



Deep Learning for Fraud Detection using GCP



Global IT Security Summit
5th Edition





Table of Contents

- 1- Fraud and it's Cost
- 2- Fighting Fraud with AI and Cloud
- 3- The Datasets we have used
- 4- Deep Learning models for Fraud Detection
- 5- Leveraging AI models with GCP
- 6- Conclusion and Perspectives

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

The screenshot displays the Google Cloud Platform (GCP) console interface. The top navigation bar shows the GCP logo, the account ID '03-e6fec888554c', and a search bar containing 'cloud storage'. The left sidebar contains navigation links for Cloud Storage, Browser, Monitoring, and Settings. The main content area is titled 'Object details' and shows the path 'Buckets > 03-e6fec888554c > PS_20174392719_1491204439457_log.csv'. Below the path are tabs for 'LIVE OBJECT' and 'VERSION HISTORY'. Action buttons for 'DOWNLOAD', 'EDIT METADATA', 'EDIT PERMISSIONS', and 'DELETE' are visible. The 'Overview' section contains a table with the following details:

Type	text/csv
Size	470.7 MB
Created	Oct 19, 2021, 11:52:04 AM
Last modified	Oct 19, 2021, 11:52:04 AM
Storage class	Standard
Custom time	—
Public URL	Not applicable
Authenticated URL	https://storage.cloud.google.com/03-e6fec888554c/PS_20174392719_1491204439457_log.csv?authuser=1
gsutil URI	gs://03-e6fec888554c/PS_20174392719_1491204439457_log.csv

Below the overview table is the 'Permissions' section, which includes a 'Public access' toggle set to 'Not public' and a 'Hold status' set to 'None'. A tooltip 'Explore Cloud Storage Release Notes' is visible over the 'Public access' toggle. At the bottom of the console, there is a 'CLOUD SHELL Terminal' window and an 'Open Editor' button.

Leveraging AI models with GCP

The screenshot displays the Google Cloud Platform (GCP) BigQuery console. The top navigation bar shows the GCP logo, a project ID (e6fec888554c), and a search bar. The left sidebar contains navigation icons and a list of pinned projects. The main area is divided into three sections: Explorer, Query Editor, and Query Results.

Explorer: Shows a tree view of the project structure. The 'finance' dataset is expanded, showing a table named 'model_unsupervised'.

Query Editor: Contains a SQL query for creating or replacing a machine learning model. The query is as follows:

```
1 CREATE OR REPLACE MODEL
2   finance.model_unsupervised OPTIONS(model_type='kmeans', num_clusters=5) AS
3   SELECT
4     amount, oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest, type, origzeroFlag, destzeroFlag, amountError
5   FROM
6     `finance.fraud_data_model`
```

Buttons for 'RUN', 'SAVE', 'SCHEDULE', and 'MORE' are visible. A status message indicates: "This query will process 16.1 MIB (ML) when run."

Query Results: Shows the execution details of the query. The query is complete, having taken 1 min 16 sec to execute and processed 16.1 MB of ML data.

Job information		Results	Execution details
Elapsed time	Slot time consumed	Stages	Training iterations
1 min 16 sec	14 min 10.785 sec	<ul style="list-style-type: none">Preprocess: 13.659 secTrain: 57.656 secEvaluate: 4.012 sec	Completed: 6 Planned: 20

At the bottom, there is a 'Loss' section with a 'Duration (seconds)' column. Below this, there are tabs for 'JOB HISTORY', 'QUERY HISTORY', and 'SAVED QUERIES'.

Leveraging AI models with GCP

The screenshot displays the Google Cloud Platform (GCP) BigQuery interface. The top navigation bar shows the GCP logo, the project name 'e6fec888554c', and a search bar with the text 'big'. The left sidebar contains the 'Explorer' panel with a search bar and a list of pinned projects. The main panel shows the 'model_unsupervised' model in the 'EVALUATION' tab. The 'Selected Features' dropdown lists 'amount, amountError, destzeroFlag, newbalanceDest, newbalanceOrig, oldbalan...'. Below this is a table with 10 columns: Centroid Id, Count, amount, amountError, destzeroFlag, newbalanceDest, and newbalanceOrig. The table contains 5 rows of data. Below the table is a section for 'Categorical features' with a description and a 'Selected Features' dropdown. The bottom of the interface shows the 'CLOUD SHELL' terminal and a 'Terminal' tab.

Google Cloud Platform e6fec888554c big

FEATURES & INFO SHORTCUT DISABLE EDITOR TABS

Explorer + ADD DATA

model_unsupervised

DETAILS TRAINING EVALUATION SCHEMA

numeric features.

Selected Features
amount, amountError, destzeroFlag, newbalanceDest, newbalanceOrig, oldbalan...

Centroid Id	Count	amount	amountError	destzeroFlag	newbalanceDest	newbalanceOrig
1	79,318	167,862.6702	0.0000	1.0000	1,598,937.0859	0.0000
2	116,113	267,914.4605	0.0000	0.0000	1,078,378.5061	2.0000
3	4,534	1,624,631.5856	0.9352	0.0626	578,283.5899	2.0000
4	7,414	1,465,995.0856	0.0000	0.6481	16,608,629.9893	9.0000
5	21,171	685,064.4304	0.0000	0.9749	2,774,179.2326	0.0000

Categorical features

Each chart below shows the category value distribution for a particular feature. Use the select menu to view more categorical features.

Selected Features

JOB HISTORY QUERY HISTORY SAVED QUERIES

CLOUD SHELL Terminal e6fec888554c

Open Editor

Leveraging AI models with GCP

The screenshot displays the Google Cloud Platform (GCP) BigQuery console. The top navigation bar shows the project ID 'e6fec888554c' and a search bar with 'big'. The left sidebar contains the 'Explorer' panel with a search bar and a tree view of the project structure. The main panel shows a SQL query being executed, with a 'RUN' button and a status message indicating the query will process 4.4 MiB. Below the query editor, the 'Query results' section shows the execution status and a table of results.

Explorer Panel:

- Project: e6fec88...
- Dataset: finance
- Model: model_unsupervised
- Tables: fraud_data, fraud_data_model, fraud_data_sample, fraud_data_test

Query Editor:

```
1 SELECT
2   centroid_id, sum(isfraud) as fraud_cnt, count(*) total_cnt
3 FROM
4   ML.PREDICT(MODEL `finance.model_unsupervised`,
5   (
6     SELECT *
7     FROM `finance.fraud_data_test`))
8 group by centroid_id
9 order by centroid_id
```

Query Results:

Query complete (1.7 sec elapsed, 4.4 MB processed)

Job information | **Results** | JSON | Execution details

Row	centroid_id	fraud_cnt	total_cnt
1	1	4	19717
2	2	742	28561
3	3	851	1084

Bottom Panel:

- CLOUD SHELL Terminal
- Open Editor
- Job History | Query History | Saved Queries

Leveraging AI models with GCP

Google Cloud Platform

1efca51e287b

notebooks

X

▼

📄

?

1

⋮

👤

Notebook details

OPEN JUPYTERLAB

EDIT

VIEW VM DETAILS

▶ START

■ STOP

🔄 RESET

⬆️ UPGRADE

🗑️ DELETE

🏠

📄

📁

🔑

📄

🔊

☰

🏠

💡

🔁

🔔

📄

🛒

ID

✅ tensorflow-1-15-20211019-121438

BASIC INFO

HEALTH

MONITORING

LOGS

Region	us-west1 (Oregon)
Zone	us-west1-b
Environment ?	TensorFlow Enterprise 1.15 (with LTS and Intel® MKL-DNN/MKL)
Environment version	M81
Machine type ?	n1-standard-4 (4 vCPUs, 15 GB RAM)
GPU ?	None
Boot disk	100 GB disk
Data disk	100 GB disk
Created	Oct 19, 2021, 12:14:42 PM
Last modified	Oct 19, 2021, 12:17:39 PM
Backup	Not specified
Subnetwork	default
Service account	43590791018-compute@developer.gserviceaccount.com
Permission mode	Service account
Sudo access	Enabled
File downloads	Enabled
nbconvert	Enabled
Instance health	System health report enabled Custom metrics reporting not enabled Cloud Monitoring agent not installed

\$97.10 monthly estimate

That's about \$0.133 hourly

Pay for what you use: No upfront costs and per second billing

Networking cost also applies. [Learn more](#)

▼ DETAILS

Leveraging AI models with GCP

```
File Edit View Run Kernel Git Tabs Settings Help

Untitled.ipynb Python 3

[ ]: 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()

[ ]: 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]

[ ]: 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)

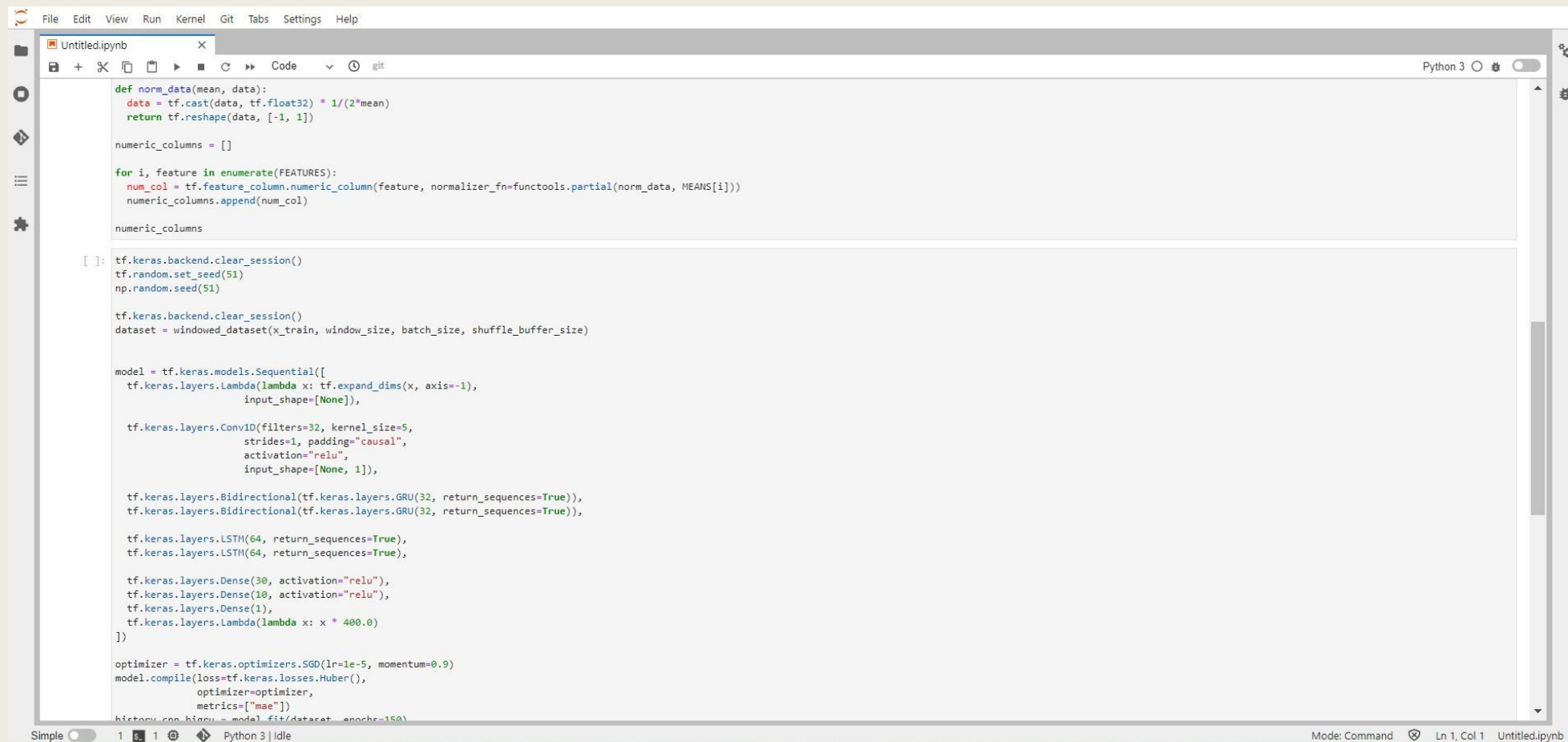
[ ]: BATCH_SIZE = 32

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

[ ]: MFANS = [94816.7387536405, -0.0011219465482001268, -0.0021445914636999603, -0.002317402958335562,
```

Leveraging AI models with GCP



The screenshot displays a Jupyter Notebook window titled 'Untitled.ipynb'. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. Below the menu is a toolbar with icons for file operations and execution. The main area contains two code cells. The first cell defines a normalization function and iterates over features to normalize them. The second cell sets up a Keras model with various layers, including a Lambda layer for expansion, a Conv1D layer with causal padding, two BiDirectional GRU layers, two LSTM layers, and three Dense layers, followed by a final Lambda layer. The model is compiled with SGD optimizer and Huber loss.

```
File Edit View Run Kernel Git Tabs Settings Help

Untitled.ipynb X
+ - 🔍 📄 ▶️ ⏏️ Code ⌵ ⓘ git Python 3 ⚙️ 🔌

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_callback = model_fit(dataset, epochs=150)
```

Simple 1 1 Python 3 | Idle Mode: Command Ln 1, Col 1 Untitled.ipynb

Thank You

Conclusion & Perspectives

Q & A

A decorative graphic in the bottom right corner consisting of two L-shaped lines. The inner line is olive green and the outer line is cream-colored, both pointing towards the bottom right corner.