessError(p.returncode, cell, output=out, std
err=err)
    246
    247     def _run_script(self, p, cell, to_close):

CalledProcessError: Command 'b'gsutil ls gs://${BUCKET}/babyweight/trained_model
\n'' returned non-zero exit status 1.
```

In [19]:

```
%%bash
MODEL_LOCATION=$(gsutil ls -ld -- gs://${BUCKET}/babyweight/trained_model/2* \
                 | tail -1)
gsutil ls ${MODEL_LOCATION}
```

```
gs://qwiklabs-gcp-01-44a3211a105a/
CommandException: One or more URLs matched no objects.
```

Deploy trained model

Deploying the trained model to act as a REST web service is a simple gcloud call.

In [20]:

```
%%bash
gcloud config set ai_platform/region global
```

```
Updated property [ai_platform/region].
```

In [21]:

```
%%bash
MODEL_NAME="babyweight"
MODEL_VERSION="ml_on_gcp"
MODEL_LOCATION=$(gsutil ls -ld -- gs://${BUCKET}/babyweight/trained_model/2* \
                 | tail -1 | tr -d '[:space:]')
echo "Deleting and deploying $MODEL_NAME $MODEL_VERSION from $MODEL_LOCATION"
# gcloud ai-platform versions delete ${MODEL_VERSION} --model ${MODEL_NAME}
# gcloud ai-platform models delete ${MODEL_NAME}
gcloud ai-platform models create ${MODEL_NAME} --regions ${REGION}
gcloud ai-platform versions create ${MODEL_VERSION} \
    --model=${MODEL_NAME} \
    --origin=${MODEL_LOCATION} \
    --runtime-version=2.3 \
    --python-version=3.7
```

```
Deleting and deploying babyweight ml_on_gcp from
```

```
CommandException: One or more URLs matched no objects.
Using endpoint [https://ml.googleapis.com/]
Created ai platform model [projects/qwiklabs-gcp-01-44a3211a105a/models/babyweig
ht].
Using endpoint [https://ml.googleapis.com/]
ERROR: (gcloud.ai-platform.versions.create) Either `--origin` must be provided o
r `deploymentUri` must be provided in the file given by `--config`.
```

```
---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
/tmp/ipykernel_19723/281812598.py in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'MODEL_NAME="babyweight"\nMODEL
_VERSION="ml_on_gcp"\nMODEL_LOCATION=$(gsutil ls -ld -- gs://${BUCKET}/babyweigh
t/trained_model/2* \\\n                 | tail -1 | tr -d \'[:space:]\')\necho
 "Deleting and deploying $MODEL_NAME $MODEL_VERSION from $MODEL_LOCATION"\n# gcl
oud ai-platform versions delete ${MODEL_VERSION} --model ${MODEL_NAME}\n# gcloud
ai-platform models delete ${MODEL_NAME}\ngcloud ai-platform models create ${MODE
L_NAME} --regions ${REGION}\ngcloud ai-platform versions create ${MODEL_VERSION}
\\\n    --model=${MODEL_NAME} \\\n    --origin=${MODEL_LOCATION} \\\n    --runti
me-version=2.3 \\\n    --python-version=3.7\n')

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_c
ell_magic(self, magic_name, line, cell)
   2401             with self.builtin_trap:
   2402                 args = (magic_arg_s, cell)
-> 2403                 result = fn(*args, **kwargs)
   2404             return result
   2405

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in named_sc
ript_magic(line, cell)
    140             else:
    141                 line = script
--> 142             return self.shebang(line, cell)
    143
    144         # write a basic docstring:

/opt/conda/lib/python3.7/site-packages/decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a,
 **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in shebang
(self, line, cell)
    243             sys.stderr.flush()
    244         if args.raise_error and p.returncode!=0:
--> 245             raise CalledProcessError(p.returncode, cell, output=out, std
err=err)
    246
    247     def _run_script(self, p, cell, to_close):

CalledProcessError: Command 'b'MODEL_NAME="babyweight"\nMODEL_VERSION="ml_on_gc
```

```
p"\nMODEL_LOCATION=$(gsutil ls -ld -- gs://${BUCKET}/babyweight/trained_model/2*
\\\n                 | tail -1 | tr -d \'[:space:]\')\necho "Deleting and deploy
ing $MODEL_NAME $MODEL_VERSION from $MODEL_LOCATION"\n# gcloud ai-platform versi
ons delete ${MODEL_VERSION} --model ${MODEL_NAME}\n# gcloud ai-platform models d
elete ${MODEL_NAME}\ngcloud ai-platform models create ${MODEL_NAME} --regions
 ${REGION}\ngcloud ai-platform versions create ${MODEL_VERSION} \\\n    --model=
${MODEL_NAME} \\\n    --origin=${MODEL_LOCATION} \\\n    --runtime-version=2.3
 \\\n    --python-version=3.7\n'' returned non-zero exit status 1.
```