

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, TensorFlow, BigQuery, AI Platform, Cloud Firestore, Pub/Sub, Dataflow,

Model Wise:

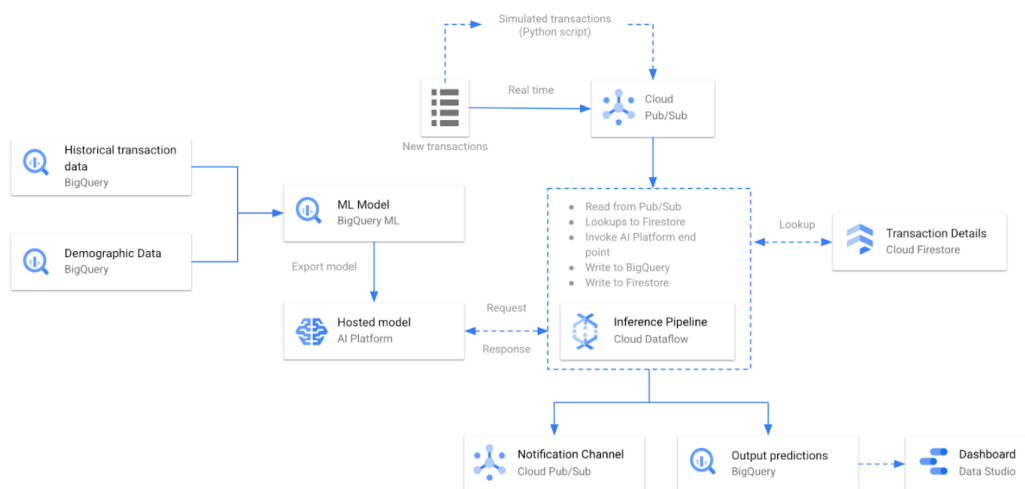
- Boost Tree Model, Deep Neural Network Model, Random Forest Model, Gradient Boost Tree Model, Cart Tree Model

Concept Wise:

- Machine Learning, 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 the 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 the 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 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 models such as BigQuery DNN model and the 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 don't rely on any feature generation techniques.

Models 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:

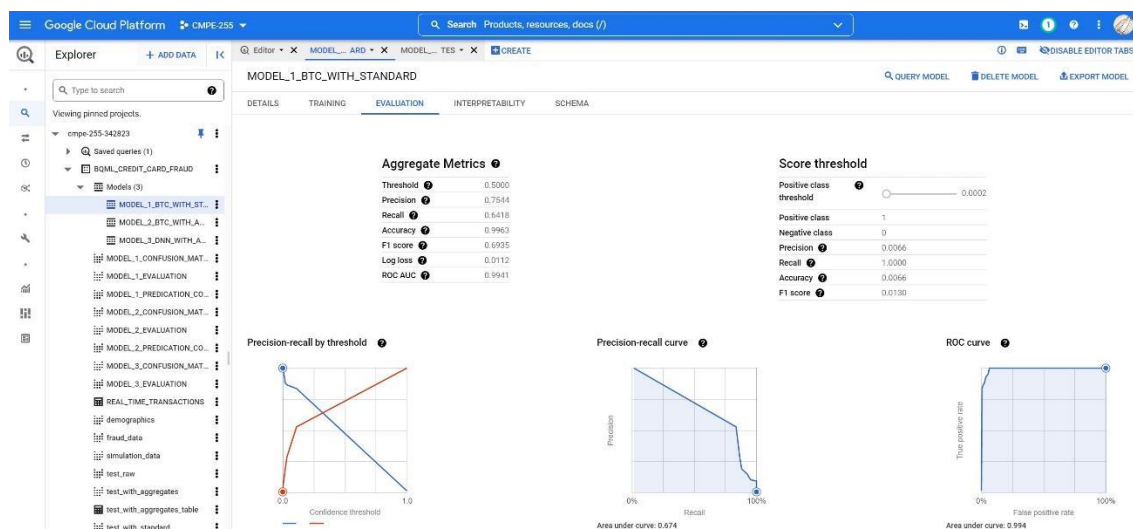
- trans_freq_24 - Number of transactions in the last 24 hours
- trans_diff - Time difference between current transaction and last transaction in seconds
- avg_spend_pw - Average transaction amount in the past 1 week
- avg_spend_pm - Average transaction amount in the past 1 month

Model Performance:

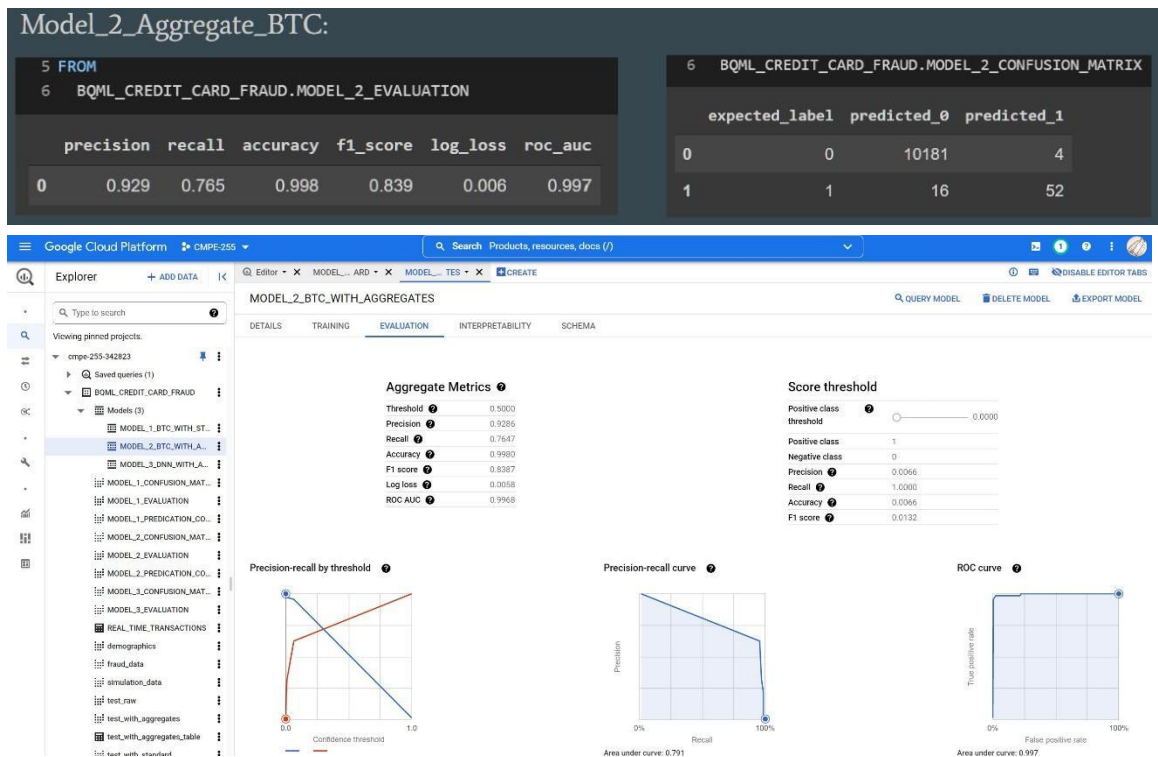
1 – Standard Boost Tree Model:

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

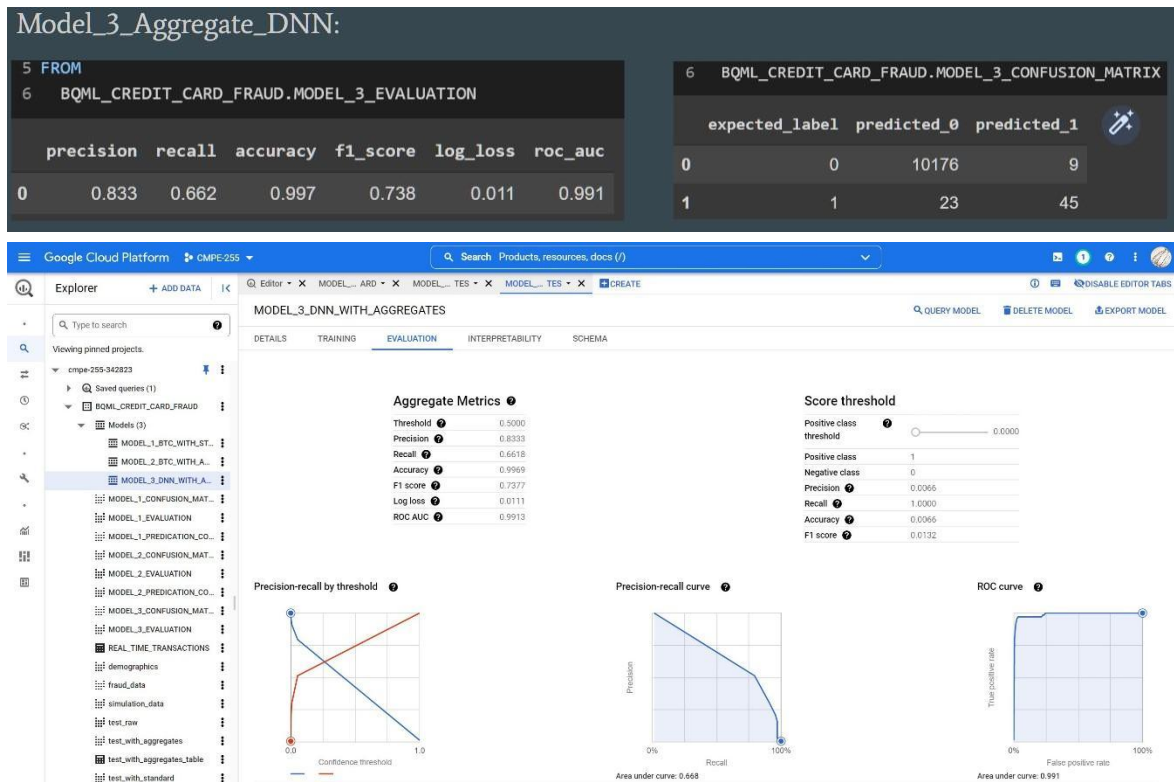
6 BQML_CREDIT_CARD_FRAUD.MODEL_1_CONFUSION_MATRIX			
	expected_label	predicted_0	predicted_1
0	0	10142	14
1	1	24	43



2 – Aggregate XGBoost Model:

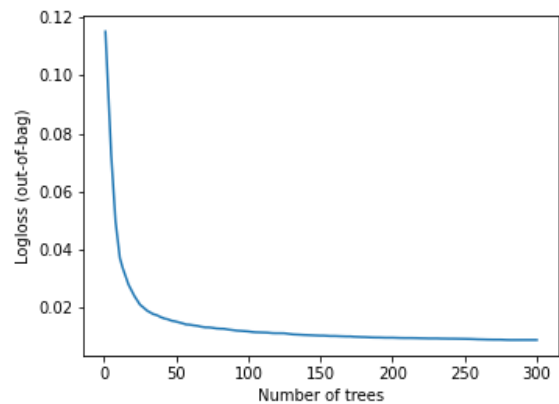
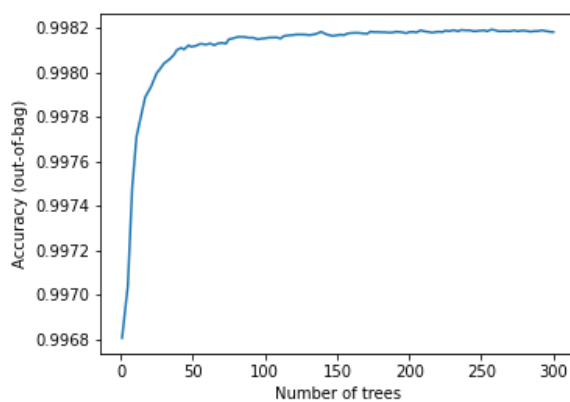
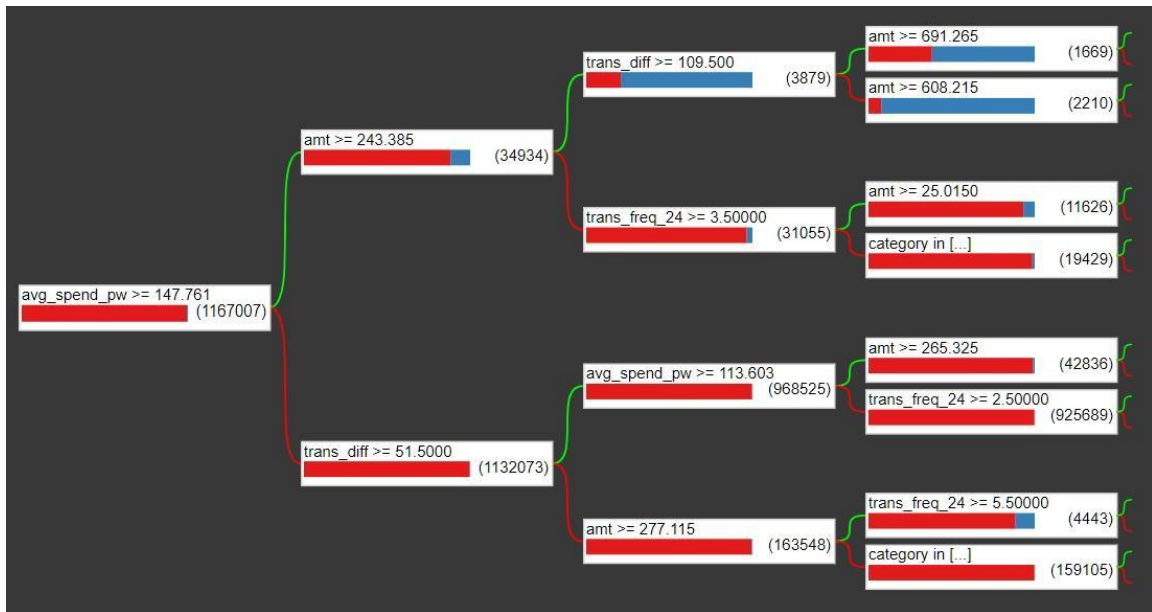


3 – Aggregate DNN Model:



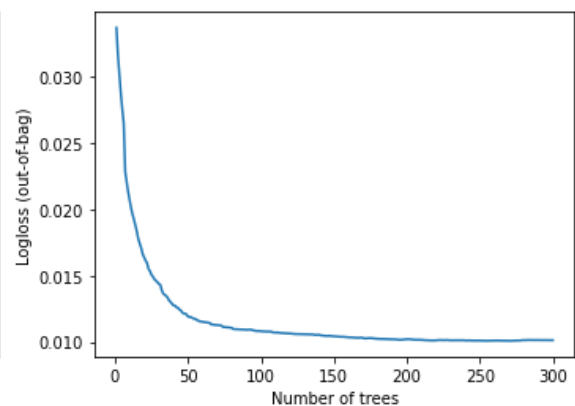
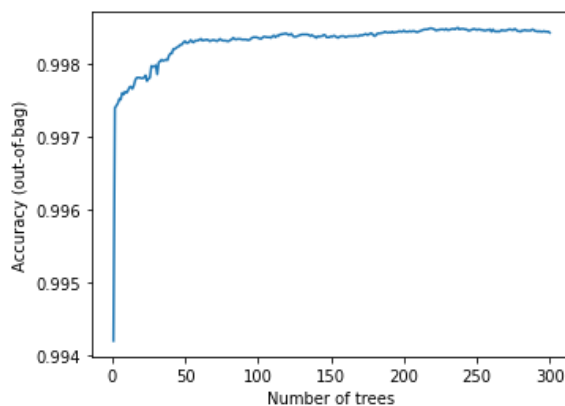
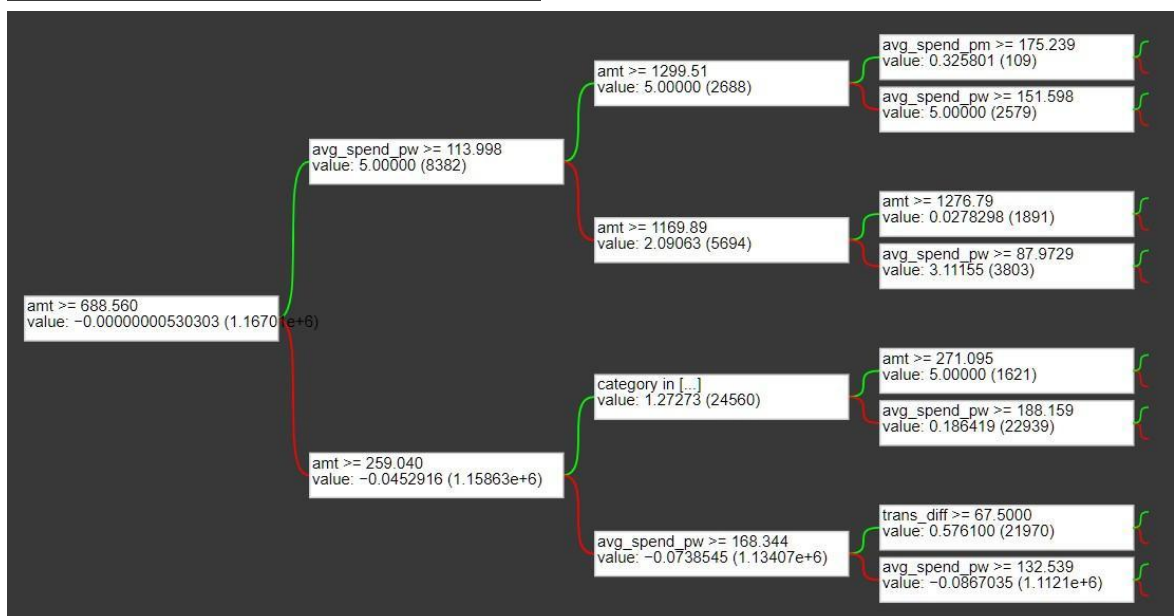
4 – Aggregate Random Forest Model:

```
loss: 0.0000
tp: 1390.0000
fp: 78.0000
tn: 497945.0000
fn: 734.0000
accuracy: 0.9984
precision: 0.9469
recall: 0.6544
auc: 0.9801
prc: 0.8551
f1: 0.7739
```



5 – Gradient Boosted Trees Model:

```
loss: 0.0000
tp: 1569.0000
fp: 249.0000
tn: 497774.0000
fn: 555.0000
accuracy: 0.9984
precision: 0.8630
recall: 0.7387
auc: 0.9749
prc: 0.8407
f1: 0.7960
```



6 – Cart Tree Model:

```
loss: 0.0000
tp: 1321.0000
fp: 282.0000
tn: 497741.0000
fn: 803.0000
accuracy: 0.9978
precision: 0.8241
recall: 0.6219
auc: 0.9178
prc: 0.6820
f1: 0.7089
```



Conclusion:

- We want to capture fraud transactions, so we aim for high TP, and low FN, high recall, and high F1, so overall performance in descending order:
 - Model 2 > Model 5 > Model 4 > Model 3 > Model 6 > Model 1
- All model works well on identifying the normal transaction, and works well on identifying the real fraud transactions (high precision) but works relatively poor on identifying all fraud transactions from all transactions (low recall)

- Compared to the standard features, the aggregates features successfully boost the overall model performance, especially precision and recall
- Without further hyperparameter tuning, the Boost Tree Model works the best for the training dataset (where data is unbalanced)

Workflow Screening:

BigQuery

Google Cloud Platform CMPE-255

Search Products, resources, docs (/)

Explorer + ADD DATA

train_wit... ble X CREATE

train_with_aggregates_table QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

Row	day	age	distance	trans_diff	avg_spend_pw	avg_spend_pm	trans_freq_24	category	amt	state	job	unix_time	city_pop
1	6	56	56055.553280189808	999	57.2825	65.905098039215687	2	gas_transport	68.47	DR	Engineer, building services	1559267973	841711
2	1	56	46866.8832992052	380	51.035384615384615	58.762318840579709	2	grocery_pos	94.04	DR	Engineer, building services	1563682172	841711
3	5	56	93266.633606514384	2017	43.374545454545455	62.29203125	0	gas_transport	51.18	DR	Engineer, building services	1567683843	841711
4	7	50	84863.851259988	43	120.01578947368421	97.673294111764706	2	grocery_pos	130.35	MI	Broadcast presenter	1561202032	673342
5	5	50	81650.503699841036	839	71.456666666666663	136.32647887323944	2	travel	4.25	MI	Broadcast presenter	1568933509	673342
6	7	50	42335.384745404561	1171	53.7184	62.559684210526314	1	grocery_pos	83.62	MI	Broadcast presenter	1577506800	673342
7	4	51	68425.7683990311	39	28.42818181818182	60.266666666666666	2	kids_pets	44.6	MI	Broadcast presenter	1587588079	673342
8	2	51	96576.325722669513	4056	36.31	52.262045454545458	0	gas_transport	49.41	MI	Broadcast presenter	1588555992	673342
9	7	39	63117.732105810588	78	70.505714285714291	86.6645	4	grocery_net	50.25	SD	Volunteer coordinator	1549092670	1126
10	6	39	17626.143887996783	1498	76.412666666666667	87.9390410958904	0	personal_care	69.88	SD	Volunteer coordinator	1549644339	1126
11	7	39	48592.189963784305	965	137.65111111111111	107.35269841269842	2	home	72.62	SD	Volunteer coordinator	1550947605	1126
12	4	39	119165.4266639124	717	96.818974358974359	104.50904411764706	4	kids_pets	37.53	SD	Volunteer coordinator	1565801650	1126
13	2	39	56501.931938941059	148	114.0891891891892	103.04711111111111	10	entertainment	71.33	SD	Volunteer coordinator	1575932221	1126
14	1	36	33123.500542112437	252	29.493478260869566	60.435	3	gas_transport	28.74	OK	Research scientist (life sciences)	1551003338	540
15	7	36	9662.7595415451	117	41.128275862089968	72.848091603053433	1	shopping_pos	4.54	OK	Research scientist (life sciences)	1564250253	540
16	1	36	17641.13918791978	69	32.6075	64.027340425531918	2	shopping_pos	8.56	OK	Research scientist (life sciences)	1572214082	540
17	3	37	76875.705961849366	96	45.1744	46.35360824722677	6	grocery_pos	129.94	OK	Research scientist (life sciences)	1586852205	540
18	4	55	116164.75479845241	155	55.652666666666669	80.526666666666671	5	kids_pets	7.78	MD	Clinical research associate	1548255424	11751
19	1	55	112348.84967307691	20	68.921052631578945	67.044754098360656	6	misc_pos	4.42	MD	Clinical research associate	1551601588	11751
20	1	56	47317.132234362723	1196	37.698	52.298461538461538	2	grocery_pos	66.44	MD	Clinical research associate	1581827016	11751
21	2	36	113756.80069575176	1	87.445	75.255692307692314	6	personal_care	2.43	VA	Chartered accountant	1567464643	116155
22	5	36	16476.107825300213	690	56.521	67.185625	3	food_dining	36.49	VA	Chartered accountant	1570715381	116155
23	1	36	10857.94392485167	266	76.318	70.704791666666665	4	food_dining	42.39	VA	Chartered accountant	1577663075	116155
24	2	37	89688.3332930845	659	51.882	82.631315789473689	2	shopping_net	437.33	VA	Chartered accountant	1584965990	116155
25	2	37	60215.279781423284	96	85.242499999999993	94.36555322033898	2	shopping_net	4.37	VA	Chartered accountant	1588008291	116155
26	1	49	93123.9028918476	6068	88.27	50.911764705882355	0	misc_pos	2.34	IL	Restaurant manager, fast food	1583070541	1943
27	2	54	46367.549199651476	284	64.113	59.374338235294118	8	misc_pos	7.97	ID	Cartographer	1563156924	129

AI Platform

Google Cloud Platform CMPE-255

Search Products, resources, docs (/)

Models + NEW MODEL

SHOW INFO PANEL

You can host your trained machine learning models in the cloud and use the AI Platform prediction service to infer target values for new data. AI Platform organizes your trained models using resources called *models* and *versions*.

Region: us-central1

Filter: Filter by prefix...

Name	Default version	Description	Endpoint	Labels
AI_MODEL_1	VERSION_2_WITH_AGGREGATES		ml.googleapis.com	

Firestore

The screenshot shows the Google Cloud Platform console for the project 'cmpe-255'. The 'Data' section is active, displaying the 'CREDIT_CARD_FRAUD' collection. The collection contains 10 documents. The right-hand pane shows the details of a selected document, which includes a 'cc_num' field with the value '36360452125889' and a 'trans_details' array containing 8 transaction records. Each record in the array includes an 'amt' (amount) and a 'trans_date_trans_time' (timestamp).

Document ID	Document Content
0Ex4MwflcZaXyvvfY5o	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0lqBbGmHhYq9NTSv5n1C	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0RCmWbR6MbyqWH9UgVv	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0Wlyod3BuRQBEu57UxaR	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0Y8eXSOPNQO0Co9fEM	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0k9Uj8H6JfJB09yCVTG	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0UmhpgzHSurPKYWM85	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0ue1novVvRwFhUdkGS	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0v1hozHCkaAiZYe5zse3	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
0xwFRD6mIM5qS3r0UJK	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
101SRySczR5nuNYNX91V	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
15798cwXfHnVdDuBwGU6	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }
1EvS0jS07cy2B45EAfP	{ "cc_num": "36360452125889", "trans_details": [{ "amt": 54.38, "trans_date_trans_time": "2020-12-19T20:38:00Z" }, { "amt": 88.78, "trans_date_trans_time": "2020-12-19T12:44:00Z" }, { "amt": 88.65, "trans_date_trans_time": "2020-12-18T16:07:00Z" }, { "amt": 7.36, "trans_date_trans_time": "2020-12-18T11:21:00Z" }, { "amt": 38.09, "trans_date_trans_time": "2020-12-18T11:50:42Z" }, { "amt": 14.99, "trans_date_trans_time": "2020-12-18T04:58:16Z" }, { "amt": 71.74, "trans_date_trans_time": "2020-12-18T02:51:55Z" }, { "amt": 1.25, "trans_date_trans_time": "2020-12-18T02:51:55Z" }] }

Pub/Sub

The screenshot shows the Google Cloud Platform console for the project 'cmpe-255'. The 'Topics' section is active, displaying a table of topics. The 'TOPIC-CREDIT-CARD-FRAUD' topic is selected, and its details are shown on the right. The details include the topic name, retention policy, and a list of permissions.

Topic ID	Encryption key	Topic name	Retention
NOTIFICATION-CREDIT-CARD-FRAUD	Google-managed	projects/cmpe-255-342823/topics/NOTIFICATION-CREDIT-CARD-FRAUD	...
TOPIC-CREDIT-CARD-FRAUD	Google-managed	projects/cmpe-255-342823/topics/TOPIC-CREDIT-CARD-FRAUD	...

The screenshot shows the Google Cloud Platform console for the project 'cmpe-255'. The 'Subscriptions' section is active, displaying a table of subscriptions. The 'SUBSCRIPTION-CREDIT-CARD-FRAUD' subscription is selected, and its details are shown on the right. The details include the subscription name, delivery type, ack deadline, retention, message ordering, and a list of permissions.

State	Subscription ID	Delivery type	Topic name	Ack deadline	Retention	Message ordering	Exactly once del
✓	ALARM-CREDIT-CARD-FRAUD	Pull	projects/cmpe-255-342823/t...	10 seconds	7 days	Disabled	Disabled
✓	SUBSCRIPTION-CREDIT-CARD-FRAUD	Pull	projects/cmpe-255-342823/t...	10 seconds	7 days	Disabled	Disabled

DataFlow

The screenshot displays the Google Cloud Platform DataFlow console for a pipeline named "predict-fraudulent-transactions". The pipeline is in a "Running" state. The job graph shows the following steps:

- Read txns from PubSub (Running, 0 sec, 1 stage)
- Convert to JSON (0 elements/s, 0 sec, 1 stage)
- Lookup Historical txns (Running, 8 sec, 1 stage)
- Invoke ML Model (Running, 14 min 2 sec, 1 stage)
- Write pred to BigQuery (Running, 2 sec, 2 stages)
- Filter fraudulent txns (Running, 0 sec, 1 stage)
- Notification to PubSub (Running, 0 sec, 1 stage)

The right-hand panel shows "Job info" and "Resource metrics".

Job info

Job name	predict-fraudulent-transactions
Job ID	2022-05-13_18_37_41-948231955907693397
Job type	Streaming
Job status	Running
SDK version	Apache Beam Python 3.7 SDK 2.38.0
Job region	us-west1
Worker location	us-west1-a
Current workers	1
Latest worker status	Worker pool started.
Start time	May 13, 2022 at 6:37:42 PM GMT-7
Elapsed time	5 min 35 sec
Encryption type	Google-managed key

Resource metrics

Current vCPUs	2
Total vCPU time	0.167 vCPU hr
Current memory	7.5 GB
Total memory time	0.625 GB hr
Current HDD PD	30 GB
Total HDD PD time	2.5 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total streaming data processed	95.63 KB

Custom Counters (Approximate)

Counter name	Value	Step
batch_latency.ms_COUNT	10	Write pred to BigQuery/.../ParDo(BigQueryWrite)
batch_latency.ms_MAX	456	Write pred to BigQuery/.../ParDo(BigQueryWrite)
batch_latency.ms_MEAN	243	Write pred to BigQuery/.../ParDo(BigQueryWrite)
batch_latency.ms_MIN	75	Write pred to BigQuery/.../ParDo(BigQueryWrite)

Batch Processing

Via Terminal RPC:

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

Using endpoint [<https://ml.googleapis.com/>]

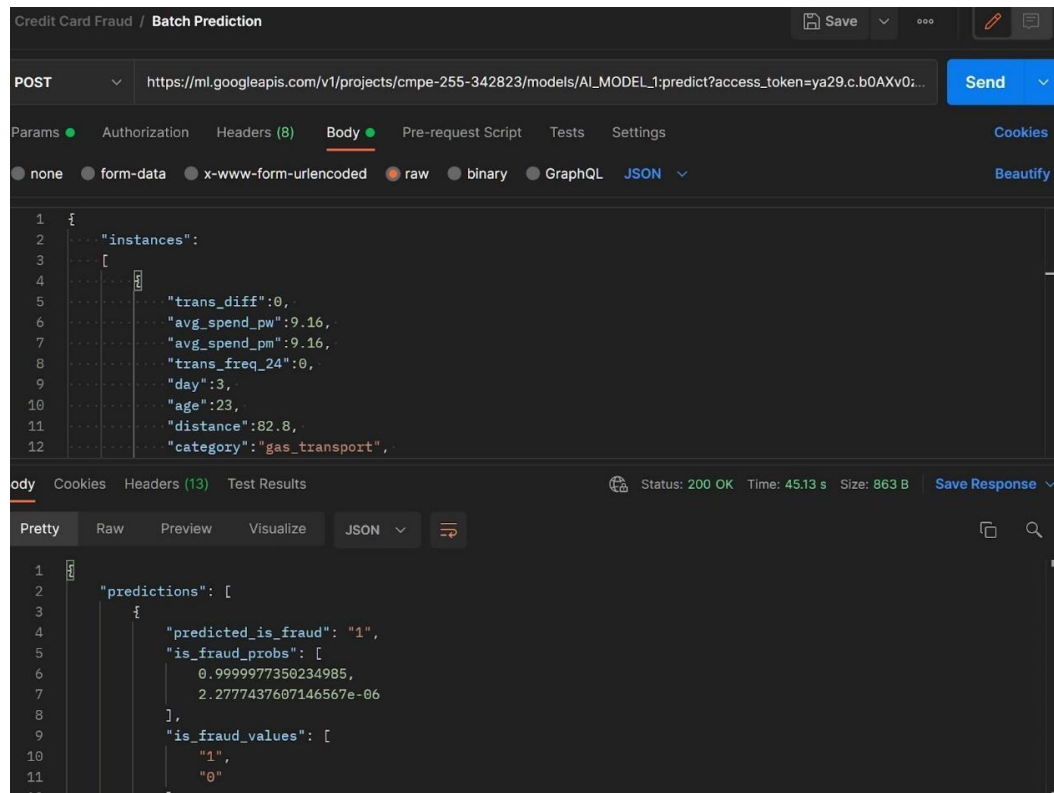
IS_FRAUD_PROBS	IS_FRAUD_VALUES	PREDICTED_IS_FRAUD
[0.9293921589851379, 0.07060789316892624]	['1', '0']	1

```
[ ] 1 !gcloud ai-platform predict --model $AI_MODEL_NAME \  
2 --version $VERSION_NAME_WITH_AGGREGATES \  
3 --region global \  
4 --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)

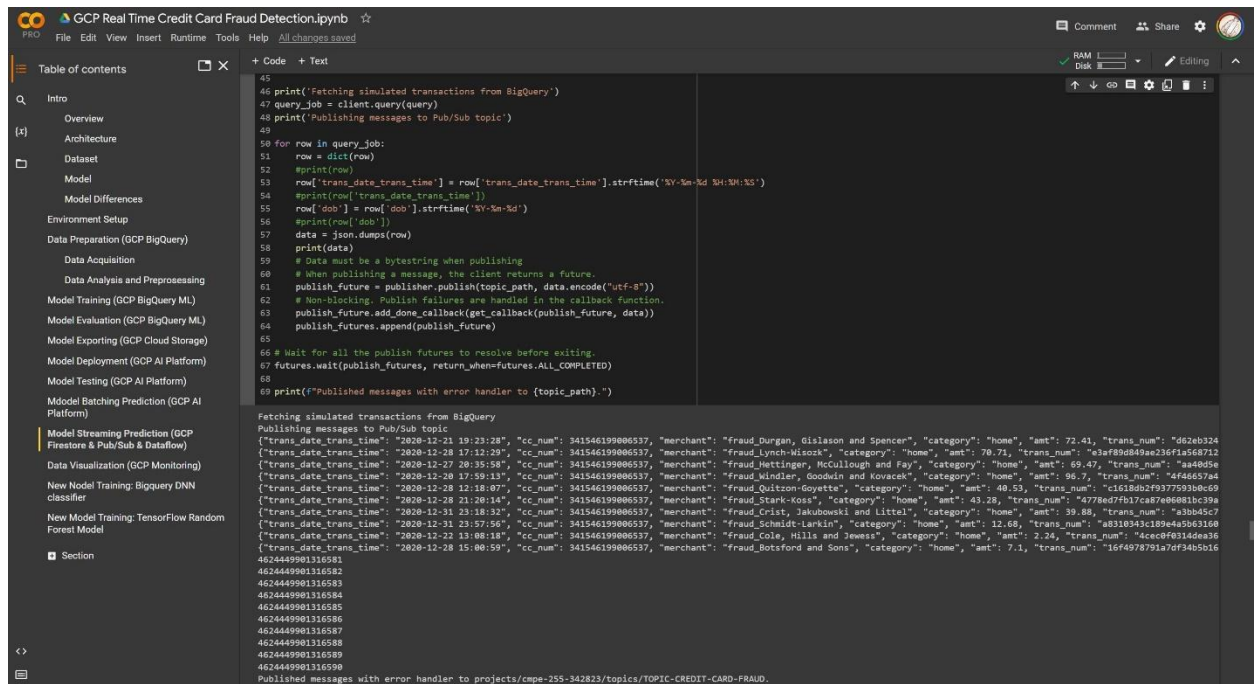
The screenshot shows the Google Cloud Platform console with a BigQuery query and its results. The query is:

```
SELECT
  trans_num, trans_date_trans_time, is_fraud_model_w_aggregates, prob_is_fraud_model_w_aggregates
FROM
  [cmpe-255-342823.BQML_CREDIT_CARD_FRAUD_REAL_TIME_TRANSACTIONS]
```

The query results are displayed in a table with the following columns: JOB INFORMATION, RESULTS, JSON, and EXECUTION DETAILS. The table contains 163 rows of data.

Row	trans_num	trans_date_trans_time	is_fraud_model_w_aggregates	prob_is_fraud_model_w_aggregates
151	7c4c482d8d1e46ceea57d552670e440c	2020-12-20 01:43:43 UTC	0	0.00048, 0.99952
152	efcc0364d173f364c3f521ccfcee11f60	2020-12-20 01:44:05 UTC	0	0.00013, 0.99987
153	7c4c482d8d1e46ceea57d552670e440c	2020-12-20 01:43:43 UTC	0	0.00048, 0.99952
154	59f1bd3bf6efed2dd16a93f6d3626c	2020-12-20 01:43:01 UTC	0	0.00026, 0.99974
155	efcc0364d173f364c3f521ccfcee11f60	2020-12-20 01:44:05 UTC	0	0.00013, 0.99987
156	31c1eccc3cd8d39a9aa9641727c49673	2020-12-20 01:43:58 UTC	0	0.00286, 0.99712
157	e4c046f1eaa84870c6c8d8fa20ea80	2020-12-20 01:43:59 UTC	0	0.01972, 0.98028
158	14392df23bb7737606b2700ac791b7aa	2020-12-31 23:59:24 UTC	0	0.00169, 0.99831
159	6d04313bfe4b661b6ca2b6a499a320fe	2020-12-31 23:58:34 UTC	0	0.00044, 0.99956
160	bd70711d5c9510a559aee196368ac80e	2020-12-31 23:58:04 UTC	0	0.00117, 0.99883
161	6c5b7c8add471975aa0fec023b2e8408	2020-12-31 23:59:15 UTC	0	0.00039, 0.99961
162	bd8f9f7d9c931682db094325097b14e	2020-12-20 01:43:56 UTC	0	0.0001, 0.9999
163	59f1bd3bf6efed2dd16a93f6d3626c	2020-12-20 01:43:01 UTC	0	0.00022, 0.99978

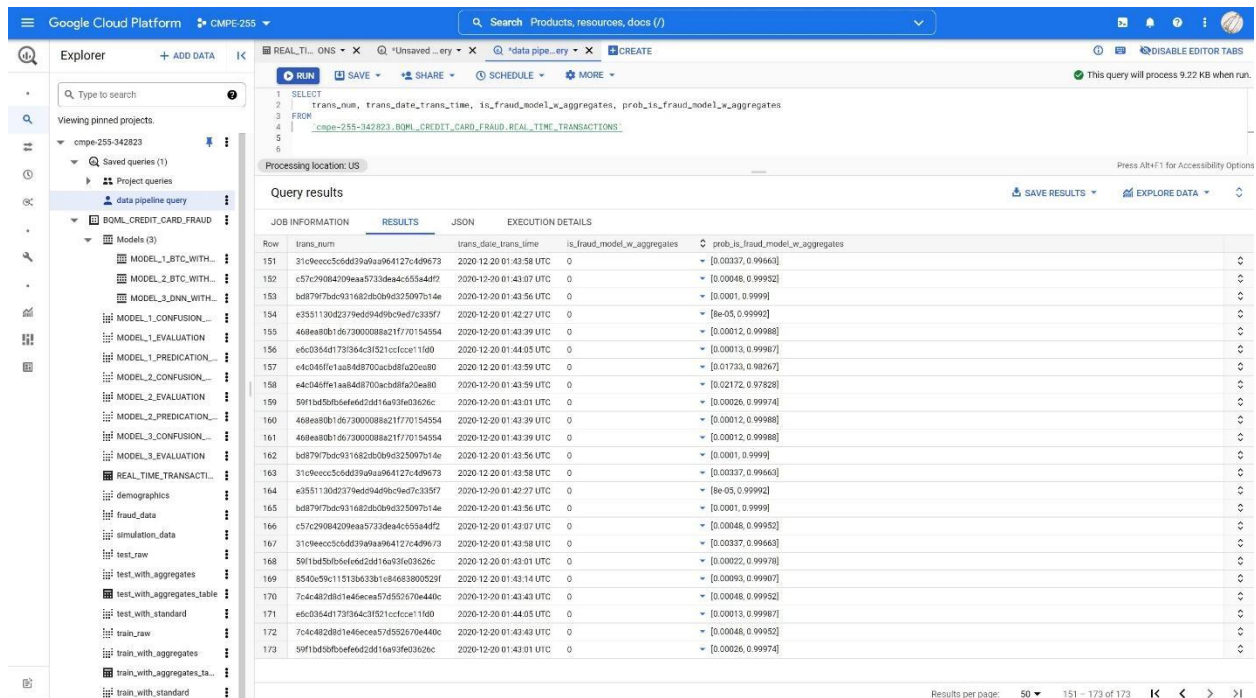
Sending transactions to the publisher (10 transactions per call...to save budget!)



```
45 print('Fetching simulated transactions from BigQuery')
46 query_job = client.query(query)
47 print('Publishing messages to Pub/Sub topic')
48
49 for row in query_job:
50     row = dict(row)
51     #print(row)
52     row['trans_date_trans_time'] = row['trans_date_trans_time'].strftime('%Y-%m-%d %H:%M:%S')
53     #print(row['trans_date_trans_time'])
54     row['dob'] = row['dob'].strftime('%Y-%m-%d')
55     #print(row['dob'])
56     data = json.dumps(row)
57     print(data)
58     # Data must be a bytestring when publishing
59     # When publishing a message, the client returns a future.
60     publish_future = publisher.publish(topic_path, data.encode('utf-8'))
61     # Non-blocking. Publish failures are handled in the callback function.
62     publish_future.add_done_callback(get_callback(publish_future, data))
63     publish_futures.append(publish_future)
64
65 # Wait for all the publish futures to resolve before exiting.
66 futures.wait(publish_futures, return_when=futures.ALL_COMPLETED)
67
68 print(f"Published messages with error handler to {topic_path}.")
```

Fetching simulated transactions from BigQuery
Publishing messages to Pub/Sub topic
{\"trans_date_trans_time\": \"2020-12-28 17:12:29\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Lynch-Wisock\", \"category\": \"home\", \"amt\": 70.71, \"trans_num\": \"e3af89d849ae236f1a568712\", \"trans_date_trans_time\": \"2020-12-27 20:35:58\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Hettinger, McCullough and Fay\", \"category\": \"home\", \"amt\": 69.47, \"trans_num\": \"aa48d5e\", \"trans_date_trans_time\": \"2020-12-28 17:59:13\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Kindler, Goodwin and Kovacek\", \"category\": \"home\", \"amt\": 98.7, \"trans_num\": \"4fd6657ad\", \"trans_date_trans_time\": \"2020-12-28 12:18:07\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Quttrub-Goyette\", \"category\": \"home\", \"amt\": 40.53, \"trans_num\": \"c1618a02f037793bb4e9\", \"trans_date_trans_time\": \"2020-12-28 21:20:14\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Stark-Koss\", \"category\": \"home\", \"amt\": 43.28, \"trans_num\": \"4778ed7fb17ca87e06081bc39a\", \"trans_date_trans_time\": \"2020-12-31 23:18:32\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Crist, Jakubowski and Little\", \"category\": \"home\", \"amt\": 39.88, \"trans_num\": \"a3bb45c7\", \"trans_date_trans_time\": \"2020-12-31 23:57:56\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Schmidt-Larkin\", \"category\": \"home\", \"amt\": 12.68, \"trans_num\": \"a0310543c109e4a0636168\", \"trans_date_trans_time\": \"2020-12-31 13:00:18\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Cole, Hillis and Jewess\", \"category\": \"home\", \"amt\": 2.24, \"trans_num\": \"4ccc0f0314de0a36\", \"trans_date_trans_time\": \"2020-12-28 15:00:59\", \"cc_num\": \"34154619906537\", \"merchant\": \"fraud_Botsford and Sons\", \"category\": \"home\", \"amt\": 7.1, \"trans_num\": \"16f4978791a7df34b5016\"}

Real-time transaction table (After sending transactions to the publisher)



Row	trans_num	trans_date_trans_time	is_fraud_model_w_aggregates	prob_is_fraud_model_w_aggregates
151	31c9eccc5c6dd39a9a94127c4d9673	2020-12-20 01:43:58 UTC	0	[0.00337, 0.99663]
152	c57c29084209aa5733deac65544df2	2020-12-20 01:43:07 UTC	0	[0.00048, 0.99952]
153	b8d797bdc931682db09d3255097b14e	2020-12-20 01:43:56 UTC	0	[0.0001, 0.9999]
154	43551130d237ed9d94dbcfed7c339f7	2020-12-20 01:42:27 UTC	0	[0.0005, 0.9995]
155	468ea81b1d673000088a21f70154554	2020-12-20 01:43:39 UTC	0	[0.00012, 0.99988]
156	e5c0354d173364c3f521ccfcee1160	2020-12-20 01:44:05 UTC	0	[0.00013, 0.99987]
157	e4c046f1eaa848700acbd8fa20ee80	2020-12-20 01:43:59 UTC	0	[0.01733, 0.98267]
158	e4c046f1eaa848700acbd8fa20ee80	2020-12-20 01:43:59 UTC	0	[0.02172, 0.97828]
159	59f1bd50b6efed62d416a93e03626c	2020-12-20 01:43:01 UTC	0	[0.00026, 0.99974]
160	468ea81b1d673000088a21f70154554	2020-12-20 01:43:39 UTC	0	[0.00012, 0.99988]
161	468ea81b1d673000088a21f70154554	2020-12-20 01:43:39 UTC	0	[0.00012, 0.99988]
162	b8d797bdc931682db09d3255097b14e	2020-12-20 01:43:56 UTC	0	[0.0001, 0.9999]
163	31c9eccc5c6dd39a9a94127c4d9673	2020-12-20 01:43:58 UTC	0	[0.00337, 0.99663]
164	43551130d237ed9d94dbcfed7c339f7	2020-12-20 01:42:27 UTC	0	[0.0005, 0.9995]
165	b8d797bdc931682db09d3255097b14e	2020-12-20 01:43:56 UTC	0	[0.0001, 0.9999]
166	c57c29084209aa5733deac65544df2	2020-12-20 01:43:07 UTC	0	[0.00048, 0.99952]
167	31c9eccc5c6dd39a9a94127c4d9673	2020-12-20 01:43:58 UTC	0	[0.00337, 0.99663]
168	59f1bd50b6efed62d416a93e03626c	2020-12-20 01:43:01 UTC	0	[0.00026, 0.99974]
169	8540e9c11513b33b1c84683800529f	2020-12-20 01:43:14 UTC	0	[0.00093, 0.99907]
170	7c4cd28d1e46eees57d552670e440c	2020-12-20 01:43:43 UTC	0	[0.00048, 0.99952]
171	e5c0354d173364c3f521ccfcee1160	2020-12-20 01:44:05 UTC	0	[0.00013, 0.99987]
172	7c4cd28d1e46eees57d552670e440c	2020-12-20 01:43:43 UTC	0	[0.00048, 0.99952]
173	59f1bd50b6efed62d416a93e03626c	2020-12-20 01:43:01 UTC	0	[0.00026, 0.99974]

References:

[How to build a fraud detection solution | Google Cloud](#)

[Real-Time Credit Card Fraud Detection | Gitlab](#)

[Credit Card Transaction Data Generator](#)

[BigQuery documentation | Google Cloud](#)

[BigQuery ML documentation | Google Cloud](#)

[Cloud Storage documentation | Google Cloud](#)

[Firestore documentation | Google Cloud](#)

[AI Platform documentation | Google Cloud](#)

[Pub/Sub documentation - Google Cloud](#)

[Dataflow documentation | Google Cloud](#)

[TensorFlow Decision Forests | TensorFlow](#)