CMPE - 255

Junjie Zhu

Project Document

Project Source Code Link:

- Final Project Folder [Google Link]

Project Track:

- Application

Application Field:

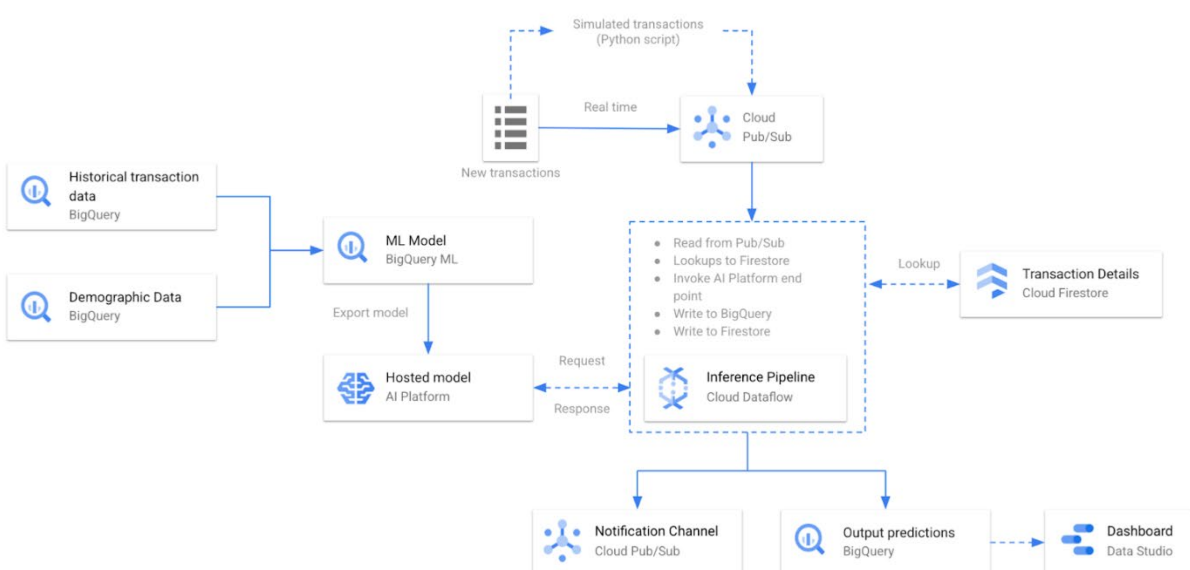
- Serverless Real-Time Credit Card Fraud Detection [Reference Google Blog Link]

Key Techniques:

- Technology Wise:
  - Colab, BigQuery, AI Platform, Cloud Firestore, Pub/Sub, Dataflow,
- Model Wise:
  - Boost Tree Model, Deep Neural Network Model, Random Forest Model
- Concept Wise:
  - Data Mining, Feature Engineering, Cloud Computing, Serverless Computing, Stream Processing, Batch Processing

Overall Architecture:

- Workflow Architecture Diagram:



Application Objective:

- This project aims to predict whether a credit card transaction is fraudulent or not in real time. After the models are trained and deployed, we will use streaming dataflow pipeline to
  - consume incoming transaction details from Cloud Pub/Sub
  - does data preprocessing (by calling Firestore for data enrichment using transaction history)
  - invokes multiple ML models deployed on AI Platform
  - stores the prediction results to BigQuery and
  - sends notification to another Pub/Sub topic when a transaction is predicted fraudulent for downstream consumptions

Utilized Dataset:

- Fake Credit Card Transaction Data Generator [GitHub Repo Link]

Dataset Details:

- BigQuery Tables (for BigQuery model Usage):
  - train_raw - Data used for ML model training
  - test_raw - Data used for ML model evaluation
  - simulation_data - Data to be used for real-time inferences
  - train_with_standard - BQ view providing standard features for training model
  - train_with_aggregates - BQ view providing aggregates features for training model
  - test_with_standard - BQ view providing standard test data for evaluating model
  - test_with_aggregates - BQ view providing aggregates test data for evaluating model
  - demographics - Data comprising customer demographics like name, gender, address
- CSV Tables (for TensorFlow model Usage):
  - train_with_aggregates.csv - Same as above
  - test_with_aggregates.csv - Same as above

About Models:

- For this pattern, we opted for XGBoost (boost tree) model which worked really well while still retaining some level of model explainability. We initially used the boosted tree classifier in BigQuery ML by using standard SQL to train the model and arrive at the probability score for each transaction. Due to the imbalanced nature of the dataset, we used F1 score and AUC to evaluate the performance of the model.
- After the initial evaluation, to boost the performance of the model we derived additional features from the dataset focusing on the frequency of the transactions and the average transaction amount over a period of time.
- As part of the solution, we have used predictions from both the models as part of the pipeline. The model using the standard features gives relatively faster results, the other model uses the features derived from looking at historical data to make the predictions, so it is relatively slow.
- For performance comparison purposes, we trained other model such as BigQuery DNN model and TensorFlow Random Forest model, we will illustrate the performance in the model performance section.

Model Differences:

- Model with standard features: It uses the features which are present in the dataset and doesn't rely on any feature generation techniques.
- Model with aggregate features: Along with the provided features, It uses feature generation techniques to compute transaction frequency, average spend etc. For a given credit card:
  - trans_freq_24 - Number of transactions in the last 24 hours
  - trans_diff - Time difference between current transaction and last transaction in seconds
  - avg_spend_pw - Average transaction amount in the past 1 week
  - avg_spend_pm - Average transaction amount in the past 1 month

Model Performance:

- 1 – Standard XGBoost Model:

Model_1_Standard_BTC:

```
5 FROM
6   BQML_CREDIT_CARD_FRAUD.MODEL_1_EVALUATION
```

| | precision | recall | accuracy | f1_score | log_loss | roc_auc |
|---|---|---|---|---|---|---|
| 0 | 0.754 | 0.642 | 0.996 | 0.694 | 0.011 | 0.994 |

```
6   BQML_CREDIT_CARD_FRAUD.MODEL_1_CONFUSION_MATRIX
```

| | expected_label | predicted_0 | predicted_1 |
|---|---|---|---|
| 0 | 0 | 10142 | 14 |
| 1 | 1 | 24 | 43 |

- 2 – Aggregate XGBoost Model:

Model_2_Aggregate_BTC:

```
5 FROM
6   BQML_CREDIT_CARD_FRAUD.MODEL_2_EVALUATION
```

| | precision | recall | accuracy | f1_score | log_loss | roc_auc |
|---|---|---|---|---|---|---|
| 0 | 0.929 | 0.765 | 0.998 | 0.839 | 0.006 | 0.997 |

```
6   BQML_CREDIT_CARD_FRAUD.MODEL_2_CONFUSION_MATRIX
```

| | expected_label | predicted_0 | predicted_1 |
|---|---|---|---|
| 0 | 0 | 10181 | 4 |
| 1 | 1 | 16 | 52 |

- 3 – Aggregate DNN Model:

Model_3_Aggregate_DNN:

```
5 FROM
6   BQML_CREDIT_CARD_FRAUD.MODEL_3_EVALUATION
```

| | precision | recall | accuracy | f1_score | log_loss | roc_auc |
|---|---|---|---|---|---|---|
| 0 | 0.833 | 0.662 | 0.997 | 0.738 | 0.011 | 0.991 |

```
6   BQML_CREDIT_CARD_FRAUD.MODEL_3_CONFUSION_MATRIX
```

| | expected_label | predicted_0 | predicted_1 |
|---|---|---|---|
| 0 | 0 | 10176 | 9 |
| 1 | 1 | 23 | 45 |

- 4 – Aggregate Random Forest Model:

Model_4_Aggregate_Random_Forest(Tensorflow):

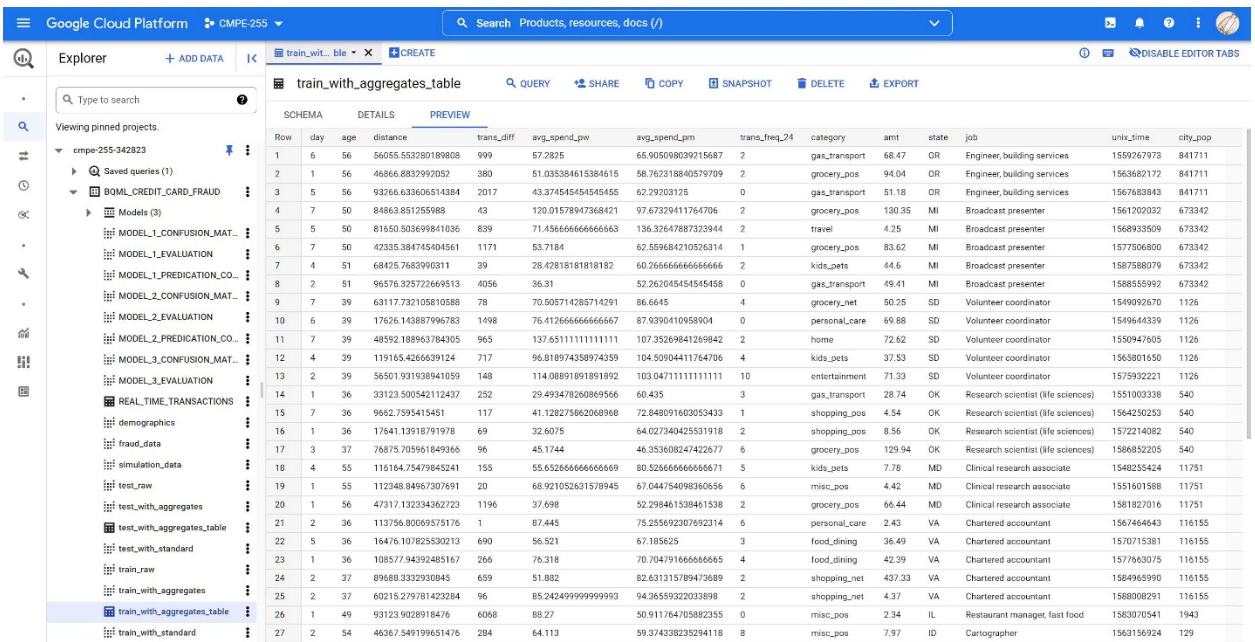
```
501/501 [===========================

loss: 0.0000
tp: 1402.0000
fp: 80.0000
tn: 497943.0000
fn: 722.0000
accuracy: 0.9984
precision: 0.9460
recall: 0.6601
auc: 0.9792
prc: 0.8568
```

F1_score: 0.7924

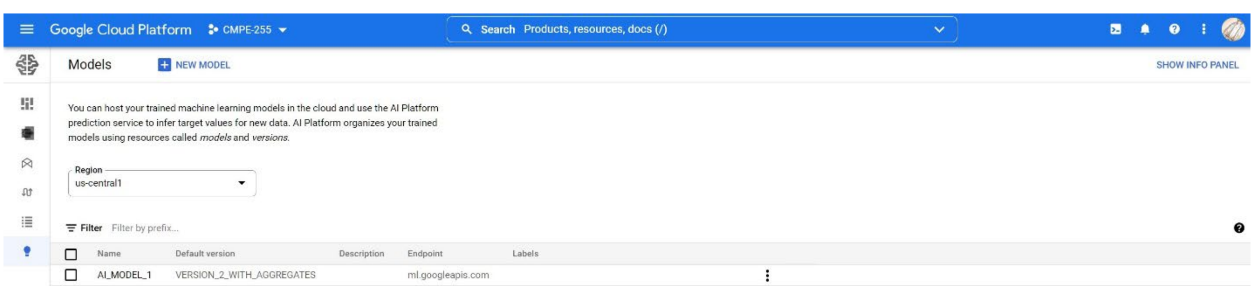
- Judge by F1 and AUC scores, the model performance in the descending order is as follow:
  - Model 2 > Model 4 > Model 3 > Model 1
- Conclusion:
  - All model works well on identifying the normal transaction but works relatively poor on identifying all fraud transactions from all transactions (low recall)
  - Compared to the standard features, the aggregates features boost the overall model performance, especially precision and recall
  - Without further hyper parameter tunning, Boost Tree Model works the best to the training dataset (where data is unbalanced)

Workflow Screening:

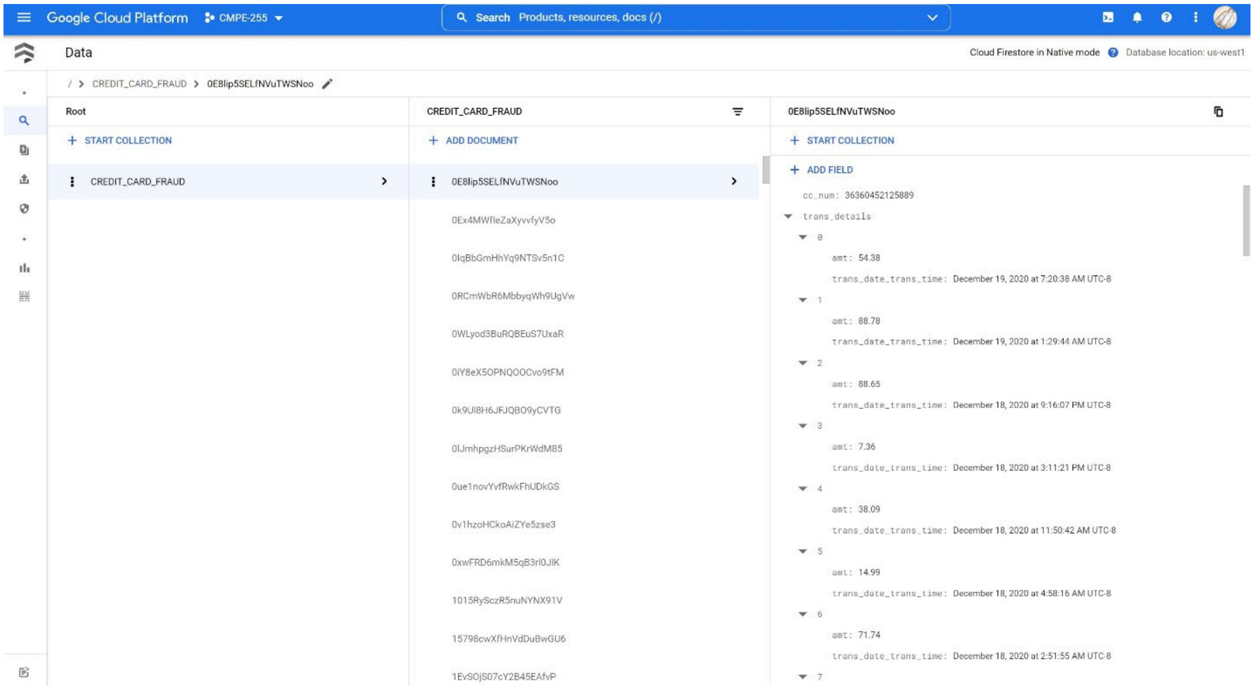
- BigQuery



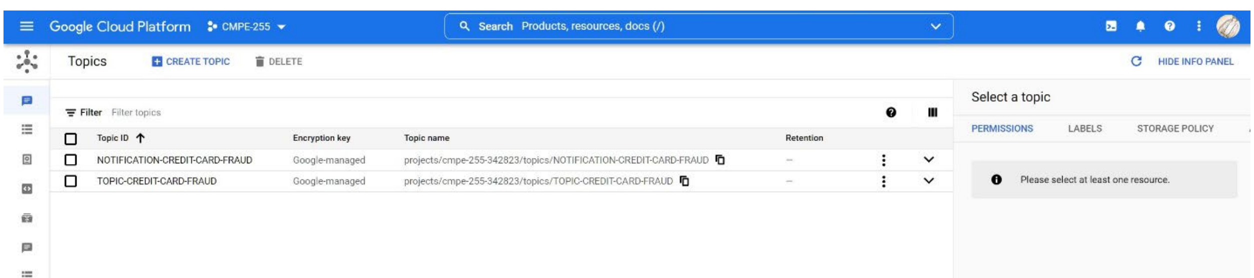

- AI Platform

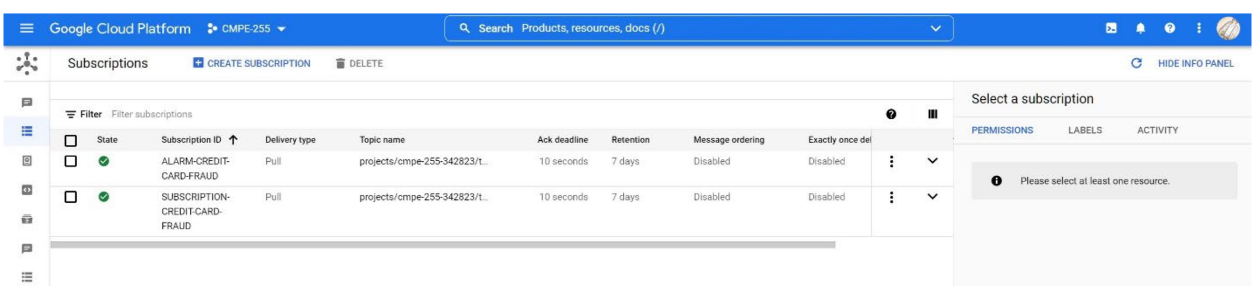


➢ Firestore

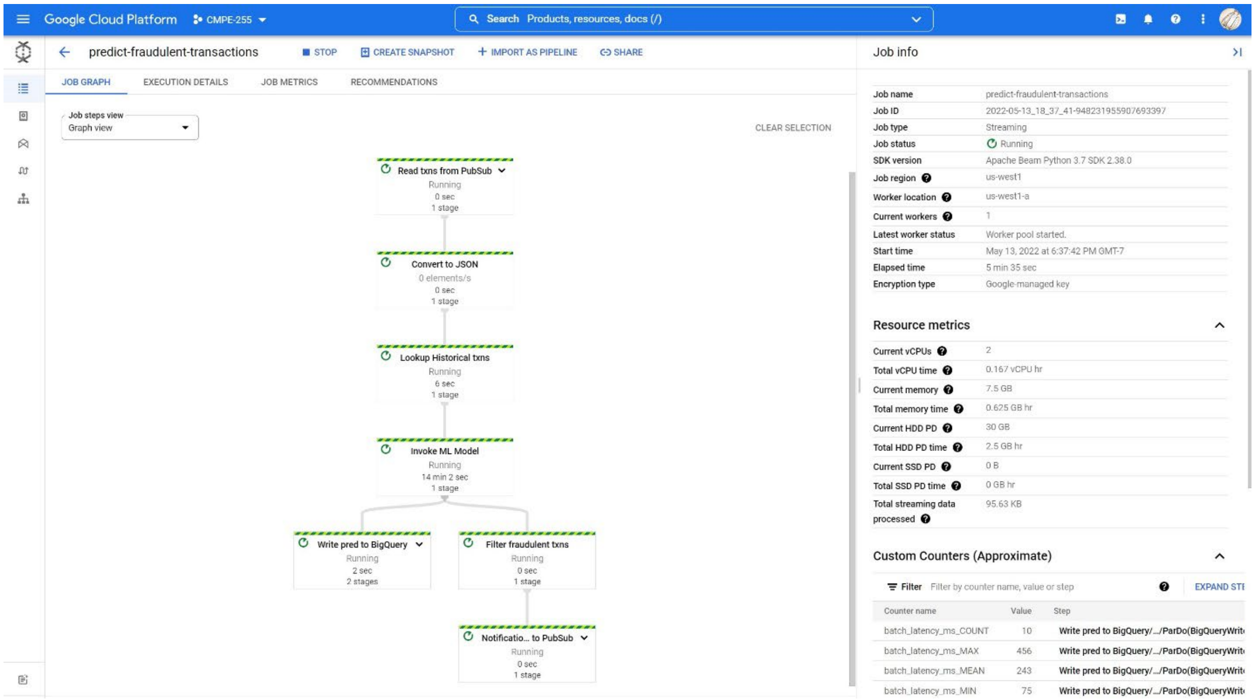



➢ Pub/Sub



➢ DataFlow



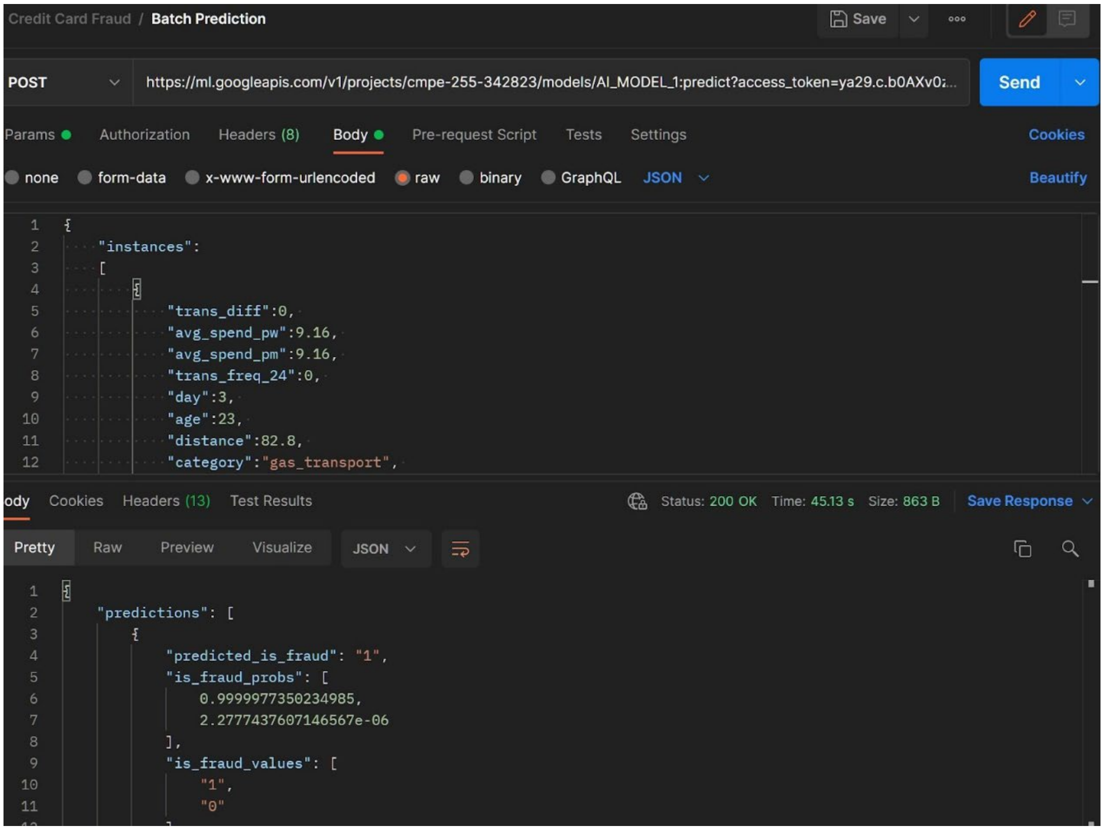

➢ Batch Processing

Via Terminal RPC:

```
[ ]  !gcloud ai-platform predict --model $AI_MODEL_NAME \
     --version $VERSION_NAME_WITH_STANDARD \
     --region global \
     --json-instances sample_inputs/input_w_standard.json

     Using endpoint [https://ml.googleapis.com/]
     IS_FRAUD_PROBS                              IS_FRAUD_VALUES  PREDICTED_IS_FRAUD
     [0.9293921589851379, 0.07060789316892624]   ['1', '0']       1
```

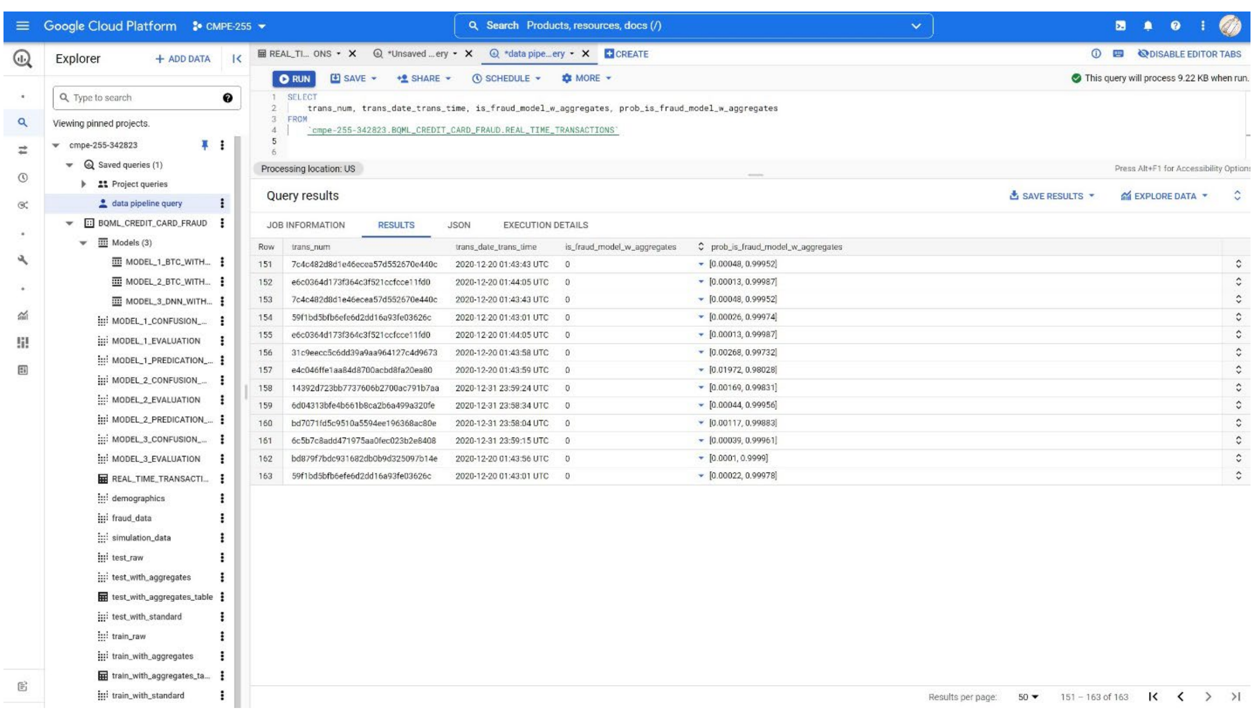
```
[ ]  !gcloud ai-platform predict --model $AI_MODEL_NAME \
     --version $VERSION_NAME_WITH_AGGREGATES \
     --region global \
     --json-instances sample_inputs/input_w_aggregates.json

     Using endpoint [https://ml.googleapis.com/]
     IS_FRAUD_PROBS                                 IS_FRAUD_VALUES  PREDICTED_IS_FRAUD
     [0.9999977350234985, 2.2777437607146567e-06]   ['1', '0']       1
```

Via Postman Rest API:

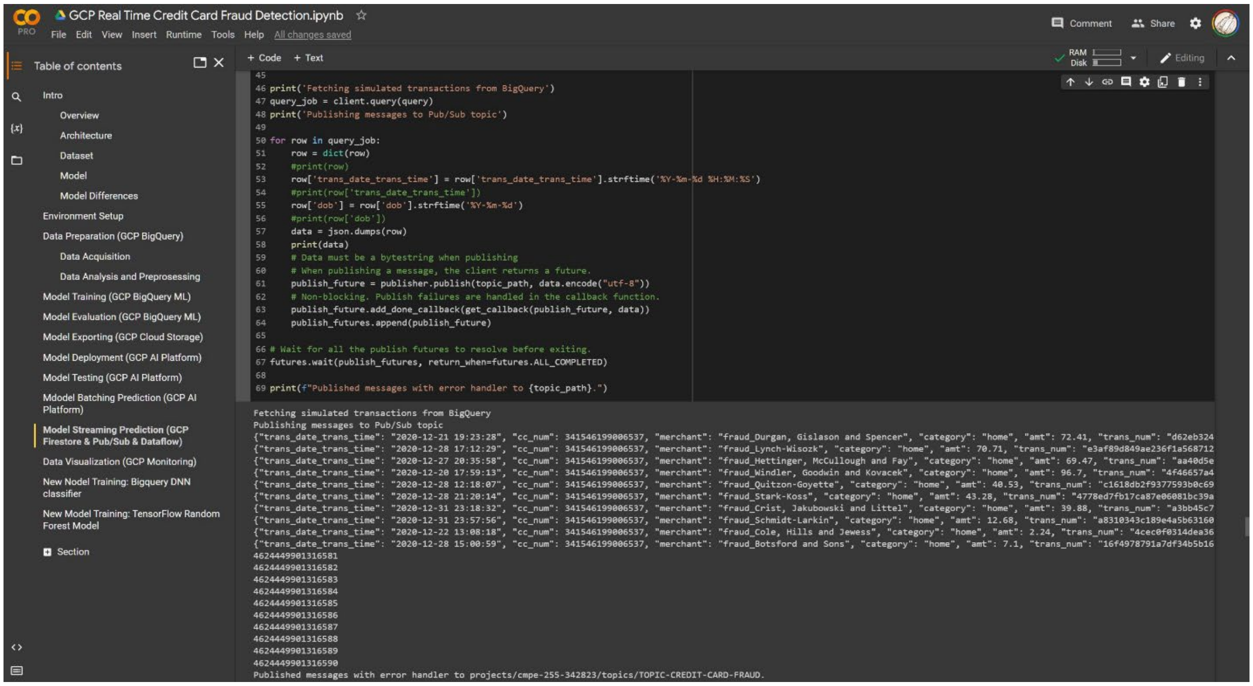



➢ Stream Processing

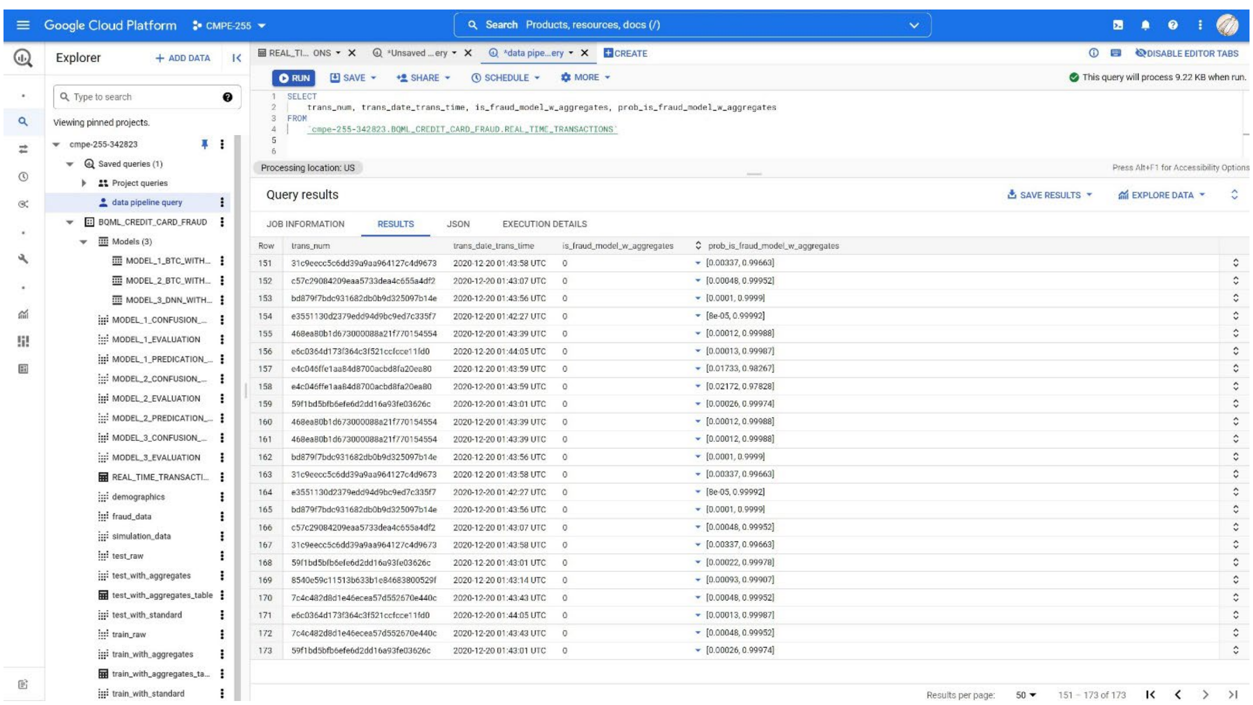
Real time transaction table (Before sending transactions to the publisher)



Sending transactions to the publisher (10 transactions per call…to save budget!)




Real time transaction table (After sent transactions to the publisher)




Thank you!