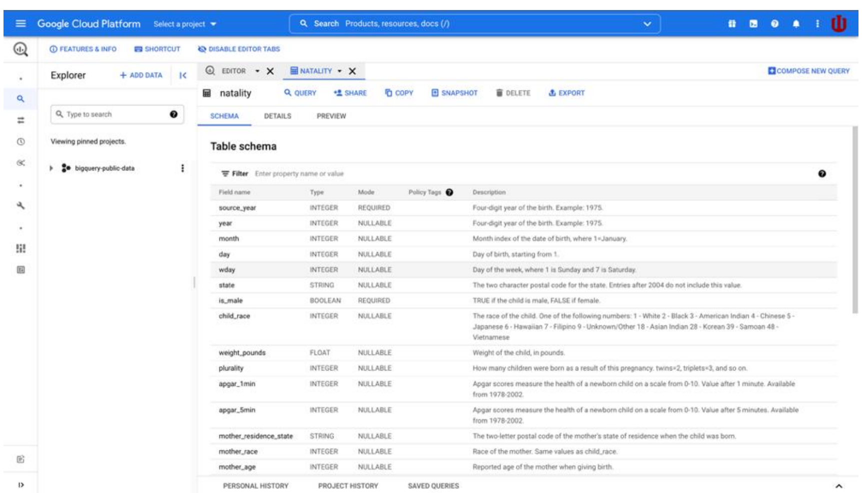
# Course Project: Big Data Concepts and Implementations

Name: **Meghana Boinpally** ID: **meboin@iu.edu**

## 1. Introduction:

The aim of this project is to predict the weights of newborns so that all babies can receive the same care. With this project we can also identify babies who may need special facilities.
The dataset used for this project is the Natality dataset. The Natality dataset reports statistics for births occurring within the United States to US residents. The Data consists of demographic attributes such as state, county, mother's race, mother's age, health and medical items. The data is retrieved from years 1969-2008 issued from birth certificates. The Births recorded are from all the 50 states.

- The data is available on Big Query: https://console.cloud.google.com/bigquery?project=bigquery-public-data&page=table&t=natality&d=samples&p=bigquery-public-data&redirect_from_classic=true
- The data represents the **Volume** characteristics of BigData: approximately **22GB** with over **137 million rows**




| Field name | Type | Mode | Policy Tags | Description |
|---|---|---|---|---|
| source_year | INTEGER | REQUIRED | | Four-digit year of the birth. Example: 1975. |
| year | INTEGER | NULLABLE | | Four-digit year of the birth. Example: 1975. |
| month | INTEGER | NULLABLE | | Month index of the date of birth, where 1=January. |
| day | INTEGER | NULLABLE | | Day of birth, starting from 1. |
| wday | INTEGER | NULLABLE | | Day of the week, where 1 is Sunday and 7 is Saturday. |
| state | STRING | NULLABLE | | The two character postal code for the state. Entries after 2004 do not include this value. |
| is_male | BOOLEAN | REQUIRED | | TRUE if the child is male, FALSE if female. |
| child_race | INTEGER | NULLABLE | | The race of the child. One of the following numbers: 1 - White 2 - Black 3 - American Indian 4 - Chinese 5 - Japanese 6 - Hawaiian 7 - Filipino 9 - Unknown/Other 18 - Asian Indian 28 - Korean 39 - Samoan 48 - Vietnamese |
| weight_pounds | FLOAT | NULLABLE | | Weight of the child, in pounds. |
| plurality | INTEGER | NULLABLE | | How many children were born as a result of this pregnancy. twins=2, triplets=3, and so on. |
| apgar_1min | INTEGER | NULLABLE | | Apgar scores measure the health of a newborn child on a scale from 0-10. Value after 1 minute. Available from 1978-2002. |
| apgar_5min | INTEGER | NULLABLE | | Apgar scores measure the health of a newborn child on a scale from 0-10. Value after 5 minutes. Available from 1978-2002. |
| mother_residence_state | STRING | NULLABLE | | The two-letter postal code of the mother's state of residence when the child was born. |
| mother_race | INTEGER | NULLABLE | | Race of the mother. Same values as child_race. |
| mother_age | INTEGER | NULLABLE | | Reported age of the mother when giving birth. |

Fig1: Table schema

## 2. Background:

All babies require some form of medical attention when they are born. Premature babies and babies born with medical concerns require additional and more specific care. According to a recent study, babies who receive more specialized care in the early stages when they need it are more likely to be healthy later in life. The date and weight of your baby's birth assist clinicians determine the type of medical care your kid will require at delivery.
The United States has 4 levels of care for the babies according to the severity. Various factors affect what the hospitals can plan for the right level care of the baby. The purpose of this study is to forecast the weight of a newborn infant. A predictive statistical model can assist better

understand a critical element of newborn health because not all babies receive the treatment they require.

## 3. Methodology:

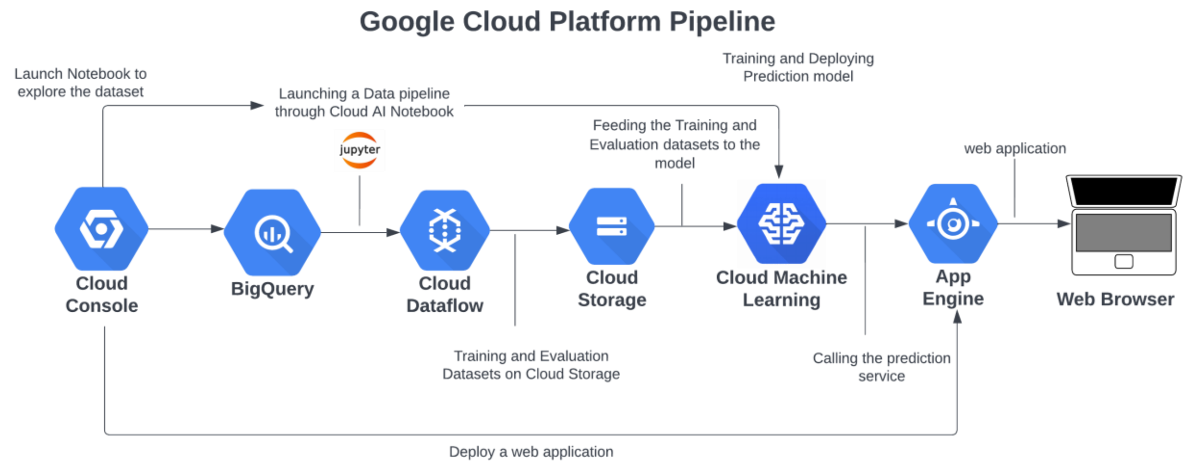



Fig 2: Project Architecture

Components Used in the **Pipeline**:

- **Cloud Storage:** Buckets represent storage in GCP Buckets contain objects which can be accessed by their own methods. Buckets contain bucketAccessControls, for use in fine-grained manipulation of an existing bucket's access controls.
- **Big Query:** BigQuery is an immutable SQL data warehouse suitable for OLAP applications. It is a serverless, cost-effective and multicloud data warehouse designed to analyze Big Data
- **Cloud DataFlow:** Google Cloud Dataflow is a cloud-based data processing service that is used for batch and real-time data streaming. We can use it to build processing pipelines for integrating, preparing, and analyzing big data collections.
- **Cloud ML:** The Google Cloud ML Engine is a hosted platform to run **distributed** machine learning training jobs and predictions at scale
- **GCP App Engine:** App Engine is a fully managed, serverless platform for developing and hosting web applications at scale

### 3.1 Creating a Project on GCP

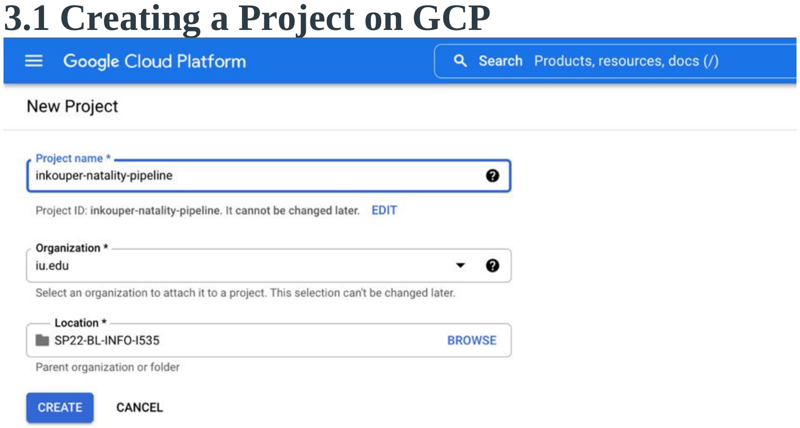



Fig 3: GCP Project

- Under the organization iu.edu I created a project “inkouper-natality-pipeline
- I also configured the subnet used for this project

## 3.2 Setting up the Cloud Storage

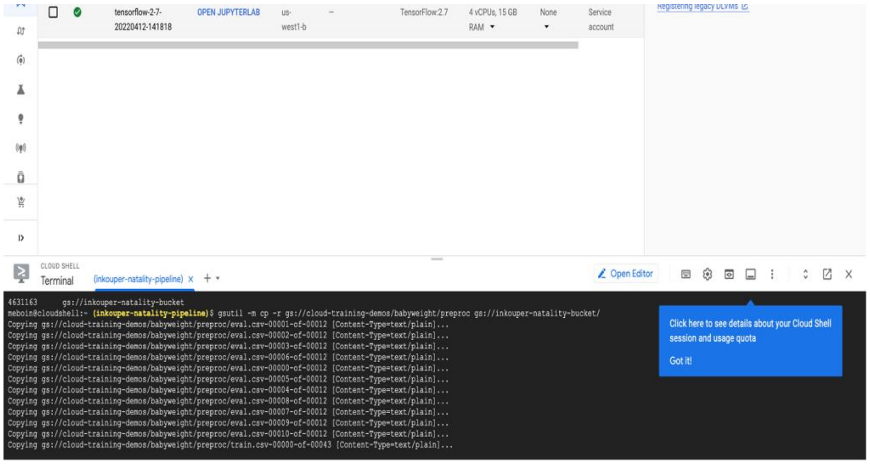



Fig 4: Setting up the GCP Bucket

- Since, I created the project I have the proper IAM permissions for it. So, I proceeded to create the storage bucket for it.
- In this I chose the bucket name “inkouper-natality-bucket” and the location for the bucket

## 3.3 Launching Notebook Instance on GCP

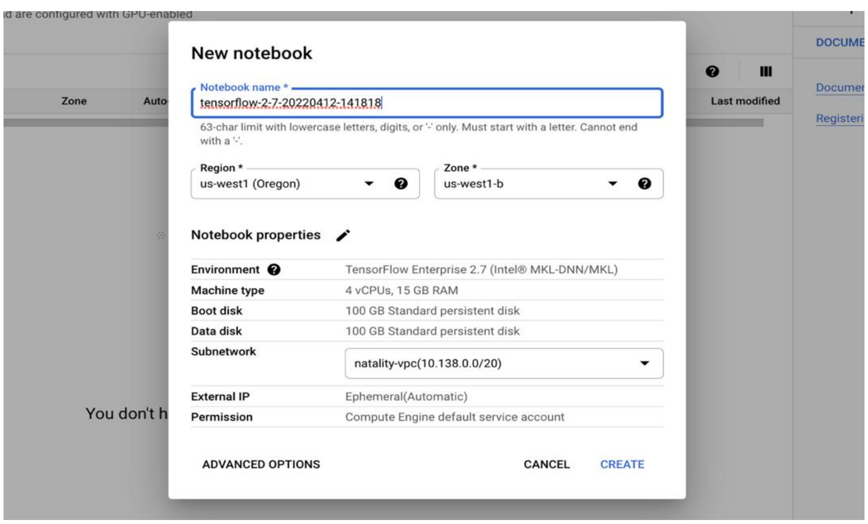



Fig 5: Launching the Jupyter Notebook

- In the navigation menu under the “**AI platform**” option I selected the notebook option
- Under the create new notebook menu I selected new instance and **TensorFlow 2.x** > **Without GPUs**

## 3.4 Invoking BigQuery

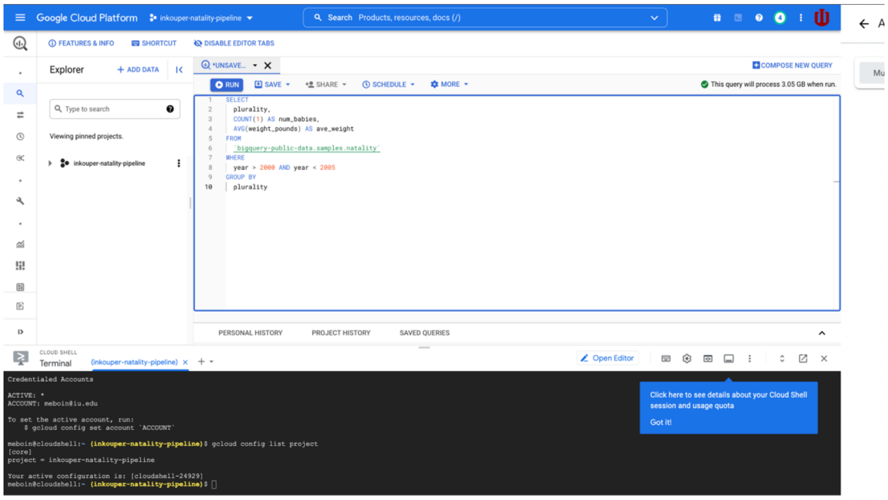



Fig 6: Executing a SQL Query on the BigQuery console

- I decided to use a BigQuery, a serverless data warehouse, to explore the natality dataset
- In the navigation menu I selected BigQuery and explored queries in the console
- The above query gives us results of plurality of babies born between 2000 and 2005

| Row | plurality | num_babies | avg_wt |
|---|---|---|---|
| 1 | 2 | 507706 | 5.166628585512564 |
| 2 | 3 | 27697 | 3.7188113817178317 |
| 3 | 5 | 325 | 2.6256986937659006 |
| 4 | 1 | 15736332 | 7.33691579350233 |
| 5 | 4 | 1846 | 2.8425094069128987 |

Results for the above executed query

## 3.5 Running Graphs on AI Platform Notebooks

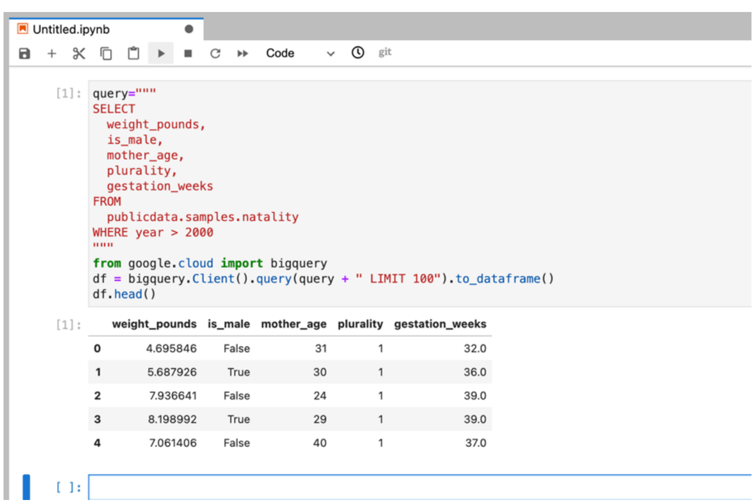



Fig 7: Running Query on Jupyter notebook

Fig 7: Getting results from BigQuery as a Pandas dataframe

- I updated to the latest version of the BigQuery Python Client Library.
- I imported BigQuery Python Client Library and initialized a client to send and receive messages from the BigQuery API.
- Then in the Jupyter notebook cell I ran queries on the BigQuery Natality dataset
- I used the results from BigQuery to create a pandas dataframe.
- I worked with the Pandas dataframe to work locally since it is smaller in size and can be locally stored
- I explored the dataset and pre-processed the features to find useful features in model building

[13]:
```
traindf.tail()
```

[13]:

| | weight_pounds | is_male | mother_age | plurality | gestation_weeks | hashmonth |
|---|---|---|---|---|---|---|
| 13075 | 8.531890 | Unknown | 31 | Single(1) | 38.0 | -774501970389208065 |
| 13076 | 6.999677 | Unknown | 22 | Single(1) | 39.0 | -774501970389208065 |
| 13077 | 6.812284 | Unknown | 28 | Single(1) | 38.0 | -774501970389208065 |
| 13078 | 7.438397 | Unknown | 33 | Single(1) | 39.0 | -774501970389208065 |
| 13079 | 7.687519 | Unknown | 20 | Multiple(2+) | 36.0 | -774501970389208065 |

[14]:
```
# Describe only does numeric columns, so you won't see plurality
traindf.describe()
```

[14]:

| | weight_pounds | mother_age | gestation_weeks | hashmonth |
|---|---|---|---|---|
| count | 25964.000000 | 25964.000000 | 25964.000000 | 2.596400e+04 |
| mean | 7.223910 | 27.357649 | 38.589817 | 3.468823e+17 |
| std | 1.338430 | 6.160435 | 2.534495 | 5.246383e+18 |
| min | 0.573202 | 13.000000 | 18.000000 | -9.183606e+18 |
| 25% | 6.563162 | 22.000000 | 38.000000 | -3.340563e+18 |
| 50% | 7.312733 | 27.000000 | 39.000000 | -3.280124e+17 |
| 75% | 8.062305 | 32.000000 | 40.000000 | 5.896568e+18 |
| max | 13.000660 | 54.000000 | 47.000000 | 8.599690e+18 |

Fig 8: Exploring the Pandas DataFrame

[4]:
```
def get_distinct_values(column_name):
  sql = """
SELECT
  {0},
  COUNT(1) AS num_babies,
  AVG(weight_pounds) AS avg_wt
FROM
  publicdata.samples.natality
WHERE
  year > 2000
GROUP BY
  {0}
  """.format(column_name)
  return bigquery.Client().query(sql).to_dataframe()

df = get_distinct_values('is_male')
df.plot(x='is_male', y='avg_wt', kind='bar');
```

[5]:
```
df = get_distinct_values('gestation_weeks')
df = df.sort_values('gestation_weeks')
df.plot(x='gestation_weeks', y='avg_wt', kind='bar');
```

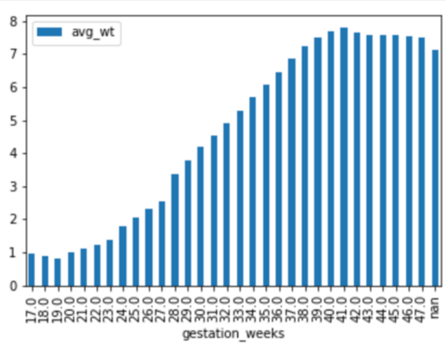



Fig 9: Weight of babies depending on the number of Gestation weeks

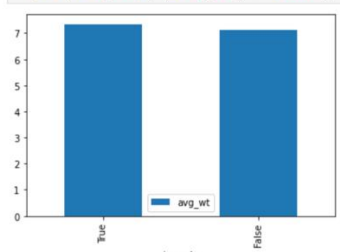



Fig 10: Avg weight of babies based on gender

- Numeric Features: mother_age, gestation_weeks, key(unique identifier)
- Categorical Features: is_male, plurality
- Target variable: weight_pounds

## 3.6 Building and Evaluating the Model

```
# Ensure the right version of Tensorflow is installed.
!pip freeze | grep tensorflow==2.1
```

```
BUCKET = 'inkouper-natality-bucket'
PROJECT = 'inkouper-natality-pipeline'
REGION = 'us-west1-b'
```

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

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

Fig 11: Setting the project name, bucket name and location to what we previously created

- I created a simple Deep Neural Network Model with the following attributes
    - Two hidden layers with **relu** activation
    - Optimizer: **adam**
    - Metric: **RMSE**
    - Loss: **mse**
    - **Batch size** of 32
    - Epochs: **5** (since the data size is huge)

- Visualized the training and validation loss to see if the model is underfitting or over fitting
- saved the model and exported it to the cloud storage
- Deployed the trained prediction model to cloud AI platform

```
[12]: # 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');
```

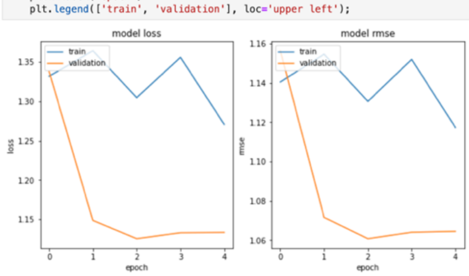



Fig 12: Model loss curve for training and validation set

## 3.7 Create ML dataset using Dataflow

- I used Cloud Dataflow to read the BigQuery data, do some preprocessing, and write it out as CSV files for further use
- Preprocessing:
    - Converting is_male from BOOL to STRING
    - Converting plurality from INTEGER to STRING where [1, 2, 3, 4, 5] becomes ["Single(1)", "Twins(2)", "Triplets(3)", "Quadruplets(4)", "Quintuplets(5)"]
- Filtering:
    - data for years later than 2000
    - baby weights greater than 0
    - mothers whose age is greater than 0
    - plurality to be greater than 0
    - the number of weeks of gestation to be greater than 0
- 
- After launching this job, since the actual processing is happening on the cloud. The job took about 17 minutes.

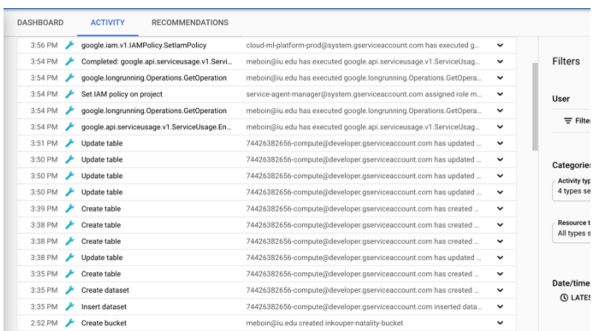



Fig 13: Job status and project activity

## 3.8 Training on Cloud AI Platform

```
[*]: %%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 \
```

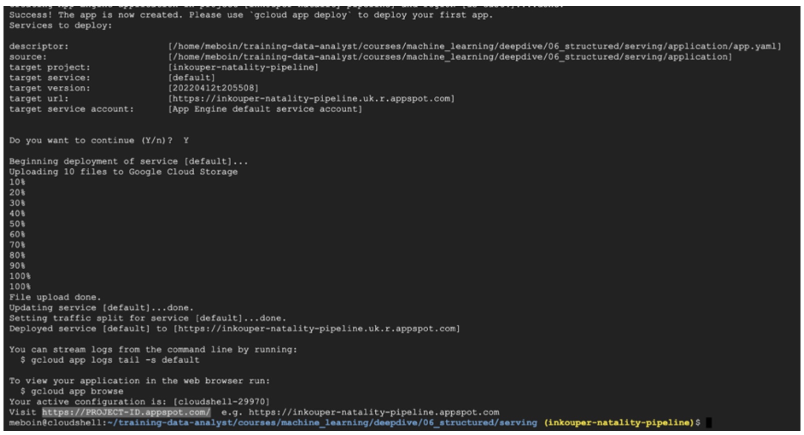
Fig 14: Model loss curve for training and validation set

- I used the BigQuery Python API to export the train and eval tables in CSV format to Google Cloud Storage
- To submit to the Cloud, I used gcloud ai platform with additional parameters for AI Platform Training Service:
    - jobname: A unique identifier for the Cloud job.
    - 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 2.1.
  - region: Cloud region to train in.
- After successfully exporting the trained model. The model is saved as saved_model.pb within the directory

## 3.9 Deploying the Trained Model

- I deployed the trained model to act as a REST web service using a gcloud call.
- Sent a JSON request to the endpoint of the service to make it predict a baby's weight.
- Cloned the github repo :
- git clone \https://github.com/GoogleCloudPlatform/training-data-analyst/
- Deployed the Web application



```
Success! The app is now created. Please use `gcloud app deploy` to deploy your first app.
Services to deploy:

descriptor:                  [/home/meboin/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving/application/app.yaml]
source:                      [/home/meboin/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving/application]
target project:              [inkouper-natality-pipeline]
target service:              [default]
target version:              [20220412t205508]
target url:                  [https://inkouper-natality-pipeline.uk.r.appspot.com]
target service account:      [App Engine default service account]


Do you want to continue (Y/n)?  Y

Beginning deployment of service [default]...
Uploading 10 files to Google Cloud Storage
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
100%
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://inkouper-natality-pipeline.uk.r.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

To view your application in the web browser run:
  $ gcloud app browse
Your active configuration is: [cloudshell-29970]
Visit https://PROJECT-ID.appspot.com/  e.g. https://inkouper-natality-pipeline.appspot.com
meboin@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving (inkouper-natality-pipeline)$
```

Fig 15: Project Url created

Url of the project

# 4. Results

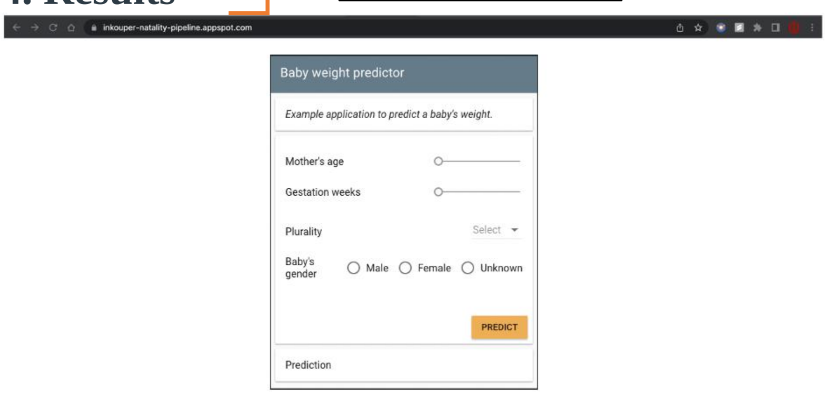



Fig 16: Screenshot of the webpage

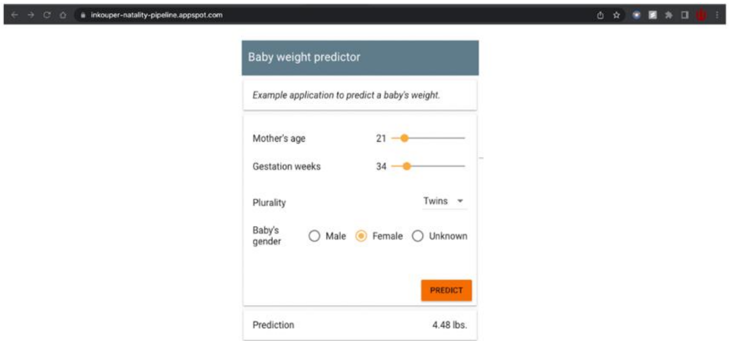



Fig 17: Example Prediction

## 4. Discussion

In the course I learned about designing Data Processing Pipeline on Google Cloud Platform. I aimed to use the workflow to implement my project. `
After some initial pre-processing and modelling, through exploratory data analysis I analyzed if the cigarette use correlated with low birth weight or pre-term birth? Does childbirth weight corelate with mothers age? I believe all these factors heavily influence what kind of care the baby should receive once it is born.

The challenge was to design a single data platform capable of handling the workflow. I read through the documentation to identify which product suited the most for the given situation at hand. Since I didn’t have the expertise, it was hard to figure out that subnet creation is really important for the project. I had a bit of trouble dealing with dependency conflicts between different Java client libraries. Each wanted a different version of a set of common dependencies. The API libraries and tools were spread across several GitHub organizations, which made it a little difficult sometimes to track down.

## 5. Conclusion

Big data analysis is a truly complex process since it has many stages to it. In this project I used different tools to manage the different aspects of big data analysis. I decided to base my project on GCP so that I can become familiar with the cloud technologies and learn and implement an end-to end ML Ops pipeline to mimic what happens at a larger company in a smaller scale.

I think it would make it safer and give an idea of what to expect from birth if we knew how the pregnancy went and what are the risk factors associated with it. Big Query, fortunately, includes this data in its example datasets. The data originates from the CDC and covers birth records in the United States from 1969 to 2008. The baby's weight, sex, race, gestational age, Apgar scores, and information regarding the mother's age, prior births, cigarette and alcohol usage. All these are the useful factors in predicting the child’s health.

## 6. References

- https://medium.com/zeotap-customer-intelligence-unleashed/designing-data-processing-pipeline-on-google-cloud-platform-gcp-part-i-5a28644c5528
- https://docs.google.com/presentation/d/e/2PACX-1vQcViYQqxjx2byva-SbOQSWcvwY3xWw8FR5K8M9q3Kv49pE4EfpFSnWgfejjEO4gGnW307ZobCvZWd-/pub?start=false&loop=false&delayms=3000&slide=id.g3431db9148_0_547
- https://medium.com/@ImJasonH/exploring-natality-data-with-bigquery-ed9b7fc6478a
- https://kiosk-dot-codelabs-site.appspot.com/codelabs/end-to-end-ml/index.html?index=..%2F..index#7
- https://github.com/GoogleCloudPlatform