# Create Keras DNN model

This notebook illustrates:

1. Creating a model using Keras. This requires TensorFlow 2.1

In [1]:
```
# Ensure the right version of Tensorflow is installed.
!pip freeze | grep tensorflow==2.1
```

In [2]:
```
# change these to try this notebook out
BUCKET = 'cloud-training-demos-ml'
PROJECT = 'cloud-training-demos'
REGION = 'us-east1' #'us-central1'
```

In [3]:
```
import os
os.environ['BUCKET'] = BUCKET
os.environ['PROJECT'] = PROJECT
os.environ['REGION'] = REGION
```

In [4]:
```
%%bash
if ! gsutil ls | grep -q gs://${BUCKET}/; then
  gsutil mb -l ${REGION} gs://${BUCKET}
fi
```

```
Creating gs://cloud-training-demos-ml/...
ServiceException: 409 A Cloud Storage bucket named 'cloud-training-demos-ml' alr
eady exists. Try another name. Bucket names must be globally unique across all G
oogle Cloud projects, including those outside of your organization.
---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
<ipython-input-4-6b1d45d375e6> in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'if ! gsutil ls | grep -q gs://
${BUCKET}/; then\n  gsutil mb -l ${REGION} gs://${BUCKET}\nfi\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'if ! gsutil ls | grep -q gs://${BUCKET}/; then\n
gsutil mb -l ${REGION} gs://${BUCKET}\nfi\n'' returned non-zero exit status 1.
```

In [ ]:

```
%%bash
ls *.csv
```

# Create Keras model

First, write an input_fn to read the data.

In [5]:

```
import shutil
import numpy as np
import tensorflow as tf
print(tf.__version__)
```

```
2.3.3
```

In [6]:

```
# Determine CSV, label, and key columns
CSV_COLUMNS = 'weight_pounds,is_male,mother_age,plurality,gestation_weeks,key'.s
LABEL_COLUMN = 'weight_pounds'
KEY_COLUMN = 'key'

# Set default values for each CSV column. Treat is_male and plurality as strings
DEFAULTS = [[0.0], ['null'], [0.0], ['null'], [0.0], ['nokey']]
```

In [7]:

```
def features_and_labels(row_data):
    for unwanted_col in ['key']:
        row_data.pop(unwanted_col)
    label = row_data.pop(LABEL_COLUMN)
    return row_data, label  # features, label

# load the training data
def load_dataset(pattern, batch_size=1, mode=tf.estimator.ModeKeys.EVAL):
```

```
  dataset = (tf.data.experimental.make_csv_dataset(pattern, batch_size, CSV_COLU
             .map(features_and_labels) # features, label
             )
  if mode == tf.estimator.ModeKeys.TRAIN:
        dataset = dataset.shuffle(1000).repeat()
  dataset = dataset.prefetch(1) # take advantage of multi-threading; 1=AUTOTUNE
  return dataset
```

Next, define the feature columns. mother_age and gestation_weeks should be numeric. The others (is_male, plurality) should be categorical.

In [8]:

```
## Build a simple Keras DNN using its Functional API
def rmse(y_true, y_pred):
    return tf.sqrt(tf.reduce_mean(tf.square(y_pred - y_true)))

# Helper function to handle categorical columns
def categorical_fc(name, values):
  return tf.feature_column.indicator_column(
    tf.feature_column.categorical_column_with_vocabulary_list(name, values))

def build_dnn_model():
    # input layer
    inputs = {
        colname : tf.keras.layers.Input(name=colname, shape=(), dtype='float32')
           for colname in ['mother_age', 'gestation_weeks']
    }
    inputs.update({
        colname : tf.keras.layers.Input(name=colname, shape=(), dtype='string')
            for colname in ['is_male', 'plurality']
    })

    # feature columns from inputs
    feature_columns = {
        colname : tf.feature_column.numeric_column(colname)
            for colname in ['mother_age', 'gestation_weeks']
    }
    if False:
        # Until TF-serving supports 2.0, so as to get servable model
        feature_columns['is_male'] = categorical_fc('is_male', ['True', 'False',
        feature_columns['plurality'] = categorical_fc('plurality',
                          ['Single(1)', 'Twins(2)', 'Triplets(3)',
                           'Quadruplets(4)', 'Quintuplets(5)','Multiple(2+)'])

    # the constructor for DenseFeatures takes a list of numeric columns
    # The Functional API in Keras requires that you specify: LayerConstructor()(
    dnn_inputs = tf.keras.layers.DenseFeatures(feature_columns.values())(inputs)

    # two hidden layers of [64, 32] just in like the BQML DNN
    h1 = tf.keras.layers.Dense(64, activation='relu', name='h1')(dnn_inputs)
    h2 = tf.keras.layers.Dense(32, activation='relu', name='h2')(h1)

    # final output is a linear activation because this is regression
    output = tf.keras.layers.Dense(1, activation='linear', name='babyweight')(h2


    model = tf.keras.models.Model(inputs, output)
    model.compile(optimizer='adam', loss='mse', metrics=[rmse, 'mse'])
    return model
```

```
print("Here is our DNN architecture so far:\n")

# note how to use strategy to do distributed training
strategy = tf.distribute.MirroredStrategy()
with strategy.scope():
    model = build_dnn_model()
print(model.summary())
```

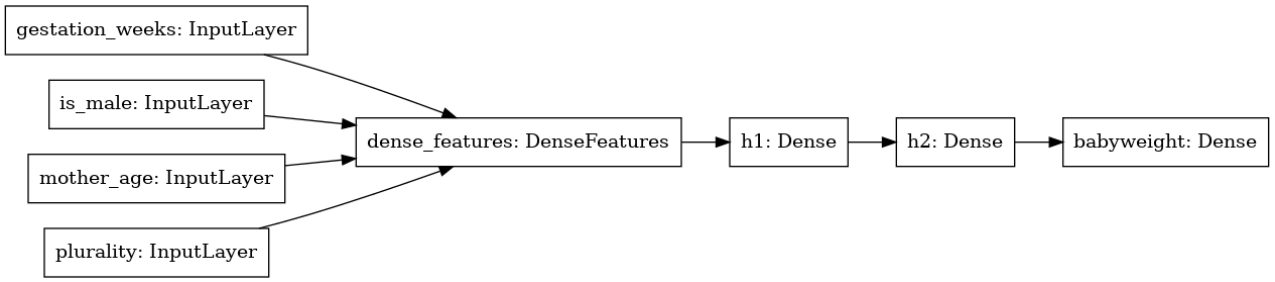
```
Here is our DNN architecture so far:

WARNING:tensorflow:There are non-GPU devices in `tf.distribute.Strategy`, not us
ing nccl allreduce.
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/t
ask:0/device:CPU:0',)
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
==================================================================
gestation_weeks (InputLayer) [(None,)]                 0
_________________________________________________________________
is_male (InputLayer)         [(None,)]                 0
_________________________________________________________________
mother_age (InputLayer)      [(None,)]                 0
_________________________________________________________________
plurality (InputLayer)       [(None,)]                 0
_________________________________________________________________
dense_features (DenseFeatures) (None, 2)               0         gestation_weeks
[0][0]
                                                                 is_male[0][0]
                                                                 mother_age[0]
[0]
                                                                 plurality[0][0]
_________________________________________________________________
h1 (Dense)                   (None, 64)                192       dense_features
[0][0]
_________________________________________________________________
h2 (Dense)                   (None, 32)                2080      h1[0][0]
_________________________________________________________________
babyweight (Dense)           (None, 1)                 33        h2[0][0]
==================================================================
Total params: 2,305
Trainable params: 2,305
Non-trainable params: 0
_________________________________________________________________
None
```

We can visualize the DNN using the Keras plot_model utility.

In [9]:

```
tf.keras.utils.plot_model(model, 'dnn_model.png', show_shapes=False, rankdir='LR
```

Out[9]:




# Train and evaluate

In [10]:

```
TRAIN_BATCH_SIZE = 32
NUM_TRAIN_EXAMPLES = 10000 * 5 # training dataset repeats, so it will wrap aroun
NUM_EVALS = 5  # how many times to evaluate
NUM_EVAL_EXAMPLES = 10000 # enough to get a reasonable sample, but not so much t

trainds = load_dataset('train*', TRAIN_BATCH_SIZE, tf.estimator.ModeKeys.TRAIN)
evalds = load_dataset('eval*', 1000, tf.estimator.ModeKeys.EVAL).take(NUM_EVAL_E

steps_per_epoch = NUM_TRAIN_EXAMPLES // (TRAIN_BATCH_SIZE * NUM_EVALS)

history = model.fit(trainds,
                    validation_data=evalds,
                    epochs=NUM_EVALS,
                    steps_per_epoch=steps_per_epoch)
```

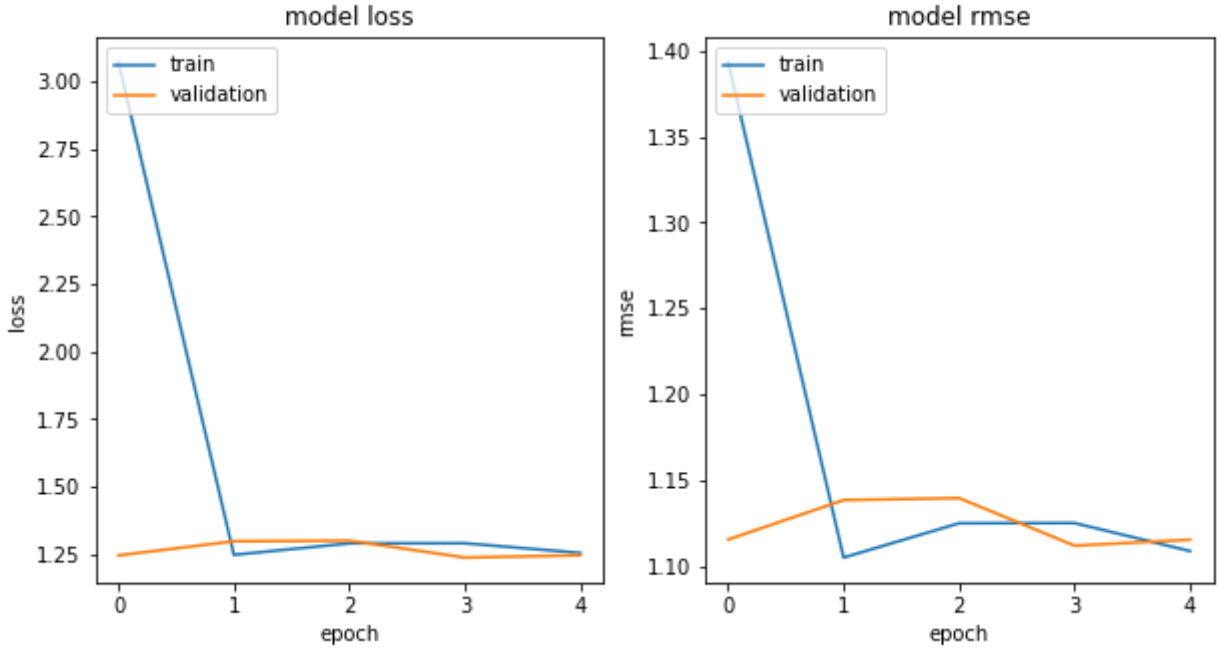
```
Epoch 1/5
312/312 [==============================] - 3s 11ms/step - loss: 3.0729 - rmse:
1.3935 - mse: 3.0729 - val_loss: 1.2441 - val_rmse: 1.1153 - val_mse: 1.2441
Epoch 2/5
312/312 [==============================] - 2s 7ms/step - loss: 1.2462 - rmse: 1.
1047 - mse: 1.2462 - val_loss: 1.2964 - val_rmse: 1.1382 - val_mse: 1.2964
Epoch 3/5
312/312 [==============================] - 2s 8ms/step - loss: 1.2887 - rmse: 1.
1247 - mse: 1.2887 - val_loss: 1.2986 - val_rmse: 1.1394 - val_mse: 1.2986
Epoch 4/5
312/312 [==============================] - 2s 8ms/step - loss: 1.2891 - rmse: 1.
1249 - mse: 1.2891 - val_loss: 1.2360 - val_rmse: 1.1117 - val_mse: 1.2360
Epoch 5/5
312/312 [==============================] - 2s 7ms/step - loss: 1.2533 - rmse: 1.
1085 - mse: 1.2533 - val_loss: 1.2445 - val_rmse: 1.1152 - val_mse: 1.2445
```

# Visualize loss curve

In [11]:

```
# plot
import matplotlib.pyplot as plt
nrows = 1
ncols = 2
fig = plt.figure(figsize=(10, 5))

for idx, key in enumerate(['loss', 'rmse']):
    ax = fig.add_subplot(nrows, ncols, idx+1)
    plt.plot(history.history[key])
    plt.plot(history.history['val_{}'.format(key)])
    plt.title('model {}'.format(key))
    plt.ylabel(key)
```

```
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left');
```




# Save the model

Let's wrap the model so that we can supply keyed predictions, and get the key back in our output

In [12]:
```
# Serving function that passes through keys
@tf.function(input_signature=[{
      'is_male': tf.TensorSpec([None,], dtype=tf.string, name='is_male'),
      'mother_age': tf.TensorSpec([None,], dtype=tf.float32, name='mother_age'),
      'plurality': tf.TensorSpec([None,], dtype=tf.string, name='plurality'),
      'gestation_weeks': tf.TensorSpec([None,], dtype=tf.float32, name='gestatio
      'key': tf.TensorSpec([None,], dtype=tf.string, name='key')
}])
def my_serve(inputs):
    feats = inputs.copy()
    key = feats.pop('key')
    output = model(feats)
    return {'key': key, 'babyweight': output}
```

In [13]:
```
import shutil, os, datetime
OUTPUT_DIR = './export/babyweight'
shutil.rmtree(OUTPUT_DIR, ignore_errors=True)
EXPORT_PATH = os.path.join(OUTPUT_DIR, datetime.datetime.now().strftime('%Y%m%d%
tf.saved_model.save(model, EXPORT_PATH, signatures={'serving_default': my_serve}
print("Exported trained model to {}".format(EXPORT_PATH))
os.environ['EXPORT_PATH'] = EXPORT_PATH
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow/pytho
n/training/tracking/tracking.py:111: Model.state_updates (from tensorflow.pytho
n.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied autom
```

```
atically.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow/pytho
n/training/tracking/tracking.py:111: Layer.updates (from tensorflow.python.kera
s.engine.base_layer) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied autom
atically.
INFO:tensorflow:Assets written to: ./export/babyweight/20210818182455/assets
Exported trained model to ./export/babyweight/20210818182455
```

In [14]:
```
!find $EXPORT_PATH
```

```
./export/babyweight/20210818182455
./export/babyweight/20210818182455/assets
./export/babyweight/20210818182455/variables
./export/babyweight/20210818182455/variables/variables.data-00000-of-00001
./export/babyweight/20210818182455/variables/variables.index
./export/babyweight/20210818182455/saved_model.pb
```

# Deploy trained model to Cloud AI Platform

In [15]:
```
!saved_model_cli show --tag_set serve --signature_def serving_default --dir {EXP
```

```
The given SavedModel SignatureDef contains the following input(s):
  inputs['gestation_weeks'] tensor_info:
      dtype: DT_FLOAT
      shape: (-1)
      name: serving_default_gestation_weeks:0
  inputs['is_male'] tensor_info:
      dtype: DT_STRING
      shape: (-1)
      name: serving_default_is_male:0
  inputs['key'] tensor_info:
      dtype: DT_STRING
      shape: (-1)
      name: serving_default_key:0
  inputs['mother_age'] tensor_info:
      dtype: DT_FLOAT
      shape: (-1)
      name: serving_default_mother_age:0
  inputs['plurality'] tensor_info:
      dtype: DT_STRING
      shape: (-1)
      name: serving_default_plurality:0
The given SavedModel SignatureDef contains the following output(s):
  outputs['babyweight'] tensor_info:
      dtype: DT_FLOAT
      shape: (-1, 1)
      name: StatefulPartitionedCall:0
  outputs['key'] tensor_info:
      dtype: DT_STRING
      shape: (-1)
      name: StatefulPartitionedCall:1
Method name is: tensorflow/serving/predict
```

In [16]:
```
%%bash
MODEL_NAME="babyweight"
VERSION_NAME="dnn"
```

```
MODEL_LOCATION=$EXPORT_PATH
echo "Deleting and deploying $MODEL_NAME $MODEL_VERSION from $MODEL_LOCATION ...

if [[ $(gcloud ai-platform models list --format='value(name)' | grep $MODEL_NAME
    echo "The model named $MODEL_NAME already exists."
else
    # create model
    echo "Creating $MODEL_NAME model now."
    gcloud ai-platform models create --regions=$REGION $MODEL_NAME
fi

if [[ $(gcloud ai-platform versions list --model $MODEL_NAME --format='value(nam
    echo "Deleting already the existing model $MODEL_NAME:$VERSION_NAME ... "
    gcloud ai-platform versions delete --model=$MODEL_NAME $VERSION_NAME
    echo "Please run this cell again if you don't see a Creating message ... "
    sleep 2
fi

# create model
echo "Creating $MODEL_NAME:$VERSION_NAME"
gcloud ai-platform versions create --model=$MODEL_NAME $VERSION_NAME --async \
       --framework=tensorflow --python-version=3.7 --runtime-version=2.1 \
       --origin=$MODEL_LOCATION --staging-bucket=gs://$BUCKET
```

```
Deleting and deploying babyweight  from ./export/babyweight/20210818182455 ... t
his will take a few minutes
Creating babyweight model now.
Using endpoint [https://us-central1-ml.googleapis.com/]
API [ml.googleapis.com] not enabled on project [571462072167]. Would
you like to enable and retry (this will take a few minutes)? (y/N)?
ERROR: (gcloud.ai-platform.models.list) User [571462072167-compute@developer.gse
rviceaccount.com] does not have permission to access projects instance [qwiklabs
-gcp-00-a849e5f2dbfd] (or it may not exist): AI Platform Training & Prediction A
PI has not been used in project 571462072167 before or it is disabled. Enable it
by visiting https://console.developers.google.com/apis/api/ml.googleapis.com/ove
rview?project=571462072167 then retry. If you enabled this API recently, wait a
few minutes for the action to propagate to our systems and retry.
- '@type': type.googleapis.com/google.rpc.Help
  links:
  - description: Google developers console API activation
    url: https://console.developers.google.com/apis/api/ml.googleapis.com/overvi
ew?project=571462072167
- '@type': type.googleapis.com/google.rpc.ErrorInfo
  domain: googleapis.com
  metadata:
    consumer: projects/571462072167
    service: ml.googleapis.com
  reason: SERVICE_DISABLED
Using endpoint [https://ml.googleapis.com/]
API [ml.googleapis.com] not enabled on project [571462072167]. Would
you like to enable and retry (this will take a few minutes)? (y/N)?
ERROR: (gcloud.ai-platform.models.create) User [571462072167-compute@developer.g
serviceaccount.com] does not have permission to access projects instance [qwikla
bs-gcp-00-a849e5f2dbfd] (or it may not exist): AI Platform Training & Prediction
API has not been used in project 571462072167 before or it is disabled. Enable i
t by visiting https://console.developers.google.com/apis/api/ml.googleapis.com/o
verview?project=571462072167 then retry. If you enabled this API recently, wait
a few minutes for the action to propagate to our systems and retry.
- '@type': type.googleapis.com/google.rpc.Help
  links:
```

```
  - description: Google developers console API activation
    url: https://console.developers.google.com/apis/api/ml.googleapis.com/overvi
ew?project=571462072167
- '@type': type.googleapis.com/google.rpc.ErrorInfo
  domain: googleapis.com
  metadata:
    consumer: projects/571462072167
    service: ml.googleapis.com
  reason: SERVICE_DISABLED

---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
<ipython-input-16-f05401309629> in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'MODEL_NAME="babyweight"\nVERSI
ON_NAME="dnn"\nMODEL_LOCATION=$EXPORT_PATH\necho "Deleting and deploying $MODEL_
NAME $MODEL_VERSION from $MODEL_LOCATION ... this will take a few minutes"\n\nif
[[ $(gcloud ai-platform models list --format=\'value(name)\' | grep $MODEL_NAME)
]]; then\n    echo "The model named $MODEL_NAME already exists."\nelse\n    # cr
eate model\n    echo "Creating $MODEL_NAME model now."\n    gcloud ai-platform m
odels create --regions=$REGION $MODEL_NAME\nfi\n\nif [[ $(gcloud ai-platform ver
sions list --model $MODEL_NAME --format=\'value(name)\' | grep $VERSION_NAME)
 ]]; then\n    echo "Deleting already the existing model $MODEL_NAME:$VERSION_NA
ME ... "\n    gcloud ai-platform versions delete --model=$MODEL_NAME $VERSION_NA
ME\n    echo "Please run this cell again if you don\'t see a Creating message
 ... "\n    sleep 2\nfi\n\n# create model\necho "Creating $MODEL_NAME:$VERSION_N
AME"\ngcloud ai-platform versions create --model=$MODEL_NAME $VERSION_NAME --asy
nc \\\n       --framework=tensorflow --python-version=3.7 --runtime-version=2.1
 \\\n       --origin=$MODEL_LOCATION --staging-bucket=gs://$BUCKET\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"\nVERSION_NAME="dnn"\nMODE
L_LOCATION=$EXPORT_PATH\necho "Deleting and deploying $MODEL_NAME $MODEL_VERSION
from $MODEL_LOCATION ... this will take a few minutes"\n\nif [[ $(gcloud ai-plat
form models list --format=\'value(name)\' | grep $MODEL_NAME) ]]; then\n    echo
"The model named $MODEL_NAME already exists."\nelse\n    # create model\n    ech
o "Creating $MODEL_NAME model now."\n    gcloud ai-platform models create --regi
ons=$REGION $MODEL_NAME\nfi\n\nif [[ $(gcloud ai-platform versions list --model
 $MODEL_NAME --format=\'value(name)\' | grep $VERSION_NAME) ]]; then\n    echo
 "Deleting already the existing model $MODEL_NAME:$VERSION_NAME ... "\n    gclou
d ai-platform versions delete --model=$MODEL_NAME $VERSION_NAME\n    echo "Pleas
e run this cell again if you don\'t see a Creating message ... "\n    sleep 2\nf
i\n\n# create model\necho "Creating $MODEL_NAME:$VERSION_NAME"\ngcloud ai-platfo
rm versions create --model=$MODEL_NAME $VERSION_NAME --async \\\n       --framew
ork=tensorflow --python-version=3.7 --runtime-version=2.1 \\\n       --origin=$M
ODEL_LOCATION --staging-bucket=gs://$BUCKET\n'' returned non-zero exit status 1.
```

Monitor the model creation at GCP Console > AI Platform and once the model version dnn is created, proceed to the next cell.

In [17]:

```
%%writefile input.json
{"key": "b1", "is_male": "True", "mother_age": 26.0, "plurality": "Single(1)", "
{"key": "b2", "is_male": "True", "mother_age": 33.0, "plurality": "Single(1)", "
{"key": "g1", "is_male": "False", "mother_age": 26.0, "plurality": "Single(1)",
{"key": "g2", "is_male": "False", "mother_age": 33.0, "plurality": "Single(1)",
```

```
Writing input.json
```

In [ ]:

```
!gcloud ai-platform predict --model babyweight --json-instances input.json --ver
```

```
Please specify a region:
(For the global endpoint the region needs to be specified as
'global'.)
 [1] global
 [2] asia-east1
 [3] asia-northeast1
 [4] asia-southeast1
 [5] australia-southeast1
 [6] europe-west1
 [7] europe-west2
 [8] europe-west3
 [9] europe-west4
 [10] northamerica-northeast1
 [11] us-central1
 [12] us-east1
 [13] us-east4
 [14] us-west1
 [15] cancel
Please enter your numeric choice:
```

# main.py

This is the code that exists in serving/application/main.py, i.e. the code in the web application that accesses the ML API.

In [ ]:

```
from oauth2client.client import GoogleCredentials
from googleapiclient import discovery

credentials = GoogleCredentials.get_application_default()
api = discovery.build('ml', 'v1', credentials=credentials)
project = PROJECT
model_name = 'babyweight'
version_name = 'dnn'

input_data = {
  'instances': [
    {
      'key': 'b1',
      'is_male': 'True',
      'mother_age': 26.0,
      'plurality': 'Single(1)',
      'gestation_weeks': 39
    },
    {
      'key': 'g1',
      'is_male': 'False',
      'mother_age': 29.0,
      'plurality': 'Single(1)',
      'gestation_weeks': 38
    },
    {
      'key': 'b2',
      'is_male': 'True',
      'mother_age': 26.0,
      'plurality': 'Triplets(3)',
      'gestation_weeks': 39
    },
    {
      'key': 'u1',
      'is_male': 'Unknown',
      'mother_age': 29.0,
      'plurality': 'Multiple(2+)',
      'gestation_weeks': 38
    },
  ]
}

parent = 'projects/%s/models/%s/versions/%s' % (project, model_name, version_nam
prediction = api.projects().predict(body=input_data, name=parent).execute()
print(prediction)
print(prediction['predictions'][0]['babyweight'][0])
```