COURSE PROJECT REPORT

CREDIT-CARD FRAUD DETECTION PIPELINE

Amit Kasera

ENGR-E516 Engineering Cloud Computing

1. Introduction (brief overview of the problem or topic, 1-2 paragraphs)

In order to prevent customers from being charged for items they did not buy, credit card companies must be able to accurately identify fraudulent credit card transactions. An excellent article by Stripe[1] highlights that companies lose more loyal customers (approximately 33%) due to blocked transactions. On the other hand, fraudulent transactions going unnoticed could lead to potential lawsuits, operational dispute fees, etc.

2. Background (why this problem or topic was selected, why it is interesting / important to address, 1-2 paragraphs)

A subtle line needs to be drawn and my main focus here is to reduce blocked-transactions of legitimate customers. We can design the Null and Alternative Hypothesis as "A transaction is legitimate until proven otherwise":

$$H_0 : The\ transaction\ is\ legitimate$$

$$H_1 : The\ transaction\ is\ fraudulent$$

A Type-II error would be a False-Negative: expected H_0 but wrongly predicted H_1. This is what we want to minimize to deliver tangible business-value.

My aim is to provide stakeholders with an automated cloud-framework that enables distinction between fraud and legitimacy, by leveraging Machine Learning. Simultaneously, the framework should allow for downstream analysis like dashboarding through effective data-management.

As such, I’ve created and deployed an end-to-end Machine Learning pipeline on GCP using Airflow as the pipeline orchestrator, and Cloud-Functions to monitor source-file changes on Google-Cloud-Storage.

The dataset is openly available as part of a Data Science challenge[7]. Here's some metadata that I've interpreted:

Column	Data-Type	Description
accountNumber	Integer	Account-Number of customer who made the transaction
customerId	Integer	
creditLimit	Integer	Current credit-limit of customer who made the transaction
availableMoney	Float	Available credit for customer
transactionDateTime	String	Timestamp of transaction
transactionAmount	Float	Amount for transaction
merchantName	String	Name of merchant (POS)
acqCountry	String	Country where credit card was acquired
merchantCountryCode	String	Country code for merchant (POS)
posEntryMode	Float	2-digit code for info on how card was entered (https://help.globalmerchantportal.com/en/code-glossary/pos-entry-modes)
posConditionCode	Float	2-digit code to identifies mode of initiation of the transactions.
merchantCategoryCode	String	General category (fast food, entertainment, etc.)
currentExpDate	String	Card expiry date
accountOpenDate	String	Account opened (Start) date
dateOfLastAddressChange	String	Last updated date of address
cardCVV	Integer	Expected CVV available on credit-card
enteredCVV	Integer	Entered CVV at POS
cardLast4Digits	Integer	Last 4 digits of Credit Card
transactionType	String	Purchase, Reversal, Address Verification, etc.
currentBalance	Float	CreditLimit - AvailableMoney
merchantCity	Float	City for POS
merchantState	Float	State for POS
merchantZip	Float	Zipcode for POS
cardPresent	Boolean	True/False for card presented in-person
posOnPremises	Float	POS on premise of store
isFraud	Boolean	Target-Label for transaction being marked as Fraud

3. Methodology (steps taken to address the problem, including the technological setup 1-2 pgs)

As with any major Machine Learning project, I started with exploring the data and creating modularized functions. I released a package on PyPi [2] wherein I modularized my code for the entire training and prediction phase. Eventually, this containerized package was used in Airflow DAGs within a `PythonVirtualenvOperator()`, on a Cloud Composer environment.

Overall, here's a visual picture of what my containerized Airflow DAGs and package accomplish:

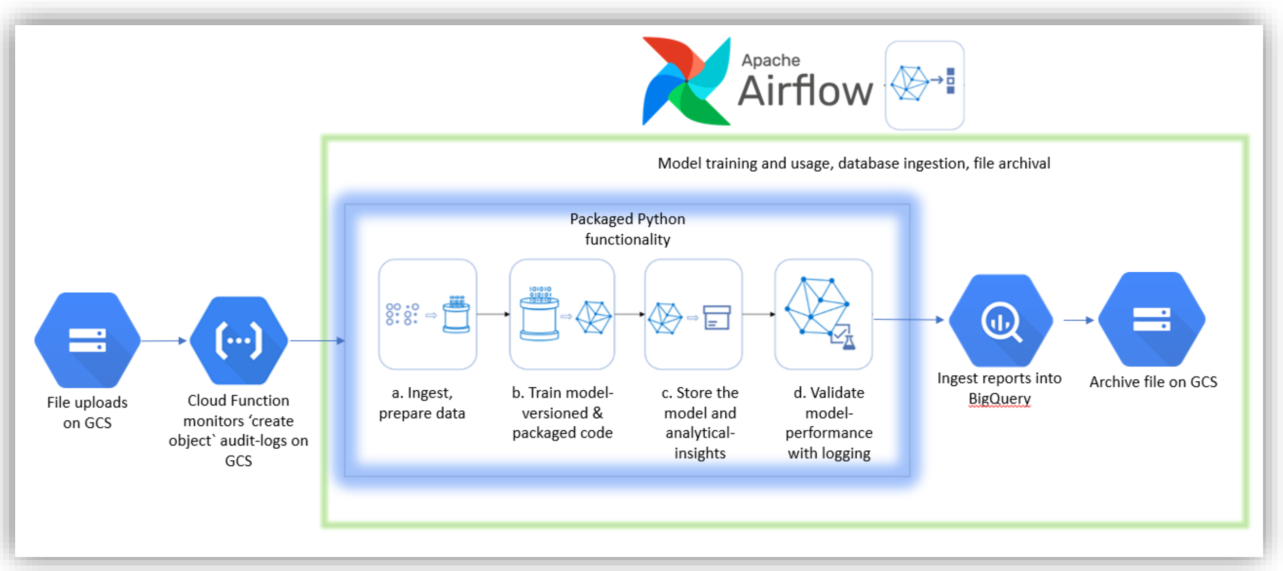

Apache Airflow
Model training and usage, database ingestion, file archival
Packaged Python functionality
File uploads on GCS
Cloud Function monitors 'create object' audit-logs on GCS
a. Ingest, prepare data
b. Train model- versioned & packaged code
c. Store the model and analytical-insights
d. Validate model-performance with logging
Ingest reports into BigQuery
Archive file on GCS

Fig.1.1. Overview of Airflow DAGs and containerized Python package

I'm maintaining my storage objects in the "e516-course-project-bucket" GCS Storage bucket. There are 2 Airflow DAGs, one for the model-training phase, and one for using the registered model to make predictions. The training-phase DAG gets triggered when a "Train_transactions.csv" file is uploaded onto Google-Cloud-Storage in the specified bucket. Similarly, for the prediction DAG but for "Test_transactions.csv".

2 Google Cloud Functions in Python: "gcs-trainfile-watcher-trigger-function", and "gcs-testfile-watcher-trigger-function", perform the source-file-upload monitoring and eventual Airflow DAG triggering activity.

The following 4 points should help summarize the package quickly:

a. Ingest and prepare data:

A series of data-cleansing, standardization, scaling, and feature-engineering steps take place:

1. **read_src_file_as_df()-** The Airflow task reads the "Train/Test_transactions.csv" file from GCS, and converts it into a dataframe.

2. **add_time_dependent_features()-** For the input datetime columns, engineer new date and time features. Also drop out the original column.
3. **levenshtein_distance()-** Rather than just binary non-equality, engineer a score of mismatch for enteredCVV vs cardCVV: a single digit wrong might be just a minor blunder.
4. **drop_unary_columns()-** Filter out features with <=1 unique values.
5. **get_reversals_report()-** Create a report of reversal transactions.
6. **get_multiswipe_transactions()-** Create a report of multiswipe transactions.
7. **convert_boolean_to_int()-** Convert Boolean-columns in the data to integer encodings.
8. **encode_categorical_cols()-** Apply Ordinal-Encoding to each of categorical-columns and keep track of the transformations, ie- store the sklearn models as pickle objects on GCS for use during prediction.
9. **scaledown_numerical_cols()-** Apply MinMax-Scaling to each of numerical-columns and keep track of the transformations, ie- store the sklearn models as pickle objects on GCS for use during prediction.
10. **drop_irrelevant_columns()-** Filters out the non-numerical columns from data.

b. Train model- versioned and packaged code:

1. **encode_categorical_cols()**, **scaledown_numerical_cols()**- Train Ordinal-Encoder and MinMax-Scaler sklearn models for standardizing/transforming data.
2. **train_random_forest_classifier()**- Apply grid-search-cv for a random-forest-classifier.

c. Store the model and analytical insights:

1. **encode_categorical_cols()**, **scaledown_numerical_cols(), and train_random_forest_classifier()**- Store the respective models as pickle objects on GCS for use during batch-prediction.
2. The **model.predict()** outputs are stored in GCS for the train and test phase as "Random_Forest_Validation_Set_predictions.csv" and "Random_Forest_Test_Set_predictions.csv" respectively.
3. The analytical csv-reports from **get_reversals_report()** and **get_multiswipe_transactions()**, "Overall_Report_Reversals.csv", "Overall_Report_Multiswipes.csv" are later ingested into BigQuery as a Dataset**.**

d. Validate model-performance with logging:

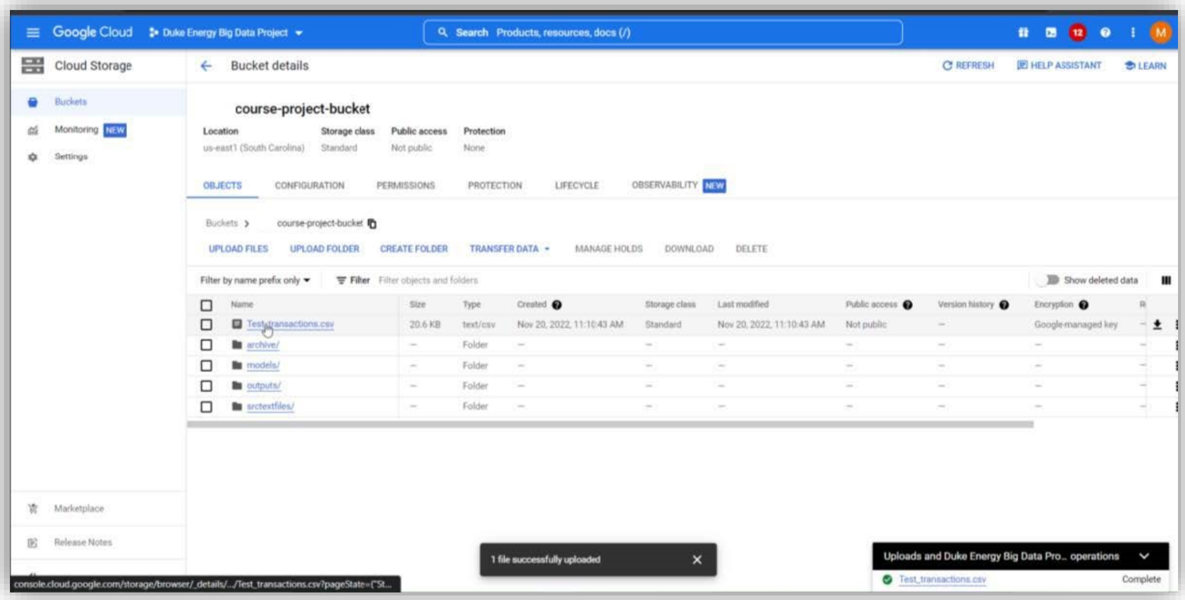
The package supports efficient logging within a containerized environment (configurable verbosity of logs), with appropriate try-catch blocks and a **print_neat_metrics()** to evaluate model performance after each training-phase execution.

The next Airflow task is for ingestion of the csv reports currently on GCS, into BigQuery. I've also added in a logic to capture source date and last updated date changes, to mimic a data-warehouse SCD-Type 2.

Finally, the source CSV files are archived into a folder on the same bucket's "/archive/" folder. Their file-names are appended with the date when the archival happened ie- the date of being processed.

4. Results (1-3 pgs)

A working demonstration can be found on the wikis of my Github Repository: here. Nevertheless, the following screenshots should showcase the algorithmic flow of tasks.



course-project-bucket

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	None

Buckets > course-project-bucket

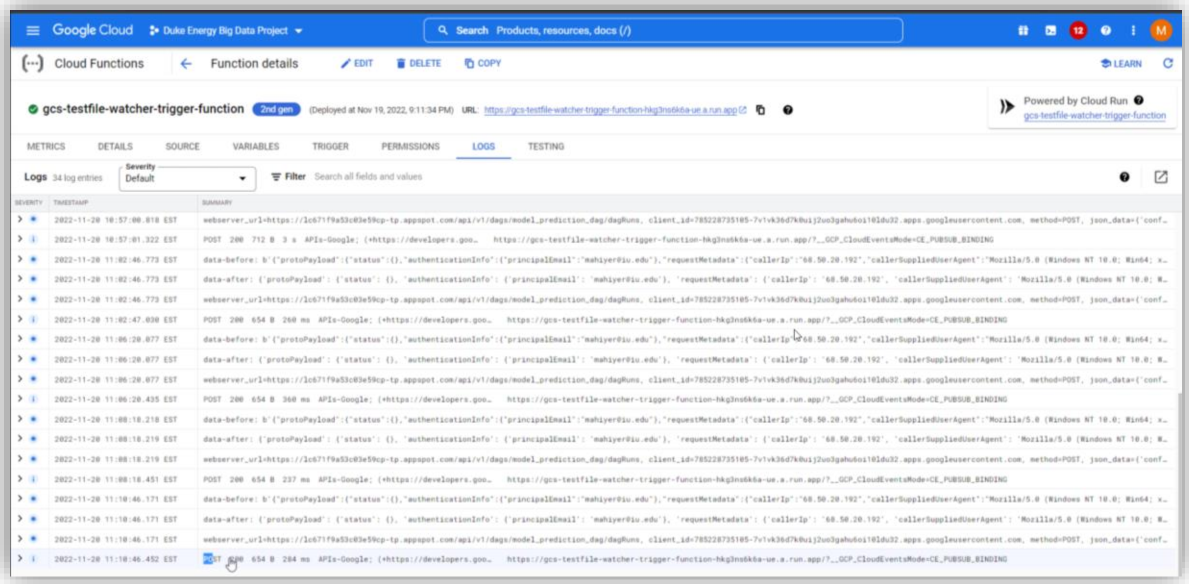
Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
Test_transactions.csv	20.6 KB	text/csv	Nov 20, 2022, 11:10:43 AM	Standard	Nov 20, 2022, 11:10:43 AM	Not public	–	Google-managed key
archive/	–	Folder	–	–	–	–	–	–
models/	–	Folder	–	–	–	–	–	–
outputs/	–	Folder	–	–	–	–	–	–
srctextfiles/	–	Folder	–	–	–	–	–	–

1 file successfully uploaded

Uploads and Duke Energy Big Data Pro... operations
Test_transactions.csv Complete

console.cloud.google.com/storage/browser/_details/.../Test_transactions.csv?pageState=("St...

Fig.4.1. File upload onto GCS



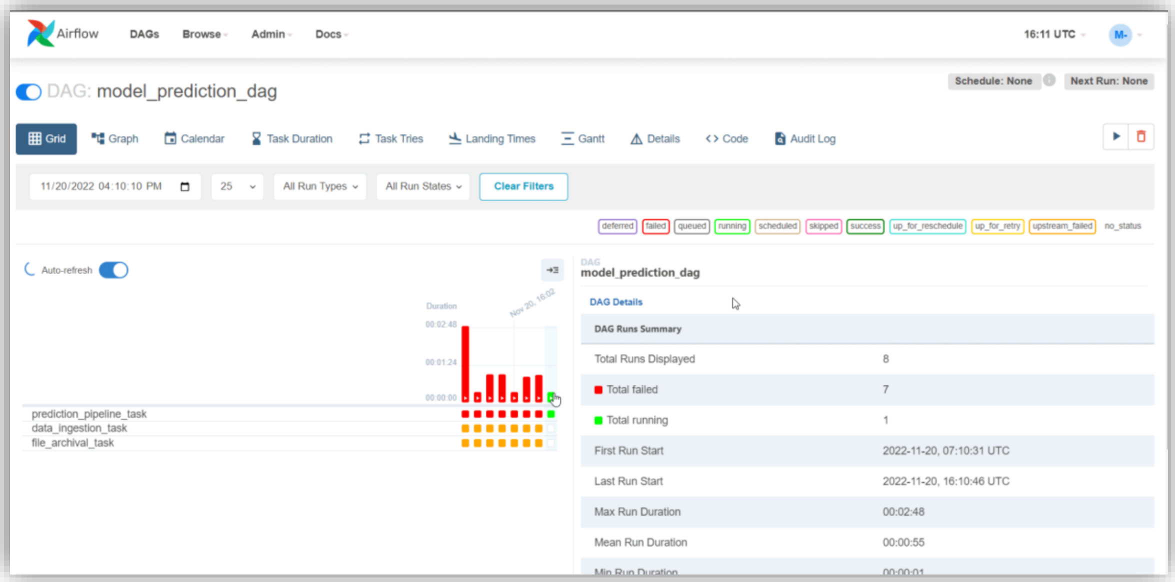
gcs-testfile-watcher-trigger-function 2nd gen (Deployed at Nov 19, 2022, 9:11:34 PM) URL: https://gcs-testfile-watcher-trigger-function-hkg3ns6k6a-ue.a.run.app

Powered by Cloud Run
gcs-testfile-watcher-trigger-function

Logs 34 log entries
Severity: Default

```
SEVERITY  TIMESTAMP                    SUMMARY
          2022-11-20 10:57:00.818 EST  webserver_url=https://1c671f9a53c03e59cp-tp.appspot.com/api/v1/dags/model_prediction_dag/dagRuns, client_id=785228735105-7v1vk36d7k0uij2uo3gahu6oi10ldu32.apps.googleusercontent.com, method=POST, json_data={'conf…
          2022-11-20 10:57:01.322 EST  POST  200  712 B  3 s  APIs-Google; (+https://developers.goo…  https://gcs-testfile-watcher-trigger-function-hkg3ns6k6a-ue.a.run.app/?__GCP_CloudEventsMode=CE_PUBSUB_BINDING
          2022-11-20 11:02:46.773 EST  data-before: b'{"protoPayload":{"status":{},"authenticationInfo":{"principalEmail":"mahiyer@iu.edu"},"requestMetadata":{"callerIp":"68.50.20.192","callerSuppliedUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x…
          2022-11-20 11:02:46.773 EST  data-after: {'protoPayload': {'status': {}, 'authenticationInfo': {'principalEmail': 'mahiyer@iu.edu'}, 'requestMetadata': {'callerIp': '68.50.20.192', 'callerSuppliedUserAgent': 'Mozilla/5.0 (Windows NT 10.0; W…
          2022-11-20 11:02:46.773 EST  webserver_url=https://1c671f9a53c03e59cp-tp.appspot.com/api/v1/dags/model_prediction_dag/dagRuns, client_id=785228735105-7v1vk36d7k0uij2uo3gahu6oi10ldu32.apps.googleusercontent.com, method=POST, json_data={'conf…
          2022-11-20 11:02:47.030 EST  POST  200  654 B  260 ms  APIs-Google; (+https://developers.goo…  https://gcs-testfile-watcher-trigger-function-hkg3ns6k6a-ue.a.run.app/?__GCP_CloudEventsMode=CE_PUBSUB_BINDING
          2022-11-20 11:06:20.077 EST  data-before: b'{"protoPayload":{"status":{},"authenticationInfo":{"principalEmail":"mahiyer@iu.edu"},"requestMetadata":{"callerIp":"68.50.20.192","callerSuppliedUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x…
          2022-11-20 11:06:20.077 EST  data-after: {'protoPayload': {'status': {}, 'authenticationInfo': {'principalEmail': 'mahiyer@iu.edu'}, 'requestMetadata': {'callerIp': '68.50.20.192', 'callerSuppliedUserAgent': 'Mozilla/5.0 (Windows NT 10.0; W…
          2022-11-20 11:06:20.077 EST  webserver_url=https://1c671f9a53c03e59cp-tp.appspot.com/api/v1/dags/model_prediction_dag/dagRuns, client_id=785228735105-7v1vk36d7k0uij2uo3gahu6oi10ldu32.apps.googleusercontent.com, method=POST, json_data={'conf…
          2022-11-20 11:06:20.435 EST  POST  200  654 B  360 ms  APIs-Google; (+https://developers.goo…  https://gcs-testfile-watcher-trigger-function-hkg3ns6k6a-ue.a.run.app/?__GCP_CloudEventsMode=CE_PUBSUB_BINDING
          2022-11-20 11:08:18.218 EST  data-before: b'{"protoPayload":{"status":{},"authenticationInfo":{"principalEmail":"mahiyer@iu.edu"},"requestMetadata":{"callerIp":"68.50.20.192","callerSuppliedUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x…
          2022-11-20 11:08:18.219 EST  data-after: {'protoPayload': {'status': {}, 'authenticationInfo': {'principalEmail': 'mahiyer@iu.edu'}, 'requestMetadata': {'callerIp': '68.50.20.192', 'callerSuppliedUserAgent': 'Mozilla/5.0 (Windows NT 10.0; W…
          2022-11-20 11:08:18.219 EST  webserver_url=https://1c671f9a53c03e59cp-tp.appspot.com/api/v1/dags/model_prediction_dag/dagRuns, client_id=785228735105-7v1vk36d7k0uij2uo3gahu6oi10ldu32.apps.googleusercontent.com, method=POST, json_data={'conf…
          2022-11-20 11:08:18.451 EST  POST  200  654 B  237 ms  APIs-Google; (+https://developers.goo…  https://gcs-testfile-watcher-trigger-function-hkg3ns6k6a-ue.a.run.app/?__GCP_CloudEventsMode=CE_PUBSUB_BINDING
          2022-11-20 11:10:46.171 EST  data-before: b'{"protoPayload":{"status":{},"authenticationInfo":{"principalEmail":"mahiyer@iu.edu"},"requestMetadata":{"callerIp":"68.50.20.192","callerSuppliedUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x…
          2022-11-20 11:10:46.171 EST  data-after: {'protoPayload': {'status': {}, 'authenticationInfo': {'principalEmail': 'mahiyer@iu.edu'}, 'requestMetadata': {'callerIp': '68.50.20.192', 'callerSuppliedUserAgent': 'Mozilla/5.0 (Windows NT 10.0; W…
          2022-11-20 11:10:46.171 EST  webserver_url=https://1c671f9a53c03e59cp-tp.appspot.com/api/v1/dags/model_prediction_dag/dagRuns, client_id=785228735105-7v1vk36d7k0uij2uo3gahu6oi10ldu32.apps.googleusercontent.com, method=POST, json_data={'conf…
          2022-11-20 11:10:46.452 EST  POST  200  654 B  284 ms  APIs-Google; (+https://developers.goo…  https://gcs-testfile-watcher-trigger-function-hkg3ns6k6a-ue.a.run.app/?__GCP_CloudEventsMode=CE_PUBSUB_BINDING
```

Fig.4.2. Cloud Function makes POST-request to Cloud Composer (Airflow) after file-upload



DAG Runs Summary

Total Runs Displayed	8
🟥 Total failed	7
🟩 Total running	1
First Run Start	2022-11-20, 07:10:31 UTC
Last Run Start	2022-11-20, 16:10:46 UTC
Max Run Duration	00:02:48
Mean Run Duration	00:00:55
Min Run Duration	00:00:01

Fig.4.3. New Airflow run triggered due to POST-request from Cloud Function

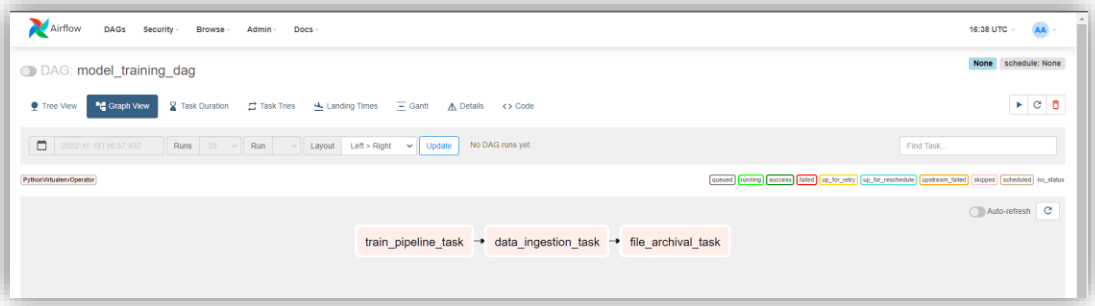


Fig. 4.4. Airflow DAG to train and register a model, ingest data into BigQuery, and archive files on a GCS bucket

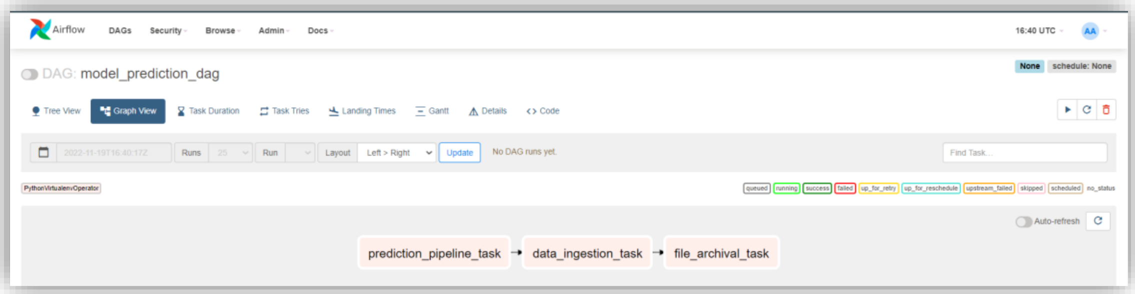
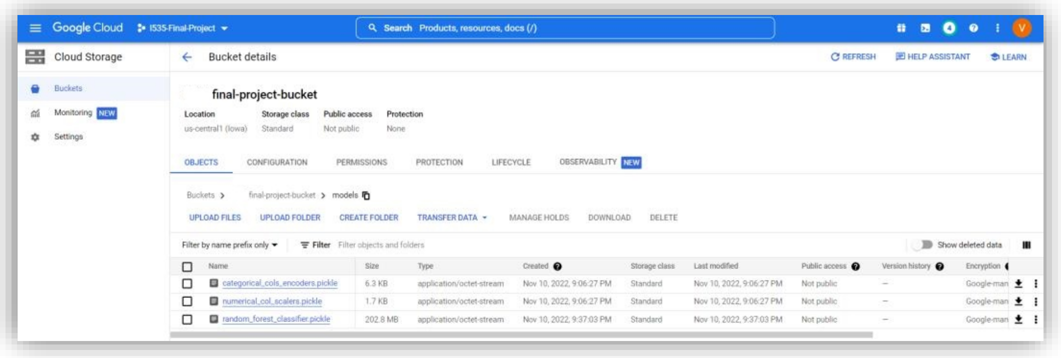


Fig.4.5. Airflow DAG to use registered-model for batch prediction on test-set, ingest data into BigQuery, and archive files on a GCS bucket



final-project-bucket

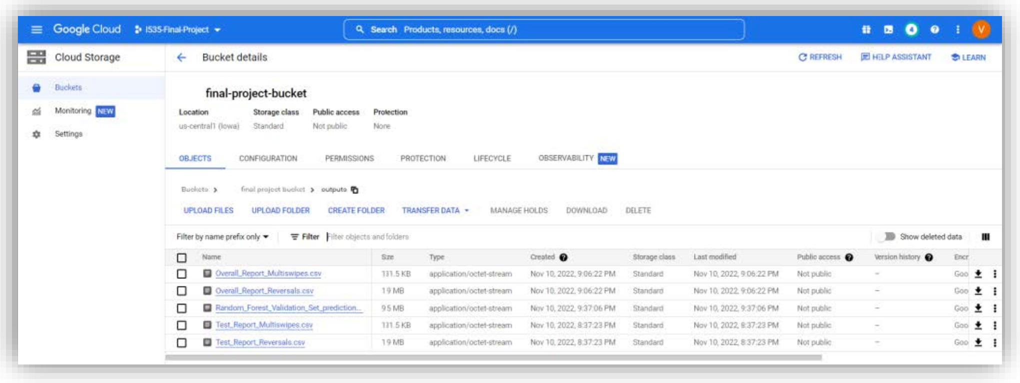
Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Not public	None

Buckets > final-project-bucket > models

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
categorical_cols_encoders.pickle	6.3 KB	application/octet-stream	Nov 10, 2022, 9:06:27 PM	Standard	Nov 10, 2022, 9:06:27 PM	Not public	–	Google-man
numerical_col_scalers.pickle	1.7 KB	application/octet-stream	Nov 10, 2022, 9:06:27 PM	Standard	Nov 10, 2022, 9:06:27 PM	Not public	–	Google-man
random_forest_classifier.pickle	202.8 MB	application/octet-stream	Nov 10, 2022, 9:37:03 PM	Standard	Nov 10, 2022, 9:37:03 PM	Not public	–	Google-man

Fig.4.5. Models are cached in GCS bucket's */models/* directory

Ideally, the model objects would be cached into a Model Registry like Vertex-AI on GCP. However, in the interest of time, I chose to simply cache and retrieve these pickled objects from GCS itself.



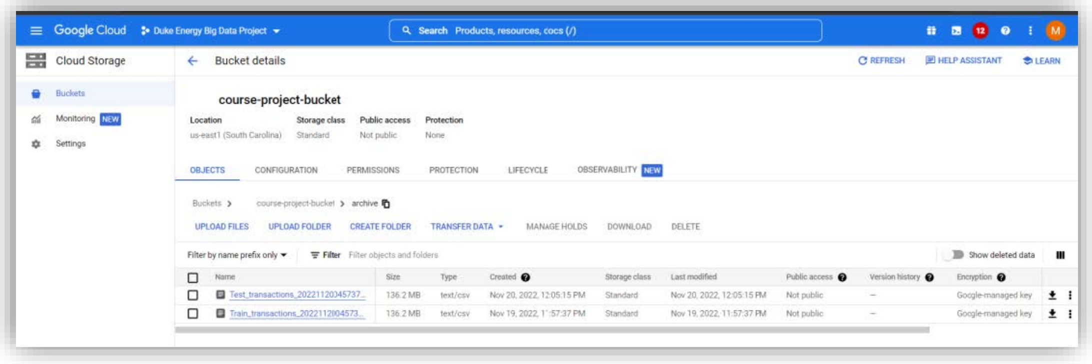
final-project-bucket

Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Not public	None

Buckets > final project bucket > outputs

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encr
Overall_Report_Multiswipes.csv	131.5 KB	application/octet-stream	Nov 10, 2022, 9:06:22 PM	Standard	Nov 10, 2022, 9:06:22 PM	Not public	–	Goo
Overall_Report_Reversals.csv	19 MB	application/octet-stream	Nov 10, 2022, 9:06:22 PM	Standard	Nov 10, 2022, 9:06:22 PM	Not public	–	Goo
Random_Forest_Validation_Set_prediction...	9.5 MB	application/octet-stream	Nov 10, 2022, 9:37:06 PM	Standard	Nov 10, 2022, 9:37:06 PM	Not public	–	Goo
Test_Report_Multiswipes.csv	131.5 KB	application/octet-stream	Nov 10, 2022, 8:37:23 PM	Standard	Nov 10, 2022, 8:37:23 PM	Not public	–	Goo
Test_Report_Reversals.csv	19 MB	application/octet-stream	Nov 10, 2022, 8:37:23 PM	Standard	Nov 10, 2022, 8:37:23 PM	Not public	–	Goo

Fig.4.6. Analytical reports captured and stored within GCS bucket's */outputs/* directory.
The next Airflow Task loads these into BigQuery tables.



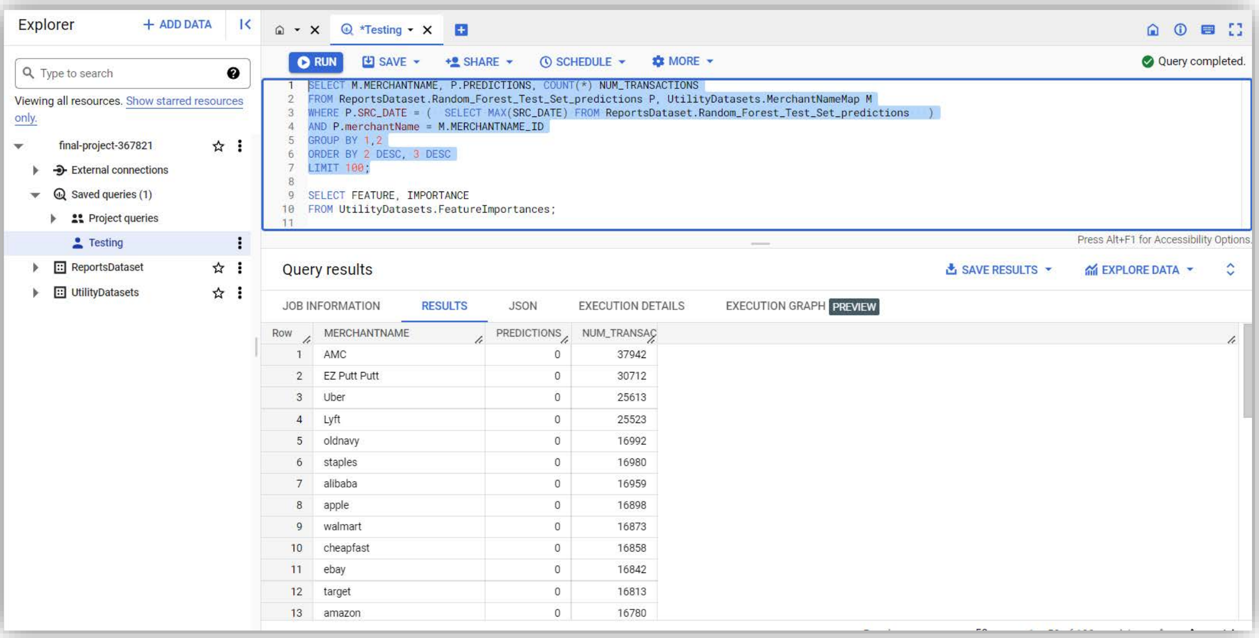
course-project-bucket

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	None

Buckets > course-project-bucket > archive

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
Test_transactions_20221120045737...	136.2 MB	text/csv	Nov 20, 2022, 12:05:15 PM	Standard	Nov 20, 2022, 12:05:15 PM	Not public	–	Google-managed key
Train_transactions_2022112004573...	136.2 MB	text/csv	Nov 19, 2022, 11:57:37 PM	Standard	Nov 19, 2022, 11:57:37 PM	Not public	–	Google-managed key

Fig.4.6. Final Airflow task archives source files in GCS bucket again



```
SELECT M.MERCHANTNAME, P.PREDICTIONS, COUNT(*) NUM_TRANSACTIONS
FROM ReportsDataset.Random_Forest_Test_Set_predictions P, UtilityDatasets.MerchantNameMap M
WHERE P.SRC_DATE = (  SELECT MAX(SRC_DATE) FROM ReportsDataset.Random_Forest_Test_Set_predictions   )
AND P.merchantName = M.MERCHANTNAME_ID
GROUP BY 1,2
ORDER BY 2 DESC, 3 DESC
LIMIT 100;

SELECT FEATURE, IMPORTANCE
FROM UtilityDatasets.FeatureImportances;
```

Query results

Row	MERCHANTNAME	PREDICTIONS	NUM_TRANSAC
1	AMC	0	37942
2	EZ Putt Putt	0	30712
3	Uber	0	25613
4	Lyft	0	25523
5	oldnavy	0	16992
6	staples	0	16980
7	alibaba	0	16959
8	apple	0	16898
9	walmart	0	16873
10	cheapfast	0	16858
11	ebay	0	16842
12	target	0	16813
13	amazon	0	16780

Fig.4.7. Data Management allows for downstream analytics and dashboarding

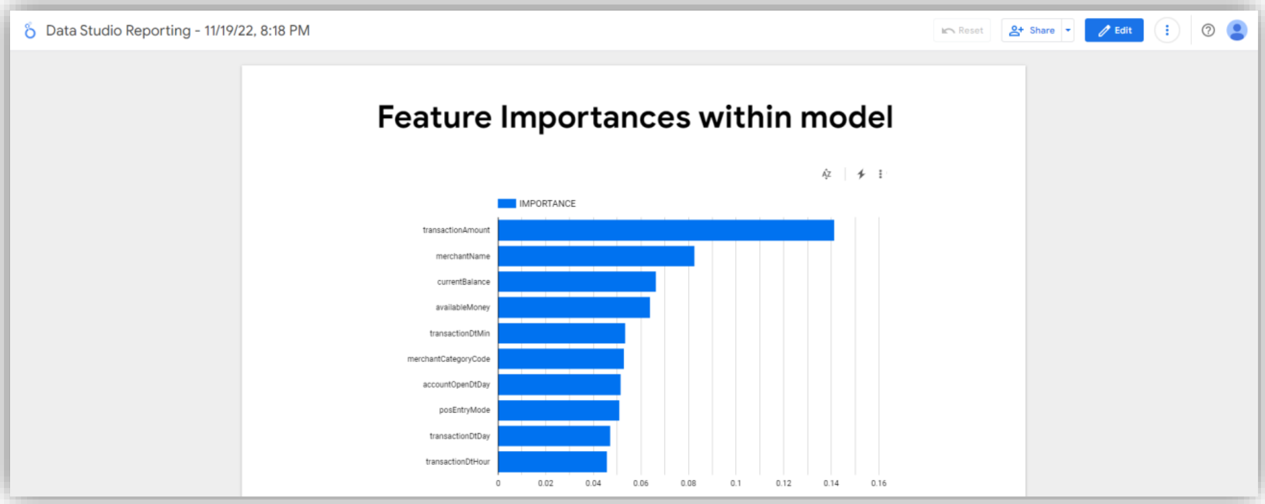
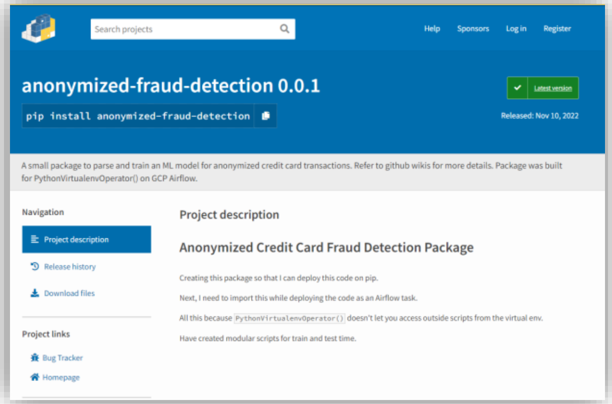


Fig.4.8. Data Studio dashboard of latest model performance ingested via pipeline, visible here

5. Discussion (an interpretation of the results + a discussion of how you employed the technologies / skills from this course + any barriers or failures you encountered, 1-2 pgs)

The steps to deploy the Airflow DAGs and GCS-monitoring functions may sound simple, but I ran into multiple challenges while trying to move my codebase from independent components, into the cloud. While deploying my code within Airflow, I realized that the `PythonVirtualenvOperator()` has some limitations in terms of not being able to read anything outside of its own local containerized scope. Hence it was necessary to create my own package[2], publish it on the official PyPi server, and then use the modular components within the Airflow tasks- a great learning experience!



anonymized-fraud-detection 0.0.1

```
pip install anonymized-fraud-detection
```

Released: Nov 10, 2022

A small package to parse and train an ML model for anonymized credit card transactions. Refer to github wikis for more details. Package was built for PythonVirtualenvOperator() on GCP Airflow.

Project description

Anonymized Credit Card Fraud Detection Package

Creating this package so that I can deploy this code on pip.

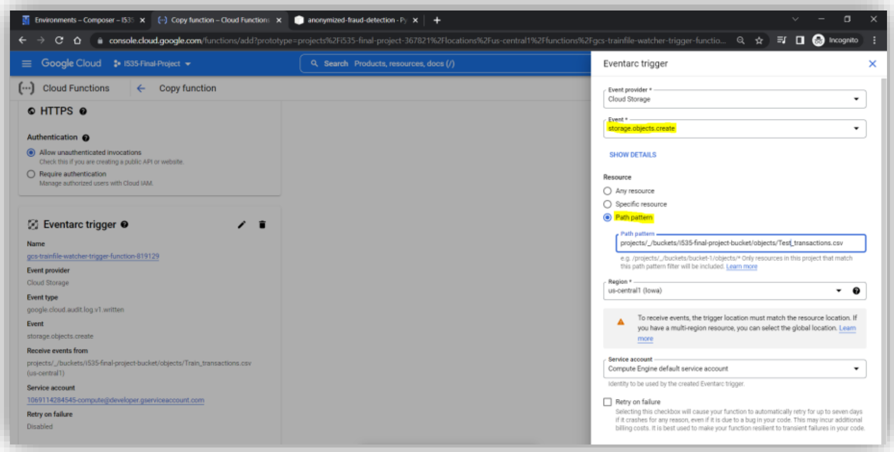
Next, I need to import this while deploying the code as an Airflow task.

All this because `PythonVirtualenvOperator()` doesn't let you access outside scripts from the virtual env.

Have created modular scripts for train and test time.

Fig.1.1. Published own custom package- necessary for use within PythonVirtualenvOperator()

Similarly while creating the cloud functions to monitor source-file changes, I ran into a bunch of issues due to lack of up-to-date documentation of GCP EventArc Triggers. Mainly, there are scarce documented examples[3] on how to configure Path patterns for monitoring specific file-creation events on a bucket. I managed to figure out the right configuration after 3 days of rigorous experimentation- a great learning experience!


Environments – Composer – I535
Copy function – Cloud Functions
anonymized-fraud-detection - Py
Incognito
Google Cloud
I535-Final-Project
Search Products, resources, docs (/)
Cloud Functions
Copy function
SHOW DETAILS
Learn more

console.cloud.google.com/functions/add?prototype=projects%2Fi535-final-project-367821%2Flocations%2Fus-central1%2Ffunctions%2Fgcs-trainfile-watcher-trigger-functio...

HTTPS

Authentication

[✓ Allow unauthenticated invocations] Check this if you are creating a public API or website.

Require authentication Manage authorized users with Cloud IAM.

Eventarc trigger

Name: gcs-trainfile-watcher-trigger-function-819129

Event provider: Cloud Storage

Event type: google.cloud.audit.log.v1.written

Event: storage.objects.create

Receive events from: projects/_/buckets/i535-final-project-bucket/objects/Train_transactions.csv (us-central1)

Service account: 1069114284545-compute@developer.gserviceaccount.com

Retry on failure: Disabled

Eventarc trigger

Event provider *: Cloud Storage

Event *: storage.objects.create

Resource: Any resource / Specific resource / [✓ Path pattern]

Path pattern: projects/_/buckets/i535-final-project-bucket/objects/Test_transactions.csv

e.g. /projects/_/buckets/bucket-1/objects/* Only resources in this project that match this path pattern filter will be included. Learn more

Region *: us-central1 (Iowa)

To receive events, the trigger location must match the resource location. If you have a multi-region resource, you can select the global location. Learn more

Service account: Compute Engine default service account

Identity to be used by the created Eventarc trigger.

Retry on failure

Selecting this checkbox will cause your function to automatically retry for up to seven days if it crashes for any reason, even if it is due to a bug in your code. This may incur additional billing costs. It is best used to make your function resilient to transient failures in your code.

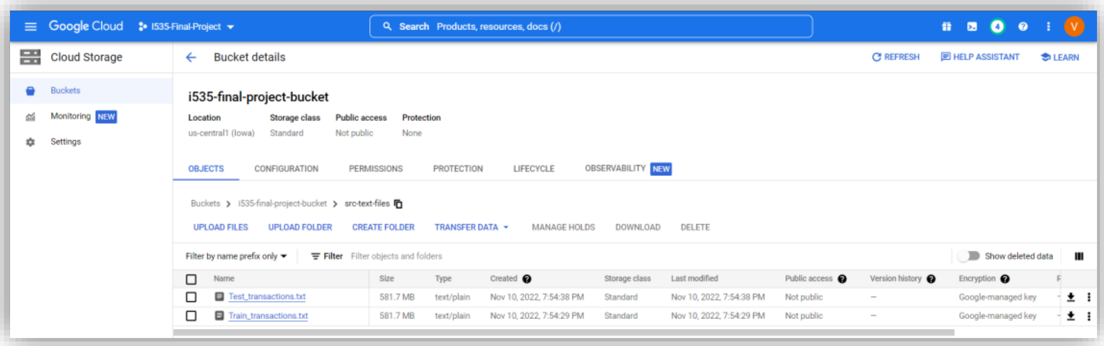
Fig.1.1. Cloud function configuration to monitor specific files on bucket

Other issues I encountered were:

1. Unable to setup connection with GCS bucket while reading and writing files.
2. Unable to store model pickle files on GCS bucket.
3. Unable to read the 580mb source text file due to limitations of compute resources (for some reason I'm unable to allocate a "Large" Cloud Composer environment).
4. Cloud Function trigger errors, due to function-"context" received in bytes rather than dictionary.
5. Cloud Function unable to communicate with Composer v1.

Eventually, I managed to fix these errors (except #3) and got my pipeline up and running!

The workaround I used for #3 was to convert the 580mb TXT file into a 360mb .csv file, and things worked fine. I didn't personally do any compression, the CSV format was just inherently more efficient in storage than the TXT format.



i535-final-project-bucket

Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Not public	None

Buckets > i535-final-project-bucket > src-text-files

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
Test_transactions.txt	581.7 MB	text/plain	Nov 10, 2022, 7:54:38 PM	Standard	Nov 10, 2022, 7:54:38 PM	Not public	–	Google-managed key
Train_transactions.txt	581.7 MB	text/plain	Nov 10, 2022, 7:54:29 PM	Standard	Nov 10, 2022, 7:54:29 PM	Not public	–	Google-managed key

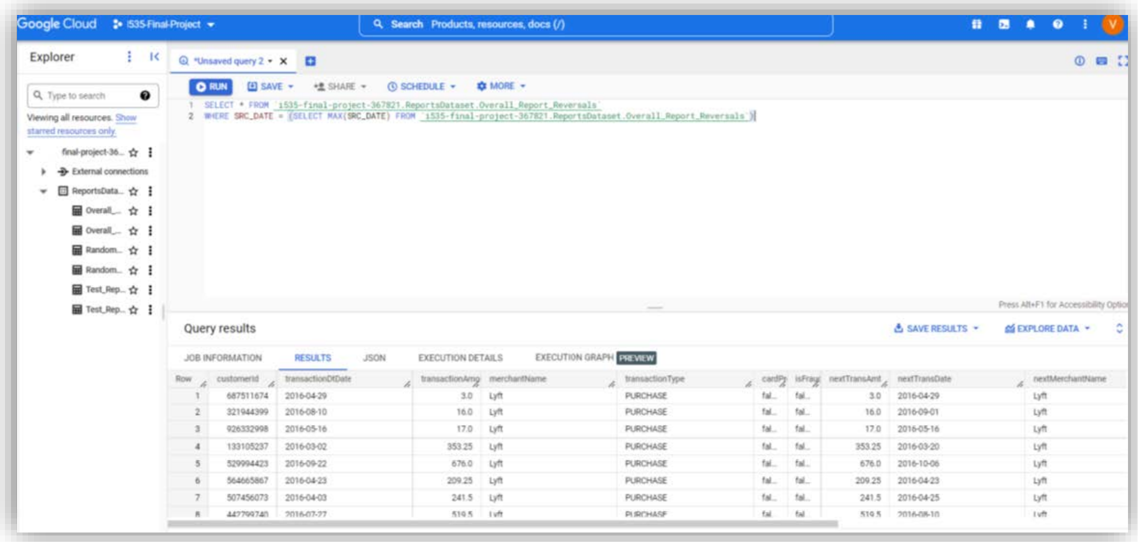
Fig.1.1. Ideally, .txt should be ingested but due to IAM access constraints, had to resort to CSV files

The workaround for #5 was frustrating because GKE services were going down around 15th November to 22nd November (autoscaling is a feature for Composer v2). Hence, I had to resort to Composer v1, but the services were still unable to communicate. After hours of searching, the solution was to make sure the Cloud Composer v1 environment has this variable configured[5]:

`api.composer_auth_user_registration_role = Op`

This basically ensures that anyone can make REST-requests to the Airflow server API.

Additionally, I also had to come up with a data management strategy, hence I decided to implement a file-archival step, as well as using a timestamp to indicate creation-date in the database table. An advanced implementation would use last-updated-date if we're not looking to completely overwrite each row (SCD Type 2 table). For now, the data is ingested into the table as a fresh new load.



```sql
SELECT * FROM `i535-final-project-367821.ReportsDataset.Overall_Report_Reversals`
WHERE SRC_DATE = (SELECT MAX(SRC_DATE) FROM `i535-final-project-367821.ReportsDataset.Overall_Report_Reversals`)
```

Row	customerId	transactionDtDate	transactionAmo	merchantName	transactionType	cardPr	isFrau	nextTransAmt	nextTransDate	nextMerchantName
1	687511674	2016-04-29	3.0	Lyft	PURCHASE	fal...	fal...	3.0	2016-04-29	Lyft
2	321944399	2016-08-10	16.0	Lyft	PURCHASE	fal...	fal...	16.0	2016-09-01	Lyft
3	926332998	2016-05-16	17.0	Lyft	PURCHASE	fal...	fal...	17.0	2016-05-16	Lyft
4	133105237	2016-03-02	353.25	Lyft	PURCHASE	fal...	fal...	353.25	2016-03-20	Lyft
5	529994423	2016-09-22	676.0	Lyft	PURCHASE	fal...	fal...	676.0	2016-10-06	Lyft
6	564665867	2016-04-23	209.25	Lyft	PURCHASE	fal...	fal...	209.25	2016-04-23	Lyft
7	507456073	2016-04-03	241.5	Lyft	PURCHASE	fal...	fal...	241.5	2016-04-25	Lyft
8	442799740	2016-07-27	519.5	Lyft	PURCHASE	fal...	fal...	519.5	2016-08-10	Lyft

Fig.1.1. BigQuery tables uses *src_date* to mark active records

6. Conclusion (1 paragraph)

In conclusion, the project incorporated data management, data cleaning/wrangling, reporting, software engineering, containerization, deployment, and most importantly, using Github to maintain README tutorials, Wikis, documentation, and versioning of code.
Versioning of models/data is made possible on the Google Cloud Storage bucket itself, and historical-datasets can be accessed on BigQuery using the Source-date field that distinguishes between active/inactive records.

Additionally, I'd enabled my entire Airflow pipeline to run on a Google Kubernetes Engine cluster using the Cloud Composer v2 service. This service supports autoscaling, load balancing, and distributed processing for Airflow tasks. However, in the last week of the course, this service was intermittently down, so I had to resort to Cloud Composer v1 (without autoscaling).

All in all, a really great and enjoyable learning experience!

7. References

1. https://stripe.com/guides/primer-on-machine-learning-for-fraud-protection
2. https://pypi.org/project/anonymized-fraud-detection/
3. https://medium.com/google-cloud/applying-a-path-pattern-when-filtering-in-eventarc-f06b937b4c34
4. https://cloud.google.com/composer/docs/how-to/using/triggering-with-gcf
5. https://cloud.google.com/composer/docs/access-airflow-api