Deep Learning for Fraud Detection using GCP

Global IT Security Summit
5th Edition

Table of Contents

Fraud and it's Cost

According to ClearTax

Financial fraud happens when someone deprives you of your money or otherwise harms your financial health through misleading, deceptive, or other illegal practices. This can be done through a variety of methods such as identity theft or investment fraud.

Jim Gee, Partner and National Head of Forensic Services at Crowe, said:

- “In the ten years since the first Financial Cost of Fraud report was published, the global economy has suffered rising losses each year, owing to a multitude of new and diverse threats. Losses in 2018 averaged 7.15% of expenditure, compared to 4.6% in 2007.
- “The figures quoted in the 2019 report are stark. Globally, fraud losses equate to a shocking US$5.127 trillion each year, which represents almost 70% of the $7.442 trillion which world spends on healthcare each year.

Fighting Fraud with AI and Cloud

According to Svetlana Belyalova, head of operational risk management at Rosbank

“The use of AI to fight financial fraud, both internal and external, has become a hot topic. AI is the future of fraud management, irrespective of the system you are using…It brings a lot of value in both data management and decision-making.”

To fight Financial Fraud we can use various variations of Classical Machine Learning Algorithms Like Isolation Forest, Reinforcement Learning techniques like MDP, or Deep Learning Family of Recurrent Neural Networks.

Moreover, putting these models into production adds another layer of complexity, therefore recuring to Cloud services such Google Cloud Platform is essential to the feasibility and efficiency of the End-to-End Artificial Intelligence pipelines.

The Datasets we have used

We have used two different datasets to benchmark our models:

- The first is :

Credit cards transactions by European cardholders anonymized dataset.

https://www.kaggle.com/mlg-ulb/creditcardfraud

- The second is :

Paysim synthetic dataset of mobile money transactions.

https://github.com/EdgarLopezPhD/PaySim

Deep Learning models for Fraud Detection

In the Realm of Artificial Intelligence we have many sub fields, and our focus we’ll be on Deep Learning, and specifically the family of Recurrent Neural Networks, that we have implemented using TensorFlow.

Through our research we have found that sequential models have better results if we introduce on top of them CNN layers(Convolutional Neural Networks) that extracts the best features and reduce the volume of the processed data.

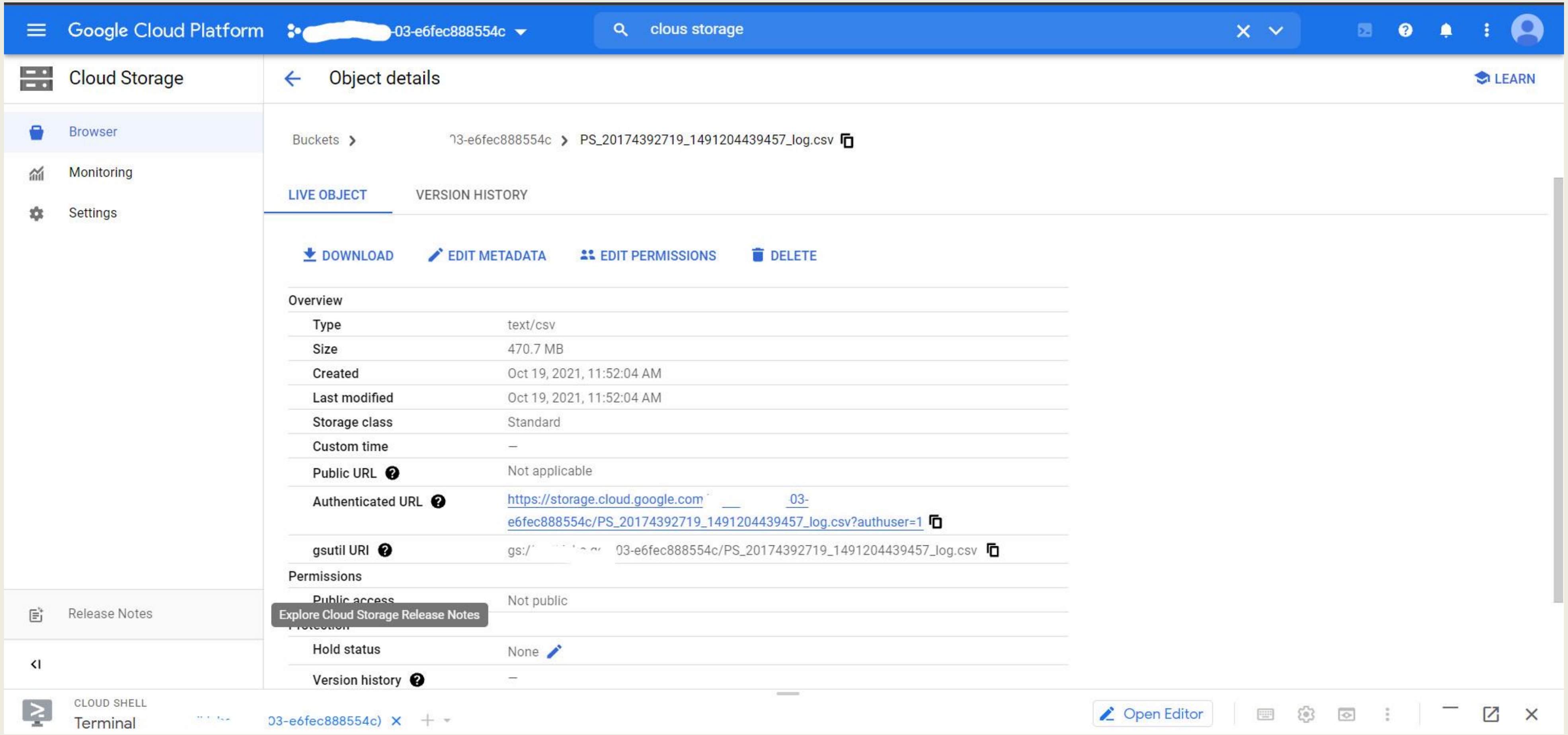
The benchmarking baseline have been established through the basic RNN(Recurrent Neural Networks), then we have used more advanced variation starting from LSTM(Long Short Term Memory) to BILSTM(Bidirectional LSTM).

These algorithms have consumed a lot of computing resources and their not time efficient, so we have used Keras which have a great hyperparameter tuning, and we replaced BILSTM with BIGRU (Bidirectional Gated Recurrent Unit).

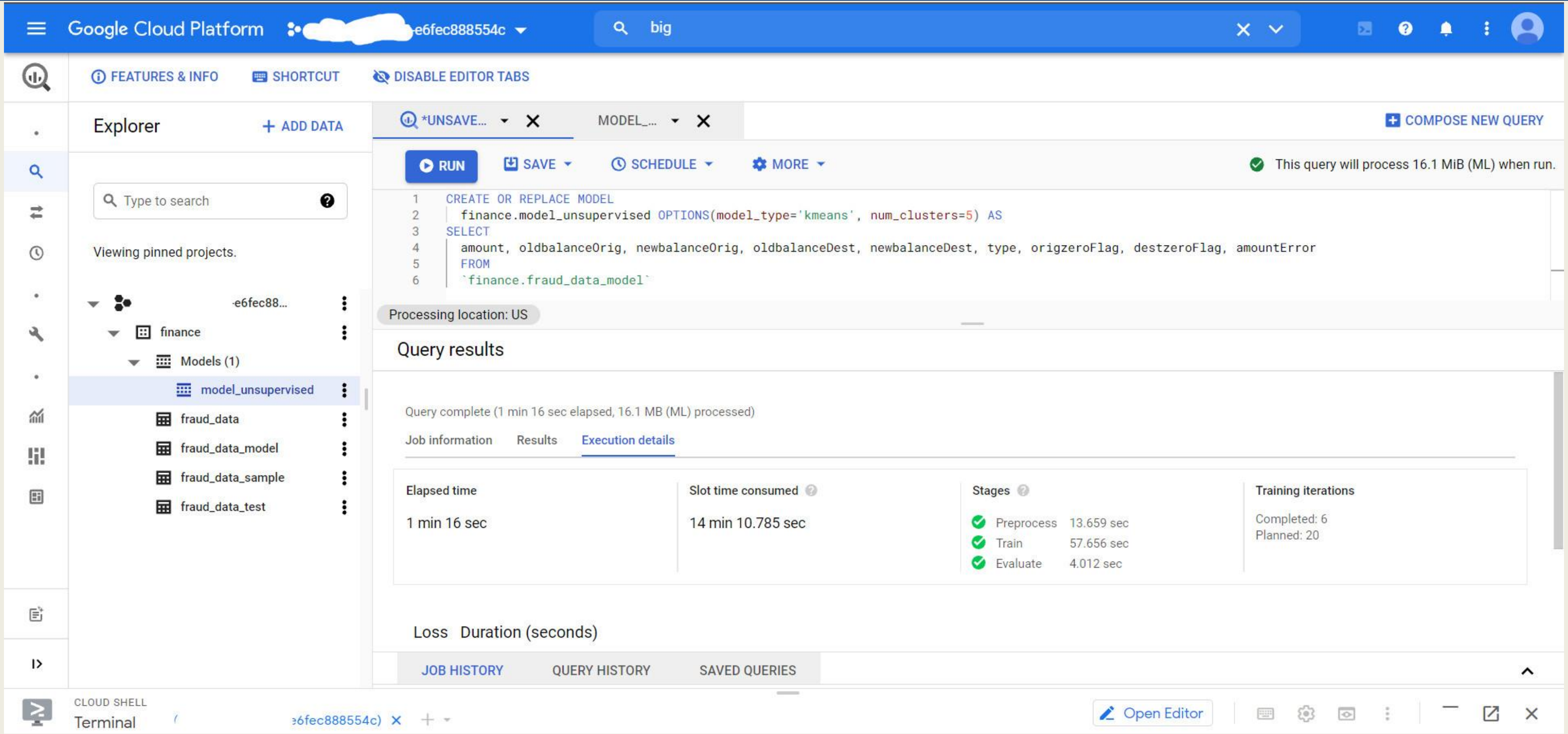
Deep Learning models for Fraud Detection

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)


model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                        input_shape=[None]),

  tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                        strides=1, padding="causal",
                        activation="relu",
                        input_shape=[None, 1]),

  tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32, return_sequences=True)),

  tf.keras.layers.LSTM(64, return_sequences=True),
  tf.keras.layers.LSTM(64, return_sequences=True),

  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 400.0)
])

optimizer = tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history_cnn_bigru = model.fit(dataset, epochs=150)
```

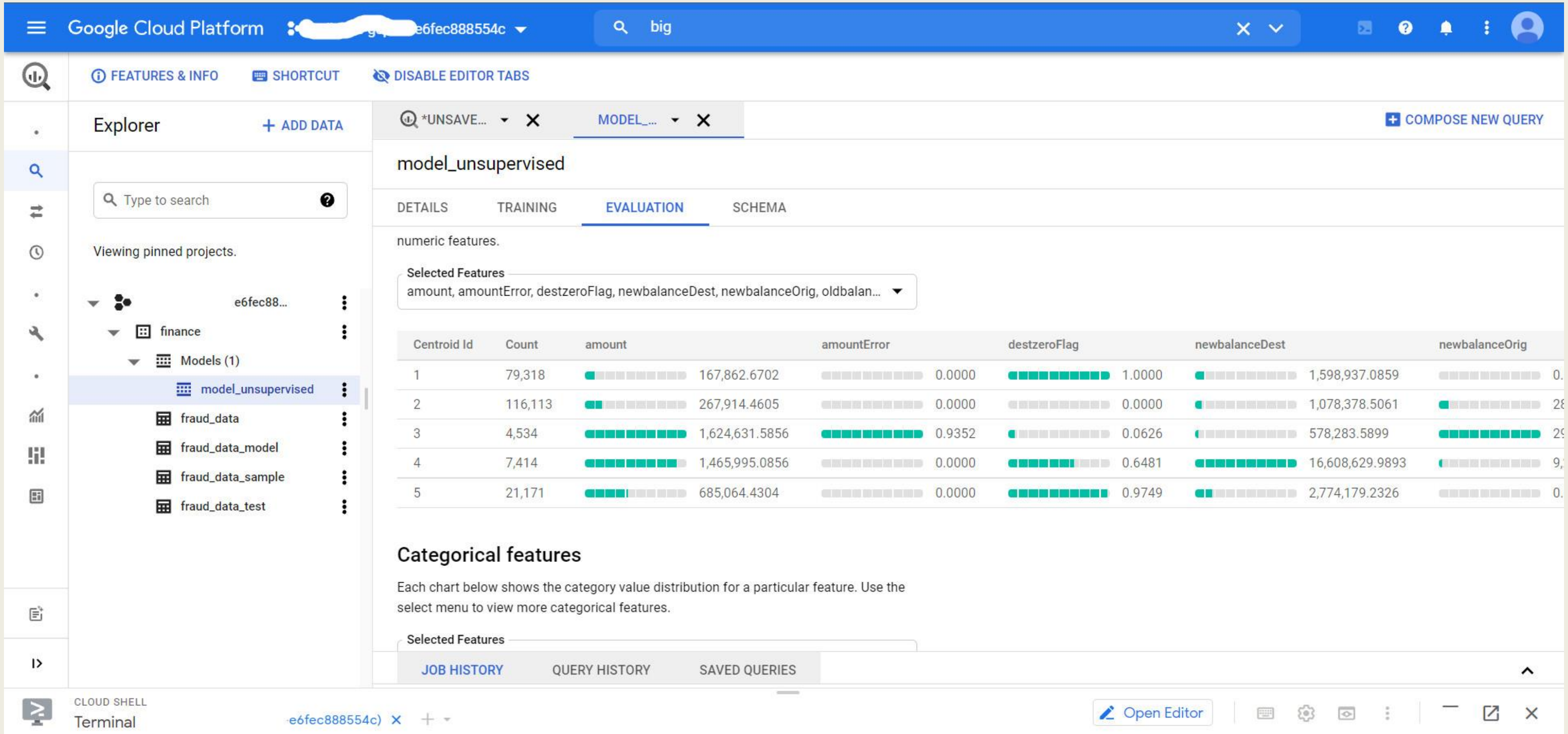
Leveraging AI models with GCP



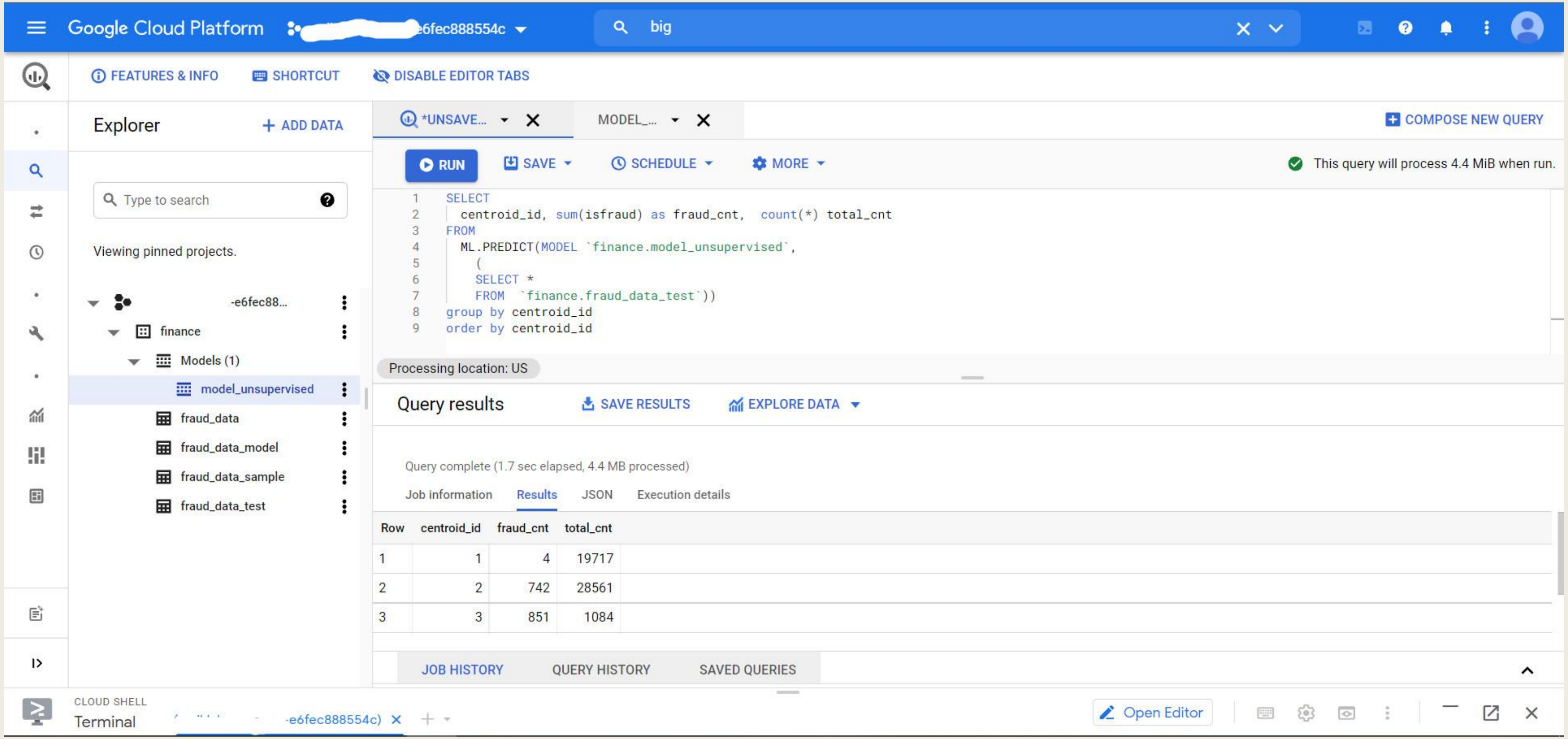
Leveraging AI models with GCP



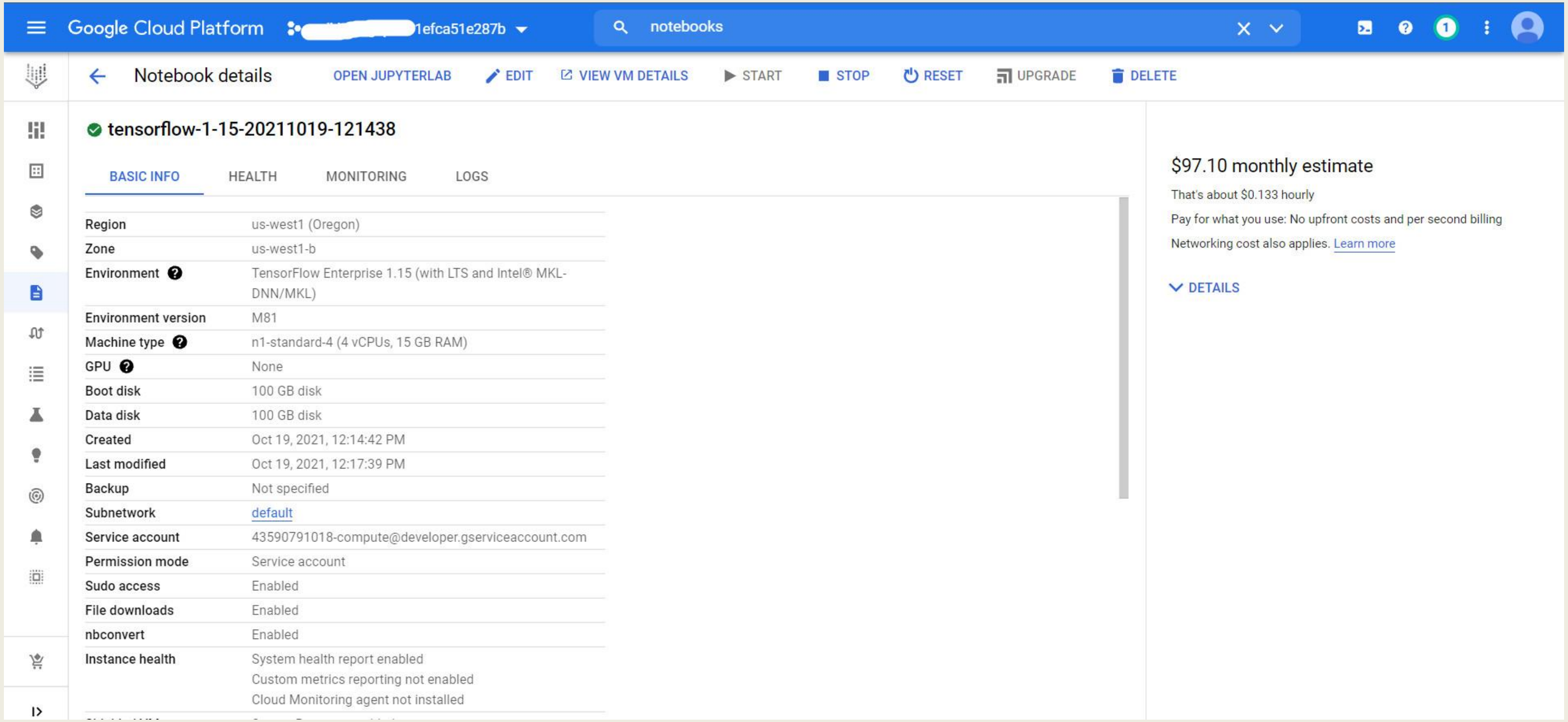
Leveraging AI models with GCP



Leveraging AI models with GCP



Leveraging AI models with GCP



Leveraging AI models with GCP

```python
import tensorflow as tf
import tensorflow.keras as keras
import tensorflow.keras.layers as layers

from tensorflow_io.bigquery import BigQueryClient

import functools

tf.enable_eager_execution()
```

```python
GCP_PROJECT_ID = 
DATASET_GCP_PROJECT_ID = GCP_PROJECT_ID # A copy of the data is saved in the user project
DATASET_ID = 'tfe_codelab'
TRAIN_TABLE_ID = 'ulb_fraud_detection_train'
VAL_TABLE_ID = 'ulb_fraud_detection_val'
TEST_TABLE_ID = 'ulb_fraud_detection_test'

FEATURES = ['Time','V1','V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V14','V15','V16','V17','V18','V19','V20','V21','V22','V23','V24','V25','V26','V27','V28','Amount']
LABEL='Class'
DTYPES=[tf.float64] * len(FEATURES) + [tf.int64]
```

```python
client = BigQueryClient()

def read_session(TABLE_ID):
    return client.read_session(
        "projects/" + GCP_PROJECT_ID, DATASET_GCP_PROJECT_ID, TABLE_ID, DATASET_ID,
        FEATURES + [LABEL], DTYPES, requested_streams=2
)

def extract_labels(input_dict):
  features = dict(input_dict)
  label = tf.cast(features.pop(LABEL), tf.float64)
  return (features, label)
```

```python
BATCH_SIZE = 32

raw_train_data = read_session(TRAIN_TABLE_ID).parallel_read_rows().map(extract_labels).batch(BATCH_SIZE)
raw_val_data = read_session(VAL_TABLE_ID).parallel_read_rows().map(extract_labels).batch(BATCH_SIZE)
raw_test_data = read_session(TEST_TABLE_ID).parallel_read_rows().map(extract_labels).batch(BATCH_SIZE)

next(iter(raw_train_data)) # Print first batch
```

```python
MEANS = [94816.7387536405, 0.0011219465482001268, -0.0021445914636999603, -0.002317402958335562,
```

Leveraging AI models with GCP

```
def norm_data(mean, data):
  data = tf.cast(data, tf.float32) * 1/(2*mean)
  return tf.reshape(data, [-1, 1])

numeric_columns = []

for i, feature in enumerate(FEATURES):
  num_col = tf.feature_column.numeric_column(feature, normalizer_fn=functools.partial(norm_data, MEANS[i]))
  numeric_columns.append(num_col)

numeric_columns
```

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)


model = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                      input_shape=[None]),

  tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                      strides=1, padding="causal",
                      activation="relu",
                      input_shape=[None, 1]),

  tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32, return_sequences=True)),
  tf.keras.layers.Bidirectional(tf.keras.layers.GRU(32, return_sequences=True)),

  tf.keras.layers.LSTM(64, return_sequences=True),
  tf.keras.layers.LSTM(64, return_sequences=True),

  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 400.0)
])

optimizer = tf.keras.optimizers.SGD(lr=1e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
history_cnn_bigru = model.fit(dataset, epochs=150)
```

Thank You

Conclusion & Perspectives

Q & A