

Credit Card Fraud Detection

A Project Report

Submitted by

Group 6

comprising the following members

Fayez Ahmad (0431346)

Khushi Dipakkumar Bhabhor (0444179)

Leyon Ibn Kamal (0441985)

Logesh Guhan Narayananamoothy (0439721)

Roya Bakhshi (0428349)

*in partial fulfilment for completing the **Intro to Cloud Technologies CPSC-5207EL-01-2023F** course instructed by **Jaspreet Bhatia***

GitHub: <https://github.com/leyoncode/GCP-CreditCardFraudDetection>

Laurentian University

December 2023

Table of Contents

Abstract	3
1. Introduction:.....	3
Methodology.....	3
2. Overview:	3
2.1 Pipeline:	4
2.2 Data Collection:	4
3. Cloud Storage:	4
4. BigQuery:	6
5. Vertex AI:.....	8
5.1 Vertex AI – Workbench and Custom model training:	9
5.2 Vertex AI – AutoML:.....	14
5.3 Vertex AI – Endpoint:	19
6. Future Work and Conclusion:	21
References	23

Abstract

In this report, we describe the methods to create an efficient and cost-effective way to build a machine learning model for classification of fraudulent credit card transactions. We tested different components of GCP to figure out the best methodology to use. We used Cloud Storage buckets, BigQuery, and Vertex AI to build a complete end-to-end ML pipeline. We tested building a custom model using Vertex AI Workbench as a platform to run python code for building a model using Pytorch, as well as using Vertex AI AutoML and learnt that AutoML while a little more expensive, it saves a lot of time and complexity while building very optimized machine learning models ready to be used in production.

1. Introduction:

Credit card fraud detection is a collective term for the policies, tools, methodologies, and practices that credit card companies and financial institutions take to combat identity fraud and stop fraudulent transactions. A credit card fraud happens when someone steals or uses your credit card or credit card information without your permission. A person might steal your credit card information by looking for bank statements or information in your trash or mailbox hacking into computers of companies and stealing credit card information, such as your ISP, installing small devices on payment terminals that record your credit card information, sending you a fraudulent email asking for your credit card information (also called phishing), asking you to use your credit card on an illegitimate website to make a purchase.

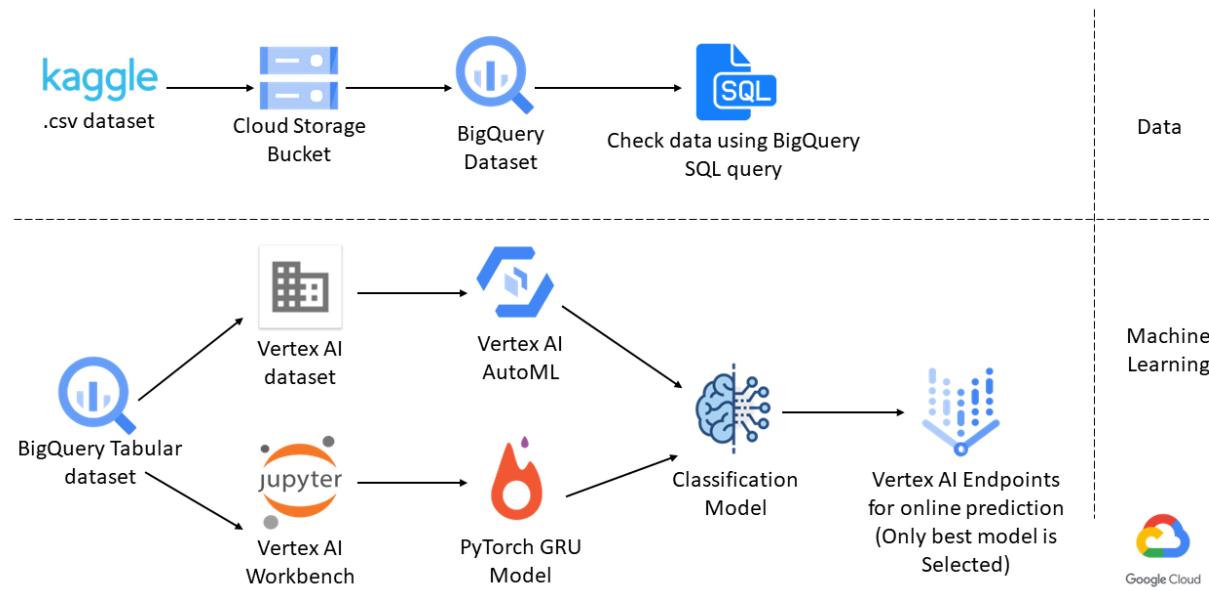
In recent years, as the amount of data has exploded and the number of payment card transactions has skyrocketed, credit card fraud detection has become largely digitized and automated. The credit card fraud tops the list of identity thefts as a result of how easy it can be done. So, it is important for the credit card companies to recognize fraudulent transactions so that customers are not charged for items they did not purchase.

Methodology

2. Overview:

We will be building our ML pipeline on GCP. Google Cloud Platform is a suite of cloud computing services provided by Google. It offers a wide range of infrastructure and platform services for computing, storage, data analytics, machine learning, networking, and more. GCP provides a scalable and flexible cloud environment that allows businesses and developers to build, deploy, and scale applications seamlessly. Some key services within GCP include Compute Engine (virtual machines), Cloud Storage (object storage), BigQuery (serverless data warehouse), and AI/ML services like Vertex AI. With a global network of data centers, GCP enables users to access resources and services with high performance and reliability. GCP is known for its innovation, security features, and a pay-as-you-go pricing model, making it a popular choice for organizations looking to leverage cloud computing for their digital initiatives.

2.1 Pipeline:



Above figure gives an overview of how we are using GCP to build an end-to-end machine learning pipeline from data collection to deployment. We take advantage of multiple GCP components. We use Cloud Storage buckets to store the CSV dataset that we collected from Kaggle. We then use BigQuery to load the dataset to be used for Machine Learning and analytical purposes. Then we use Vertex AI and its components to build and test two models, one using AutoML and another using Pytorch in a Vertex AI Workbench. We compare the two and deploy the best model to Vertex AI endpoint, which can be used by backend applications for classifying future credit card transactions for fraudulent activity.

2.2 Data Collection:

We collected our dataset from Kaggle (Check reference 2) The dataset contains credit card transactions made by European cardholders in the year 2023. It comprises over 500,000 records. It contains data relating to user information, time, location, amount, etc., that are connected to each transaction. The data in the dataset has been anonymized to protect the cardholder's privacy. The primary objective of this dataset is to facilitate the development of fraud detection algorithms and models to identify potentially fraudulent transactions.

3. Cloud Storage:

Google Cloud Storage is a managed service for storing unstructured data. It can be used to store any amount of data and can be retrieved as often as we like. GCS is an object storage system, meaning it stores data as objects rather than files in a hierarchical file system. Each object consists of data, metadata, and a unique identifier. GCS is designed to be highly scalable, allowing you to store and retrieve massive amounts of data. It automatically scales to handle growing datasets. Google Cloud Storage provides high durability for stored objects. It replicates data across multiple locations and storage devices to ensure data integrity and availability. GCS offers different storage classes to optimize costs based on the access frequency and availability requirements of your data. Classes include Standard, Nearline, Coldline, and Archive. GCS supports versioning, allowing you

to preserve, retrieve, and delete every version of every object in a bucket. This helps with data recovery and version management.

We used cloud storage to store the CSV dataset that we collected from Kaggle. This step is required as we need to store our data in BigQuery, but BigQuery cannot load a large CSV file directly from our local computer. So instead BigQuery loads the data from a Bucket where the CSV data is stored.

To create a Bucket in Cloud Storage is simple. We first need to decide what kind of storage class we need, in our case, we can use a nearline or coldline class when experimenting, but for final implementation, we can just use archive class as once the data is loaded into BigQuery, we will not need the data anymore.

Choose a storage class for your data

A storage class sets costs for storage, retrieval, and operations, with minimal differences in uptime. Choose if you want objects to be managed automatically or specify a default storage class based on how long you plan to store your data and your workload or use case. [Learn more](#)

- Autoclass ?
Automatically transitions each object to Standard or Nearline class based on object-level activity, to optimize for cost and latency. Recommended if usage frequency may be unpredictable. Can be changed to a default class at any time. [Pricing details](#)
- Set a default class
Applies to all objects in your bucket unless you manually modify the class per object or set object lifecycle rules. Best when your usage is highly predictable. Can't be changed to Autoclass once the bucket is created.
- Standard ?
Best for short-term storage and frequently accessed data
- Nearline
Best for backups and data accessed less than once a month
- Coldline
Best for disaster recovery and data accessed less than once a quarter
- Archive
Best for long-term digital preservation of data accessed less than once a year

Then we can select to prevent public access as we only need to use it for loading data into BigQuery this have no connection to the public users.

- Choose how to control access to objects

Prevent public access

Restrict data from being publicly accessible via the internet. Will prevent this bucket from being used for web hosting. [Learn more](#)

- Enforce public access prevention on this bucket

Access control

- Uniform

Ensure uniform access to all objects in the bucket by using only bucket-level permissions (IAM). This option becomes permanent after 90 days. [Learn more](#)

- Fine-grained

Specify access to individual objects by using object-level permissions (ACLs) in addition to your bucket-level permissions (IAM). [Learn more](#)

[CONTINUE](#)

We can skip versioning as well since we only need to use the data once and create our bucket.

The screenshot shows the 'Bucket details' page for a bucket named 'creditcardfraudetection-datasetbucket'. The bucket is located in 'us-central1 (Iowa)' with 'Coldline' storage class, 'Not public' public access, and 'None' protection. The 'OBJECTS' tab is selected, showing a single object named 'creditcardfraudetectiondataset...' with a size of 309.8 MB, type 'text/csv', created on Dec 7, 2023, at 12:00:44 PM, and last modified on Dec 7, 2023, at 12:00:44 PM. The object has 'Coldline' storage class, 'Not public' public access, and no version history. The page also includes navigation links like 'Upload files', 'Upload folder', 'Create folder', 'Transfer data', and 'Delete'.

4. BigQuery:

BigQuery is a serverless and cost-effective enterprise data warehouse that works across clouds and scales with your data. It uses built-in ML/AI and BI for insights at scale. BigQuery uses a familiar SQL-like language for querying and analysing data, making it accessible to users with SQL skills and is designed to handle large-scale datasets with ease. It can process and analyse petabytes of data quickly and efficiently and can also separate storage and compute, allowing you to store your data in a cost-effective manner and scale compute resources independently as needed. BigQuery supports real-time analytics, allowing you to analyse streaming data as it arrives. This is useful for applications that require up-to-the-minute insights. BigQuery offers both on-demand (pay-as-you-go) pricing and flat-rate pricing for more predictable costs. The flat-rate pricing is suitable for organizations with predictable workloads.

Since BigQuery is good for data warehousing, it makes sense to store customer transactions data in BigQuery as there will be millions of transactions in a real-world scenario. But beyond that, BigQuery also allows us to process the entire dataset quickly as shown in a screenshot below where we checked all the columns for null data which is a common pre-processing step where it took 34 seconds to process 500,000 data.

To use BigQuery, we first need to create a named dataset and create a table for it. To load data into this table, we select our CSV file in our bucket.

Row	V22	V23	V24	V25	V26	V27	V28	Amount	Class
1	0.390052723	0.9991826	-0.264927519	-0.01147596	1.568279754	0.255640676	0.26731602	5461.27	1
2	0.797662285	0.299482413	0.972730633	-1.589691024	-0.672111677	-2.913412	-2.063756562	20957.61	1
3	-0.747406813	-0.106804225	-0.191209074	0.556891781	0.253903093	0.602826548	0.700296564	1753.24	1
4	0.18782932	0.233161382	-0.330050272	-0.58490664	-0.988624849	-1.336684254	1.881743689	15945.66	1
5	-0.117533794	-0.216111057	-1.506194092	1.060795648	-0.447893487	-0.184512045	-0.0031107	2766.95	1
6	0.135871035	0.628709377	-1.941852329	-0.681935811	1.350706514	2.061035717	1.631332081	15537.06	1
7	-0.551967315	0.014833986	0.857446553	-0.003744802	0.679945328	0.406796078	0.470710342	17732.45	1
8	0.1668604	1.190288851	-0.14515751	0.405437491	0.032730234	0.212365192	-1.087499059	19694.02	1
9	-0.514868376	-2.57794449	-0.999141516	-2.509350599	-1.818791061	0.101113542	0.50255426	23218.44	1
10	0.289644456	-0.097091	0.501042647	-0.283258854	-0.716030829	-0.138222782	-0.512175571	7659.1	1
11	-0.070807036	-0.238575377	0.315096654	-0.049906303	-0.725953995	-0.584575906	-0.97277146	13763.48	1

```

1  SELECT
2  COUNTIF(V1 IS NULL) AS V1_is_null,
3  COUNTIF(V2 IS NULL) AS V2_is_null,
4  COUNTIF(V3 IS NULL) AS V3_is_null,
5  COUNTIF(V4 IS NULL) AS V4_is_null,
6  COUNTIF(V5 IS NULL) AS V5_is_null,
7  COUNTIF(V6 IS NULL) AS V6_is_null,
8  COUNTIF(V7 IS NULL) AS V7_is_null,
9  COUNTIF(V8 IS NULL) AS V8_is_null,
10 COUNTIF(V9 IS NULL) AS V9_is_null,
11 COUNTIF(V10 IS NULL) AS V10_is_null,
12 COUNTIF(V11 IS NULL) AS V11_is_null,
13 COUNTIF(V12 IS NULL) AS V12_is_null,
14 COUNTIF(V13 IS NULL) AS V13_is_null,
15 COUNTIF(V14 IS NULL) AS V14_is_null,
16 COUNTIF(V15 IS NULL) AS V15_is_null,
17 COUNTIF(V16 IS NULL) AS V16_is_null,
18 COUNTIF(V17 IS NULL) AS V17_is_null,
19 COUNTIF(V18 IS NULL) AS V18_is_null,
20 COUNTIF(V19 IS NULL) AS V19_is_null,
21 COUNTIF(V20 IS NULL) AS V20_is_null,
22 COUNTIF(V21 IS NULL) AS V21_is_null,
23 COUNTIF(V22 IS NULL) AS V22_is_null,
24 COUNTIF(V23 IS NULL) AS V23_is_null,
25 COUNTIF(V24 IS NULL) AS V24_is_null,
26 COUNTIF(V25 IS NULL) AS V25_is_null,
27 COUNTIF(V26 IS NULL) AS V26_is_null,
28 COUNTIF(V27 IS NULL) AS V27_is_null,
29 COUNTIF(V28 IS NULL) AS V28_is_null,
30 COUNTIF(Amount IS NULL) AS Amount_is_null,
31 COUNTIF(Class IS NULL) AS Class_is_null
32 FROM `teak-listener-398917.creditcardtransactions.creditcarddata`
```

Press Alt+F1 for Accessibility Optic

Query results

[SAVE RESULTS](#) [EXPLORE DATA](#)

JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH	
Row	V24_is_null 0	V25_is_null 0	V26_is_null 0	V27_is_null 0	V28_is_null 0	Amount_is_null 0	Class_is_null 0

5. Vertex AI:

Vertex AI is a machine learning platform that lets GCP users train and deploy ML models and AI based applications. Vertex AI combines data engineering, data science, and ML engineering workflows, enabling your teams to collaborate using a common toolset and scale the applications using the benefits of Google Cloud. Vertex AI provides multiple components for various ML use cases. We used Vertex AI Workbench, Datasets, AutoML, Model Registry, and Endpoint in our project. In general, all machine learning projects involve testing multiple methods to determine the best model for a particular use case. We tested Vertex AI's AutoML and a GRU model.

Vertex AI Workbench provides a managed JupyterLab notebook for various platforms, like Tensorflow, Pytorch, XgBoost, etc. We used Pytorch to build and test a custom model that uses a simple GRU based model.

5.1 Vertex AI – Workbench and Custom model training:

To build a Workbench environment, we took the following steps:

1. First we create a workbench instance with a name and region.

The screenshot shows the 'Create instance' wizard for Vertex AI. The left sidebar has 'Workbench' selected under 'TOOLS'. The main pane shows the 'Details' step, where a name 'custommodel-managed-notebook-20231208-100343' is entered, starting with a letter and containing lowercase letters, numbers, or hyphens. The region is set to 'us-central1 (Iowa)'. A 'Pricing summary' section indicates a monthly estimate of \$50.75 and hourly cost of \$0.070. A 'CONTINUE' button is visible at the bottom of the step.

2. Then we select the machine type. We selected the lowest configuration machine for cost-saving purposes.

The screenshot shows the 'Create instance' wizard on the 'Machine type' step. The left sidebar has 'Workbench' selected. The main pane shows the 'Machine type' section, with 'General purpose' selected. It lists the 'N1' series as balanced price & performance, with 1 vCPU and 3.75 GB memory. A 'Pricing summary' section shows a monthly estimate of \$50.75 and hourly cost of \$0.070. A 'CONTINUE' button is visible at the bottom of the step.

3. Now we can create our instance.

The screenshot shows the Google Cloud Vertex AI Workbench Instances page. The left sidebar includes sections for Vertex AI, Tools (Dashboard, Model Garden, Pipelines), Notebooks (Colab Enterprise, Workbench), Generative AI Studio (Overview, Language, Vision, Speech), and Data (Marketplace). The main area displays a table of instances. A preview message at the top states: "The new Workbench Instances bring the features of Managed and User-Managed Notebooks into a single offering." A "MIGRATE" button is present. The table has columns for Region, Notebook name, Owner, Version, Container images, Created, and Labels. One instance is listed: "custommodel-managed-notebook-20231208-104348" owned by "1049237380802-compute@developer.gserviceaccount.com".

4. We can then open the JupyterLab and create a notebook for our project. We will use Pytorch, so we create a Pytorch notebook.

The screenshot shows the JupyterLab interface. The left sidebar contains a file tree with "src" and "tutorials" folders. The main area is the "Launcher" tab, which is currently selected. It displays two sections: "Notebook" and "Console". Under "Notebook", there are icons for Python (Local), Serverless Spark, PySpark (Local), PyTorch 1.12 (Local), PyTorch 1.13 (Local), R (Local), TensorFlow 2.10 (Local), and TensorFlow 2.11 (Local). Under "Console", there are icons for Python (Local), PySpark (Local), PyTorch 1.12 (Local), PyTorch 1.13 (Local), R (Local), TensorFlow 2.10 (Local), TensorFlow 2.11 (Local), and XGBoost 1.1 (Local). The status bar at the bottom shows "Simple" mode, 0 notebooks, and 0 consoles.

A JupyterLab notebook contains multiple cells to separate code and make it easy to use.

The screenshot shows a JupyterLab environment. On the left, there is a file browser window titled "custommodel-managed-notebook-20231208-104348". It lists several files and folders: "src" (3 days ago), "tutorials" (3 days ago), "custom-pytorch-model.ipynb" (2 days ago, currently selected), and "my-custom-pytorch-model.pkl" (2 days ago). The main workspace is a code editor titled "custom-pytorch-model.ipynb". It contains two code cells. Cell [1] imports various Python libraries including Google Cloud API, PyTorch, and scikit-learn. Cell [2] sets up Google Cloud project details, specifies a Vertex AI Dataset ID, and defines a model bucket path. The status bar at the bottom indicates "PyTorch 1.13 (Local) | Idle".

First, we load the data directly from BigQuery into a pandas dataframe by using an SQL query.

```
# Set up BigQuery client
bq_client = bigquery.Client(project)

query_sql = "SELECT * FROM `teak-listener-398917.creditcardtransactions.creditcarddata`"

df = bq_client.query(query_sql).to_dataframe()

df.head()
```

V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0.097934	-0.293185	-0.475286	...	0.209819	0.390053	0.999183	-0.264928	-0.011476	1.568280	0.255641	0.267316	5461.27	1
-0.541339	-2.490349	-2.421127	...	-1.550943	0.797662	0.299482	0.972731	-1.589691	-0.672112	-2.913412	-2.063757	20957.61	1
0.068382	-0.357084	-0.669457	...	0.026757	-0.747407	-0.106804	-1.019209	0.556892	0.253903	0.602827	0.700297	1753.24	1
0.050744	0.553921	-0.753391	...	0.118581	0.187829	0.233161	-0.330050	-0.584907	-0.988625	-1.336684	1.881744	15945.66	1
-0.144174	0.574115	0.387199	...	-0.123330	-0.117534	-0.216111	-1.506194	1.060796	-0.447893	-0.184512	-0.003111	2766.95	1

We can then manually pre-process the dataset, remove null values, and analyse any data such as the transaction amount in the dataset.

```
num_null = df.isnull().sum().sum()

print("Number of null values:", num_null)

if num_null > 0:
    df.dropna(axis=0)
```

```
Number of null values: 0
```

```
df["Amount"].describe()
```

```
count      568630.000000
mean       12041.957635
std        6919.644449
min        50.010000
25%       6054.892500
50%       12030.150000
75%       18036.330000
max       24039.930000
Name: Amount, dtype: float64
```

```
# Separate features and target variable
X = df.drop("Class", axis=1)
y = df["Class"]
```

```
y.value_counts()
```

```
1    284315
0    284315
Name: Class, dtype: Int64
```

To create a custom model, we used Pytorch, which is a Python framework used for deep learning. We then built a simple GRU model. GRU is a type of deep learning model that is based on RNN-LSTM models. It improves on previous architectures by using gated mechanisms and with improved efficiency and performance.

```

class GRUModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(GRUModel, self).__init__()
        self.gru = nn.GRU(input_size, hidden_size, num_layers=3, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        _, h_n = self.gru(x)
        x = h_n[-1, :, :]
        x = self.fc(x)
        x = self.sigmoid(x)
        return x

# Instantiate the model with desired sizes and move to GPU
input_size = X_train.shape[1]
hidden_size = 256 # You can adjust this
output_size = 1 # For binary classification
model = GRUModel(input_size, hidden_size, output_size)

# Define loss function and optimizer
criterion = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
criterion.to(device)

```

Then we select a train-test split and get the data ready for training and testing.

```

# Perform stratified train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# Convert data to PyTorch tensors with 'Long' data type for target labels
X_train_tensor = torch.tensor(X_train.values, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.long) # Use torch.long for classification labels
X_test_tensor = torch.tensor(X_test.values, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.long)

# Create DataLoader for training and testing
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

```

From the training and testing reports, we can see that this custom training model is overfitting. Such a model will not be very useful in a real-world scenario.

Epoch 1/10 Loss: 0.6034 Accuracy: 100.00%	Test Accuracy: 94.11%
Epoch 2/10 Loss: 0.3966 Accuracy: 100.00%	Confusion Matrix:
Epoch 3/10 Loss: 0.3168 Accuracy: 100.00%	[[56618 245]
Epoch 4/10 Loss: 0.2945 Accuracy: 100.00%	[6451 50412]]
Epoch 5/10 Loss: 0.2809 Accuracy: 100.00%	Classification Report:
Epoch 6/10 Loss: 0.2958 Accuracy: 100.00%	precision recall f1-score support
Epoch 7/10 Loss: 0.2664 Accuracy: 100.00%	Class 0 0.90 1.00 0.94 56863
Epoch 8/10 Loss: 0.2666 Accuracy: 100.00%	Class 1 1.00 0.89 0.94 56863
Epoch 9/10 Loss: 0.2536 Accuracy: 100.00%	accuracy 0.95 0.94 0.94 113726
Epoch 10/10 Loss: 0.2363 Accuracy: 100.00%	macro avg 0.95 0.94 0.94 113726
Total training time (s): 1508.1906161308289	weighted avg 0.95 0.94 0.94 113726

5.2 Vertex AI – AutoML:

AutoML is a part of Vertex AI that makes it easy to create a classification model. Vertex AI AutoML simplifies the process of building machine learning models by automating various tasks that would typically require expertise in machine learning and data science. To use AutoML, we just need to specify the dataset, and select what kind of data we are working with and if we want a classification or regression model.

So first, we need to create a Vertex AI Dataset which contains metadata relating to how the data should be prepared for training and testing.

The screenshot shows the Google Cloud Vertex AI interface. On the left, there's a sidebar with categories like Vision, Speech, Feature Store, Datasets (which is selected), Labeling tasks, Training, Experiments, Metadata, Model Registry, Online prediction, and Marketplace. The main area has a header 'Create dataset' with a back arrow. It asks for a 'Dataset name' (set to 'creditcardfraud-vertexaidataset') and says 'Can use up to 128 characters'. Below that, it says 'Select a data type and objective'. It offers three options: IMAGE, TABULAR (selected), TEXT, and VIDEO. Under TABULAR, there are two sub-options: 'Regression/classification' (selected) and 'Forecasting'. 'Regression/classification' is described as 'Predict a target column's value. Supports tables with hundreds of columns and millions of rows.' 'Forecasting' is described as 'Predict the likelihood of certain events or demand.' At the bottom, there's a 'Region' dropdown set to 'us-central1 (Iowa)'.

Google Cloud My First Project Search (/) for resources, docs, products, and more

Vertex AI creditcardfraud-vertexaiddataset SOURCE ANALYZE

Add data to your dataset

Before you begin, review the data guide to make sure your data is formatted correctly and optimized for the best results.

[VIEW DATA GUIDE](#)

Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)
- Upload CSV files from your computer
- Select CSV files from Cloud Storage
- Select a table or view from BigQuery

Summary

You can build two model types with tabular data. The model type is automatically chosen based on the data type of your target column.

- Regression models predict a numeric value. For example, predicting home prices or consumer spending.
- Classification models predict a category from a fixed number of categories. Examples include predicting whether an email is spam or not, or classes a student might be interested in attending.

Select a table or view from BigQuery

Use a BigQuery table or view as your data source. You'll need [permission to access the dataset](#) and get the [dataset ID and table ID](#). [Learn more](#)

BigQuery path * teak-listener-398917.creditcardtransactions.creditcarddata BROWSE

Search by table name or path using the format: projectid.datasetid.tableid.

Vertex AI creditcardfraud-vertexaiddataset TRAIN NEW MODEL TRAIN NEW MODEL

SOURCE ANALYZE

Analyze

Summary	FLOAT	29 (96.67%)
Total columns: 30	INTEGER	1 (3.33%)
Total rows:-		

GENERATE STATISTICS

Filter Enter property name or value

Column name	BigQuery type	BigQuery mode	Missing % (count)	Distinct values
Amount	FLOAT	NULLABLE	-	-
Class	INTEGER	NULLABLE	-	-
V1	FLOAT	NULLABLE	-	-
V10	FLOAT	NULLABLE	-	-
V11	FLOAT	NULLABLE	-	-
V12	FLOAT	NULLABLE	-	-
V13	FLOAT	NULLABLE	-	-
V14	FLOAT	NULLABLE	-	-
V15	FLOAT	NULLABLE	-	-
V16	FLOAT	NULLABLE	-	-

Related resources

Training jobs and models

Use this dataset and annotation set to train a new machine learning model with AutoML or custom code. Selecting [AutoML on Pipelines](#) will create a Run on Vertex AI Pipelines. Run information will be found on the [Runs](#) tab under [Pipelines](#).

[TRAIN NEW MODEL](#)

The screenshot shows the Google Cloud Vertex AI Datasets page. The left sidebar includes sections for Vision, Speech, DATA (Feature Store, Datasets, Labeling tasks), MODEL DEVELOPMENT (Training, Experiments, Metadata), and DEPLOY AND USE (Model Registry, Online prediction, Marketplace). The main area displays a table of datasets with a single row selected: 'creditcardfraud-vertexaidataset'. The table columns include Name, ID, Status, Region, Type, Items, Last updated, and Labels.

Name	ID	Status	Region	Type	Items	Last updated	Labels
creditcardfraud-vertexaidataset	1112029567658229760	Ready	us-central1 (Iowa)	Tabular	-	December 7, 2023	

Then with the Vertex AI Dataset ready, we can start using AutoML. First, we need to select the dataset and tell it that we want a classification model.

The screenshot shows the 'Train new model' dialog. The left sidebar is identical to the previous one. The dialog has five steps: 1. Training method (selected), 2. Model details, 3. Join Featurestore (optional), 4. Training options, and 5. Compute and pricing. Step 1 is titled 'Training method' and contains fields for 'Dataset' (set to 'creditcardfraud-vertexaidataset') and 'Objective' (set to 'Classification'). A note below says: 'Please refer to the pricing guide for more details (and available deployment options) for each method.' Below the note is a button to 'GO TO PIPELINES' and a 'LEARN MORE' link. The 'Model training method' section shows two options: 'AutoML' (selected) and 'Custom training (advanced)'. The 'AutoML' option is described as 'Train high-quality models with minimal effort and machine learning expertise. Just specify how long you want to train.' The 'Custom training (advanced)' option is described as 'Run your TensorFlow, scikit-learn, and XGBoost training applications in the cloud. Train with one of Google Cloud's pre-built containers or use your own.' A 'CONTINUE' button is at the bottom.

Then we can give it a name and create a new model.

Train new model

Training method

2 Model details

Train new model
Creates a new model group and assigns the trained model as version 1

Train new version
Trains model as a version of an existing model

Name * creditcardfraud-vertexaidataset

Description

Target column *

Export test dataset to BigQuery

ADVANCED OPTIONS

CONTINUE

START TRAINING CANCEL

After that we can decide how we want to split our dataset, which we have as an 80-10-10 split for training, evaluation and testing.

Train new model

Training method

2 Model details

Random
80% of your data is randomly assigned for training, 10% for validation, and 10% for testing

Training: 80%
Validation: 10%
Test: 10%



Manual
You assign each data row for training, validation, and testing. [Learn more](#)

Chronological
The earliest 80% of your data is assigned to training, the next 10% for validation and the latest 10% for testing. This option requires a Time column in your dataset. [Learn more](#)

Training 80% Validation 10% Testing 10%



Start time End time

START TRAINING CANCEL

Then we decide how we want our model to be optimized for, which we selected as AUC-ROC.

Optimization objective *

AUC ROC

Distinguish between classes

Log loss

Keeps prediction probabilities as accurate as possible

AUC PRC

Maximize precision-recall for the less common class

Precision at recall

Maximize precision for the less common class

Recall at precision

Maximize recall for the less common class

Here we can see that it took 2 hours to train.

The screenshot shows the Vertex AI interface for training pipelines. On the left, there's a sidebar with sections for GENERATIVE AI STUDIO, DATA, and MODEL DEVELOPMENT. Under MODEL DEVELOPMENT, the 'Training' section is selected, showing options like Experiments, Metadata, and Marketplace. The main area is titled 'Training' and includes buttons for 'TRAIN NEW MODEL' and 'REFRESH'. Below this are tabs for TRAINING PIPELINES, CUSTOM JOBS, HYPERPARAMETER TUNING JOBS, and NAS JOBS. A detailed description of training pipelines is provided. A dropdown menu for 'Region' is set to 'us-central1 (Iowa)'. A search bar labeled 'Filter' with the placeholder 'Enter a property name' is present. A table lists the training job details:

Name	ID	Status	Job type	Model type	Duration	Last updated	Created	Ended	Labels
creditcardfraud-vertexaidataset	9092399517424156672	Finished	Training pipeline	Tabular classification	2 hr 2 min	Dec 8, 2023, 4:12:12 AM	Dec 8, 2023, 2:08:59 AM	—	...

After training we can see the status of our model which is saved in vertex AI Model Registry.

The screenshot shows the Vertex AI interface with the 'Model Registry' tab selected. On the left, a sidebar lists categories: DATA (Feature Store, Datasets, Labeling tasks), MODEL DEVELOPMENT (Training, Experiments, Metadata), and DEPLOY AND USE (Model Registry, Online prediction, Batch predictions, Vector Search, Marketplace). The 'Model Registry' section is expanded, showing a table of models. One row is selected: 'creditcardfraud-vertexdataset' (Version 1, Deployed, Tabular, AutoML training, Updated Dec 8, 2023, 4:12:13 AM). The table includes columns for Name, Default version, Deployment status, Description, Type, Source, Updated, and Labels. A 'Region' dropdown is set to 'us-central1 (Iowa)'. A 'Filter' input field is present. Top navigation includes 'CREATE' and 'IMPORT' buttons, and right-hand controls for 'REFRESH' and 'LEARN'.

The AutoML model gives us a 100% accuracy with high precision, recall and F1-score.

PR AUC	1
ROC AUC	1
Log loss	0.002
F1 score	0.99959564
Micro-F1	0.99959564
Macro-F1	0.99959564
Precision	100%
Recall	100%



For most common training tasks, such as image and text classification, and classification on tabular data, Google recommends using AutoML and custom training for specialized use cases. The results of the AutoML compared with our custom model are proof of why we should use AutoML for training models for tabular data like ours. It reduces the complexity of what kind of model we need to create as there are just too many types of models and optimization parameters, and AutoML automates and simplifies everything for us.

5.3 Vertex AI – Endpoint:

Vertex AI Endpoint is yet another useful part of Vertex AI where we can easily deploy machine learning models in a few clicks. All we need to do is to select the model that have been deployed to the Vertex AI Model Registry and press deploy.

Edit endpoint

1 Define your endpoint

2 Model settings

3 Model monitoring

Endpoint name * vertex-ai-deployed-model

Region us-central1 (Iowa)

Access

Determines how your endpoint can be accessed. By default, endpoints are available for prediction serving through a REST API. AutoML and custom-trained models can be added to standard endpoints.

Standard Makes the endpoint available for prediction serving through a REST API. AutoML and custom-trained models can be added to standard endpoints.

Private Create a private connection to this endpoint using a VPC network and [private services access](#). Only custom-trained and tabular models can be added to private endpoints. [Learn more](#)

ADVANCED OPTIONS

CONTINUE

Edit endpoint

1 Define your endpoint

2 Model settings

3 Model monitoring

Region us-central1

Model creditcardfraud-vertexaidataset (Version 1) Status Ready Deployment ready

UPDATE CANCEL

creditcardfraud-vertexaidataset (Version 1)

Traffic split: 100%

Model settings

Deploying a model to an endpoint lets it serve online predictions. You can also deploy multiple models to one endpoint and split traffic. This lets you test out a new model before serving all traffic. [Learn more about model deployment](#)

ADD A MODEL

CONTINUE

Online prediction

ENDPOINTS

Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.

To create an endpoint, you need at least one machine learning model. [Learn more](#)

Region us-central1 (Iowa)

Endpoints + CREATE REFRESH

Name	ID	Status	Models	Deployment resource pool	Region	Monitoring	Most recent alerts	Last updated	API	Labels
vertex-ai-deployed-model	1115496327820607488	Active	1	—	us-central1	Disabled	—	Dec 9, 2023, 2:31:29 PM	SAMPLE REQUEST	

GCP also gives us hints on how we can call for predictions on this deployed model from a REST endpoint or a Python script.

The screenshot shows two views of the Google Cloud Vertex AI interface. The left view is the 'Online prediction' section under 'Endpoints'. It lists a single endpoint named 'vertex-ai-deployed-model' with ID 1115496327820607488, which is active and associated with one model and a deployment resource pool. The right view is a 'Sample Request' panel. The 'REST' tab is selected, showing steps to execute queries using the command line interface (CLI). Step 1: Make sure you have the Google Cloud SDK installed. Step 2: Run the following command to authenticate with your Google account. Step 3: Create a JSON object to hold your tabular data. Step 4: Create environment variables to hold your endpoint and project IDs, as well as your JSON object. Step 5: Execute the request. Below these steps are the corresponding CLI commands:

```

1. Make sure you have the Google Cloud SDK installed.
2. Run the following command to authenticate with your Google account.
   $ gcloud auth application-default login

3. Create a JSON object to hold your tabular data.
{
  "instances": [
    {
      "feature_column_a": "value",
      "feature_column_b": "value",
      ...
    },
    {
      "feature_column_a": "value",
      "feature_column_b": "value",
      ...
    },
    ...
  ]
}

4. Create environment variables to hold your endpoint and project IDs, as well as your JSON object.
$ ENDPOINT_ID="1115496327820607488"
$ PROJECT_ID="1049237380802"
$ INPUT_DATA_FILE="INPUT-JSON"

5. Execute the request.
$ curl \
  -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  https://us-central1-aiplatform.googleapis.com/v1/projects/$PROJECT_ID/models/$ENDPOINT_ID:predict \
  -d @"$INPUT_DATA_FILE"

```

The right view also includes a 'Python' tab and a 'DONE' button at the bottom.

6. Future Work and Conclusion:

For our project, we implemented the entire machine learning pipeline in GCP. Our custom model was not performing as well as the AutoML model, but the AutoML proved why it is the recommended way to build classification models for tabular data in GCP. For future work, we can implement a banking application that utilizes the endpoint to classify each transaction passing through it for fraudulent transactions and prevent fraudulent activity in real time.

In this project, we built a machine learning pipeline using multiple GCP components. We used Cloud Storage buckets, BigQuery, and Vertex AI to build a complete end-to-end ML pipeline. We discovered how simple AutoML can be to use compared to building machine learning the

traditional way. To build and train the entire model using all 500,000 data, it only cost \$34, which is very cheap, and all of the cost is from using AutoML. Using AutoML reduces months long experimentation into a few clicks.

Overall, we think the pipeline we built is the most cost effective and efficient in terms of both resources as well as time taken to build our solution.

References

1. Official Google Cloud Certified – Associate Cloud Engineer – Study guide by Dan Sullivan
2. Kaggle Dataset - <https://www.kaggle.com/datasets/nlgiriyewithana/credit-card-fraud-detection-dataset-2023/data>
3. Inscribe AI - <https://www.inscribe.ai/fraud-detection/credit-fraud-detection>
4. Financial Consumer Agency of Canada - <https://www.canada.ca/en/financial-consumer-agency/services/credit-fraud.html>
5. GCP Vertex AI Documentation - <https://cloud.google.com/vertex-ai/docs/training-overview>
6. Pytorch Documentation - <https://pytorch.org/docs/stable/generated/torch.nn.GRU.html>
7. ML model evaluation - https://scikit-learn.org/stable/modules/model_evaluation.html