Deploy and predict with Keras model on Cloud AI Platform.

Learning Objectives

1. Setup up the environment
2. Deploy trained Keras model to Cloud AI Platform
3. Online predict from model on Cloud AI Platform
4. Batch predict from model on Cloud AI Platform

Introduction

Verify that you have previously Trained your Keras model. If not, go back to train_keras_ai_platform_babyweight.ipynb create them. In this notebook, we'll be deploying our Keras model to Cloud AI Platform and creating predictions.

We will set up the environment, deploy a trained Keras model to Cloud AI Platform, online predict from deployed model on Cloud AI Platform, and batch predict from deployed model on Cloud AI Platform.

Each learning objective will correspond to a **#TODO** in this student lab notebook -- try to complete this notebook first and then review the solution notebook.

Set up environment variables and load necessary libraries

Import necessary libraries.

In [1]:
```
import os
```

Lab Task #1: Set environment variables.

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

In [2]:
```
%%bash
PROJECT=$(gcloud config list project --format "value(core.project)")
echo "Your current GCP Project Name is: "$qwiklabs-gcp-01-edaeb1f42464
```

```
Your current GCP Project Name is: -gcp-01-edaeb1f42464
```

In [3]:
```
# Change these to try this notebook out
PROJECT = "qwiklabs-gcp-01-edaeb1f42464"  # TODO 1: Replace with your PROJECT
BUCKET = PROJECT  # defaults to PROJECT
REGION = "us-central1"  # TODO 1: Replace with your REGION
```

In [4]:
```
os.environ["BUCKET"] = BUCKET
os.environ["REGION"] = REGION
os.environ["TFVERSION"] = "2.1"
```

In [5]:
```
%%bash
gcloud config set compute/region $REGION
gcloud config set ai_platform/region global
```

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

Check our trained model files

Let's check the directory structure of our outputs of our trained model in folder we exported the model to in our last lab. 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 [6]:
```
%%bash
gsutil ls gs://${BUCKET}/babyweight/trained_model
```

```
CommandException: One or more URLs matched no objects.
---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
<ipython-input-6-23f55e008635> 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 [7]:

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

```
gs://qwiklabs-gcp-01-edaeb1f42464/
CommandException: One or more URLs matched no objects.
```

Lab Task #2: Deploy trained model.

Deploying the trained model to act as a REST web service is a simple gcloud call. Complete **#TODO** by providing location of saved_model.pb file to Cloud AI Platform model deployment service. The deployment will take a few minutes.

In [ ]:

```
%%bash
MODEL_NAME="babyweight"
MODEL_VERSION="ml_on_gcp"
MODEL_LOCATION=(gsutil ls -ld -- gs://${BUCKET}/babyweight/trained_model/2* | ta
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=2.1 \
    --python-version=3.7
```

Lab Task #3: Use model to make online prediction.

Complete **#TODO**s for both the Python and gcloud Shell API methods of calling our deployed model on Cloud AI Platform for online prediction.

Python API

We can use the Python API to send a JSON request to the endpoint of the service to make it predict a baby's weight. The order of the responses are the order of the instances.

In [ ]:
```
from oauth2client.client import GoogleCredentials
import requests
import json

MODEL_NAME = "babyweight" # TODO 3a: Add model name
MODEL_VERSION = "ml_on_GCP"# TODO 3a: Add model version

token = GoogleCredentials.get_application_default().get_access_token().access_to
api = "https://ml.googleapis.com/v1/projects/{}/models/{}/versions/{}:predict" \
         .format(PROJECT, MODEL_NAME, MODEL_VERSION)
headers = {"Authorization": "Bearer " + token }
data = {
  "instances": [
    {
      "is_male": "True",
      "mother_age": 26.0,
      "plurality": "Single(1)",
      "gestation_weeks": 39
    },
    {
      "is_male": "False",
      "mother_age": 29.0,
      "plurality": "Single(1)",
      "gestation_weeks": 38
    },
    {
      "is_male": "True",
      "mother_age": 26.0,
      "plurality": "Triplets(3)",
      "gestation_weeks": 39
    },
    {
      "is_male": "False"
        "mother_age": 62.0
        "plurality": "Twins(2)"
        "gestation_weeks": 33
      }
    # TODO 3a: Create another instance
  ]
}
response = requests.post(api, json=data, headers=headers)
print(response.content)
```

The predictions for the four instances were: 5.33, 6.09, 2.50, and 5.86 pounds respectively when I ran it (your results might be different).

gcloud shell API

Instead we could use the gcloud shell API. Create a newline delimited JSON file with one instance per line and submit using gcloud.

In [ ]:
```
%%writefile inputs.json
{"is_male": "True", "mother_age": 26.0, "plurality": "Single(1)", "gestation_wee
{"is_male": "False", "mother_age": 26.0, "plurality": "Single(1)", "gestation_we
```

Now call `gcloud ai-platform predict` using the JSON we just created and point to our deployed `model` and `version`.

In [ ]:
```
%%bash
gcloud ai-platform predict \
    --model=babyweight \
    --json-instances=inputs.json \
    --version=ml_on_GCP# TODO 3b: Add model version
```

Lab Task #4: Use model to make batch prediction.

Batch prediction is commonly used when you have thousands to millions of predictions. It will create an actual Cloud AI Platform job for prediction. Complete **#TODO**s so we can call our deployed model on Cloud AI Platform for batch prediction.

In [ ]:
```
%%bash
INPUT=gs://${BUCKET}/babyweight/batchpred/inputs.json
OUTPUT=gs://${BUCKET}/babyweight/batchpred/outputs
gsutil cp inputs.json $INPUT
gsutil -m rm -rf $OUTPUT
gcloud ai-platform jobs submit prediction babypred_$(date -u +%y%m%d_%H%M%S) \
    --data-format=TEXT \
    --region ${REGION} \
    --input-paths=$INPUT \
    --output-path=$OUTPUT \
    --model=babyweight \
    --version=ml_on_GCP# TODO 4: Add model version
```

Lab Summary:

In this lab, we set up the environment, deployed a trained Keras model to Cloud AI Platform, online predicted from deployed model on Cloud AI Platform, and batch predicted from deployed model on Cloud AI Platform.