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 Narayanamoorthy (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

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.

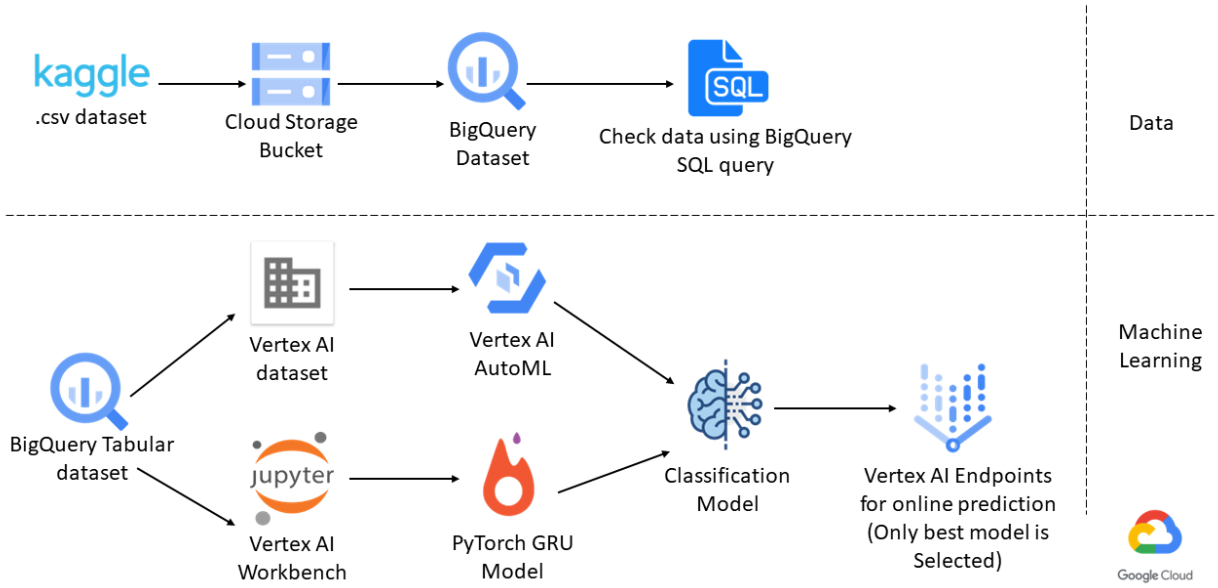
In recent years, as the amount of data has exploded and the number of payment card transactions has skyrocketed, credit card fraud detection has become largely digitized and automated. The credit card fraud tops the list of identity thefts as a result of how easy it can be done. So, it is important for 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.

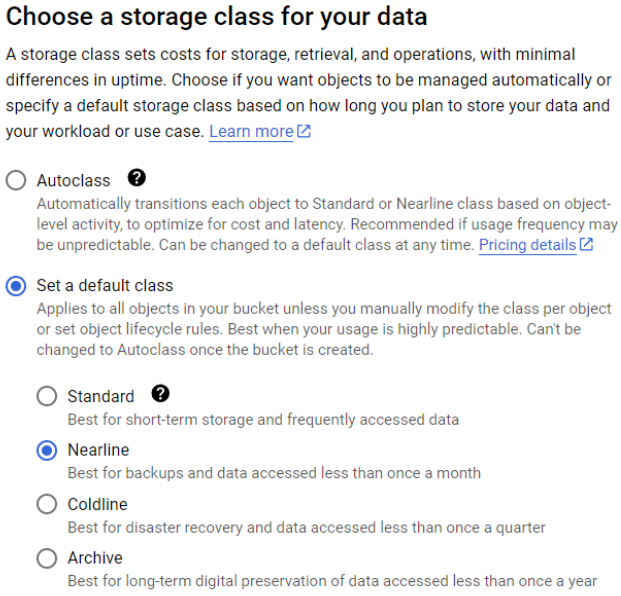
3. Cloud Storage:

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

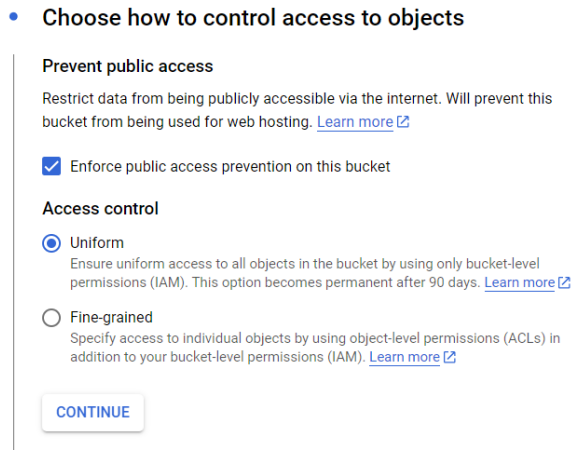
to preserve, retrieve, and delete every version of every object in a bucket. This helps with data recovery and version management.

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.

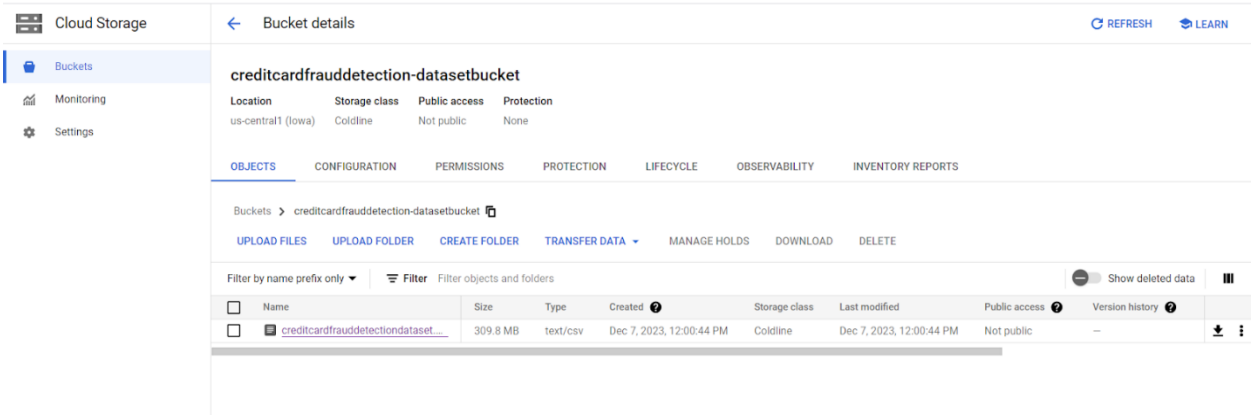
To create a Bucket in Cloud Storage is simple. We first need to decide what kind of storage class we need, in our case, we can use a nearline or coldline class when experimenting, but for final implementation, we can just use archive class as once the data is loaded into BigQuery, we will not need the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



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.



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

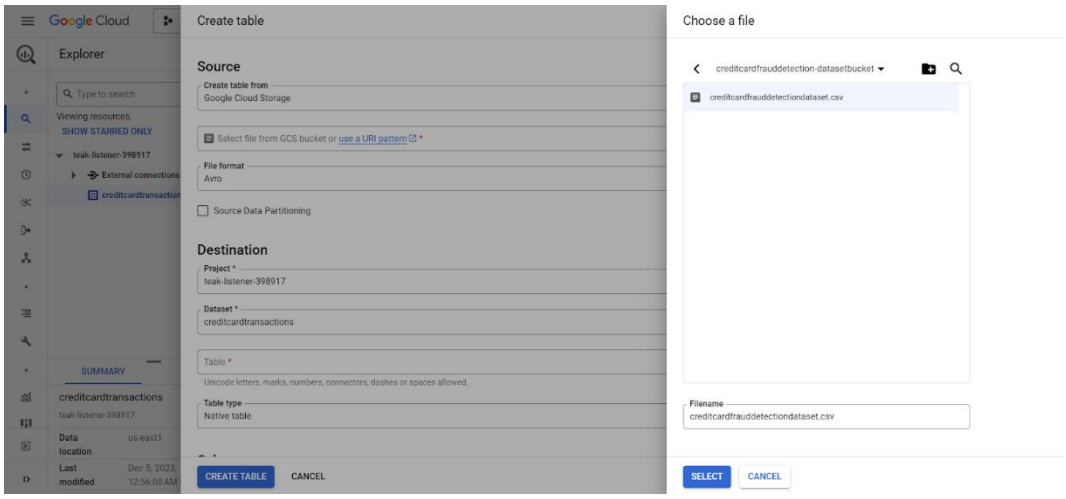


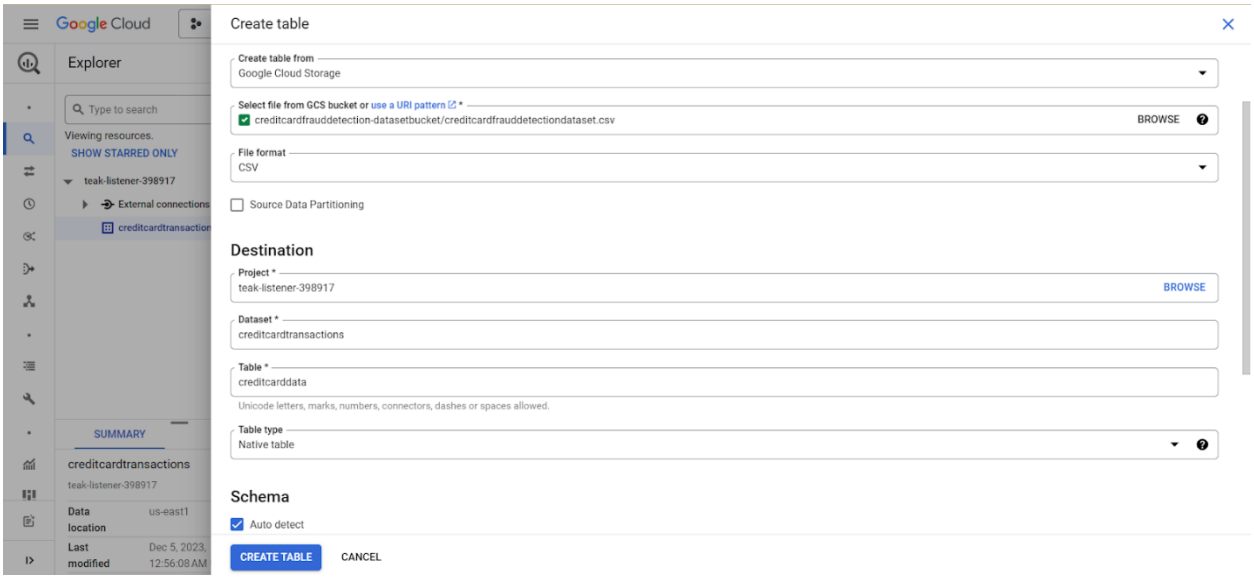
4. BigQuery:

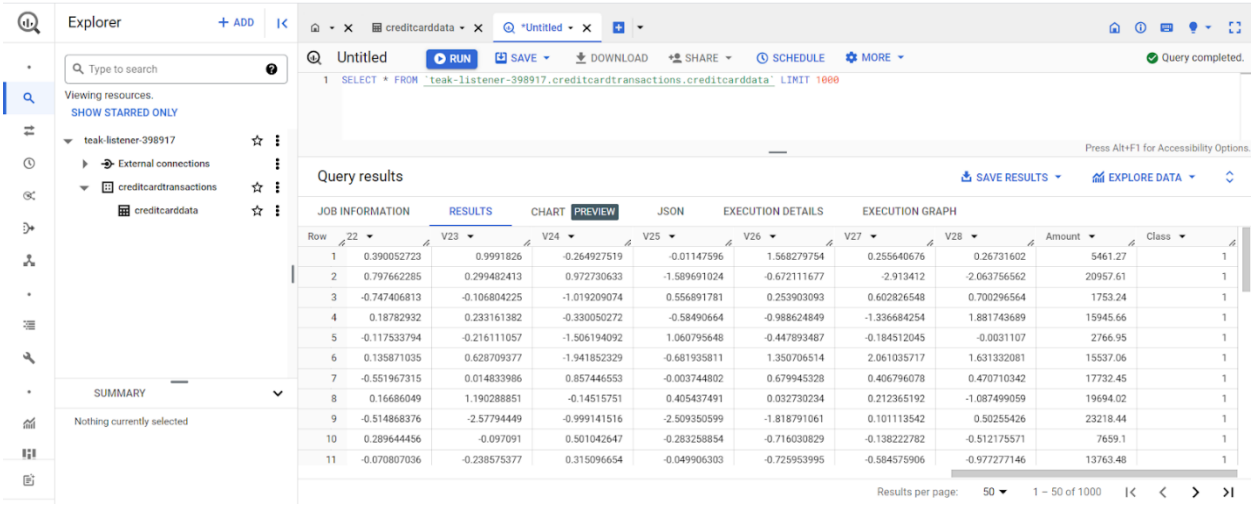
BigQuery is a serverless and cost-effective enterprise data warehouse that works across clouds and scales with your data. It uses built-in ML/AI and BI for insights at scale. BigQuery uses a familiar SQL-like language for querying and analysing data, making it accessible to users with SQL skills and 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.







```sql
SELECT
  COUNTIF(V1 IS NULL) AS V1_is_null,
  COUNTIF(V2 IS NULL) AS V2_is_null,
  COUNTIF(V3 IS NULL) AS V3_is_null,
  COUNTIF(V4 IS NULL) AS V4_is_null,
  COUNTIF(V5 IS NULL) AS V5_is_null,
  COUNTIF(V6 IS NULL) AS V6_is_null,
  COUNTIF(V7 IS NULL) AS V7_is_null,
  COUNTIF(V8 IS NULL) AS V8_is_null,
  COUNTIF(V9 IS NULL) AS V9_is_null,
  COUNTIF(V10 IS NULL) AS V10_is_null,
  COUNTIF(V11 IS NULL) AS V11_is_null,
  COUNTIF(V12 IS NULL) AS V12_is_null,
  COUNTIF(V13 IS NULL) AS V13_is_null,
  COUNTIF(V14 IS NULL) AS V14_is_null,
  COUNTIF(V15 IS NULL) AS V15_is_null,
  COUNTIF(V16 IS NULL) AS V16_is_null,
  COUNTIF(V17 IS NULL) AS V17_is_null,
  COUNTIF(V18 IS NULL) AS V18_is_null,
  COUNTIF(V19 IS NULL) AS V19_is_null,
  COUNTIF(V20 IS NULL) AS V20_is_null,
  COUNTIF(V21 IS NULL) AS V21_is_null,
  COUNTIF(V22 IS NULL) AS V22_is_null,
  COUNTIF(V23 IS NULL) AS V23_is_null,
  COUNTIF(V24 IS NULL) AS V24_is_null,
  COUNTIF(V25 IS NULL) AS V25_is_null,
  COUNTIF(V26 IS NULL) AS V26_is_null,
  COUNTIF(V27 IS NULL) AS V27_is_null,
  COUNTIF(V28 IS NULL) AS V28_is_null,
  COUNTIF(Amount IS NULL) AS Amount_is_null,
  COUNTIF(Class IS NULL) AS Class_is_null
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	null	V24_is_null	V25_is_null	V26_is_null	V27_is_null	V28_is_null	Amount_is_null	Class_is_null
1	0	0	0	0	0	0	0	0

5. Vertex AI:

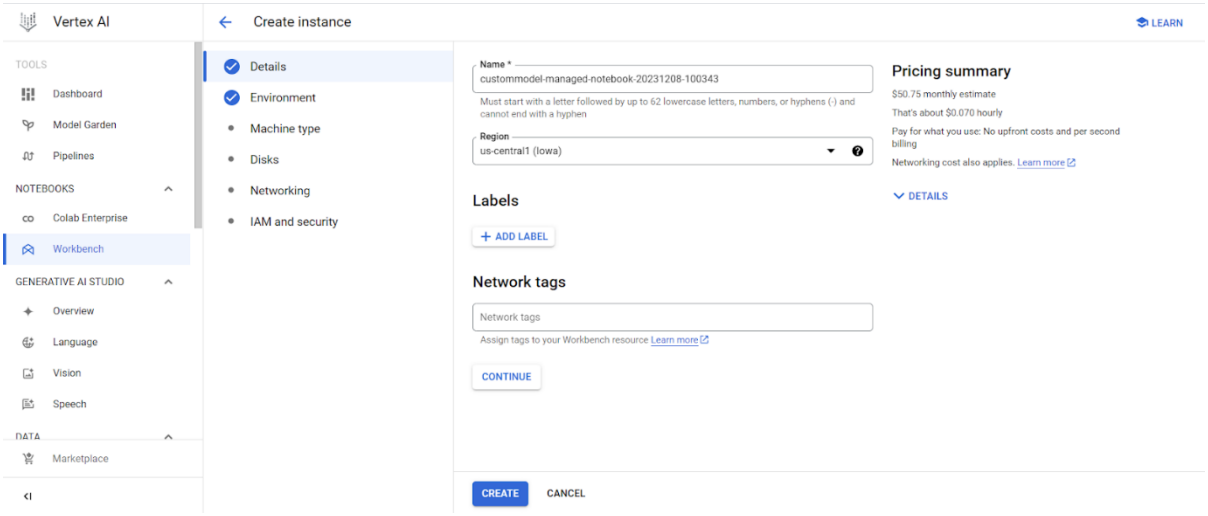
Vertex AI is a machine learning platform that lets GCP users train and deploy ML models and AI based applications. Vertex AI combines data engineering, data science, and ML engineering workflows, enabling your teams to collaborate using a common toolset and scale the applications using the benefits of 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.

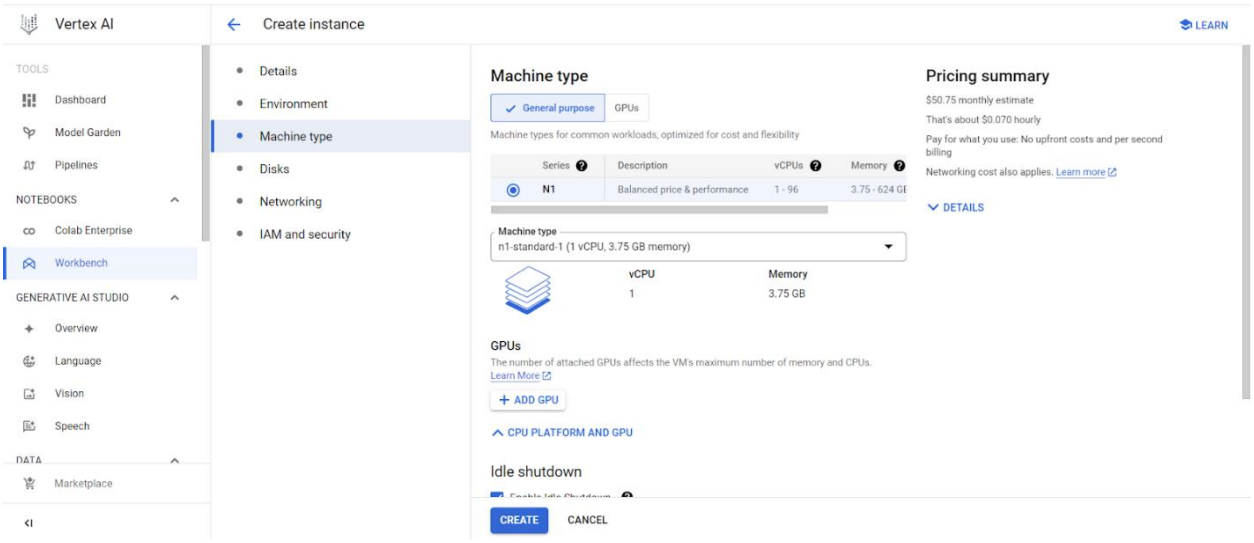
5.1 Vertex AI – Workbench and Custom model training:

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

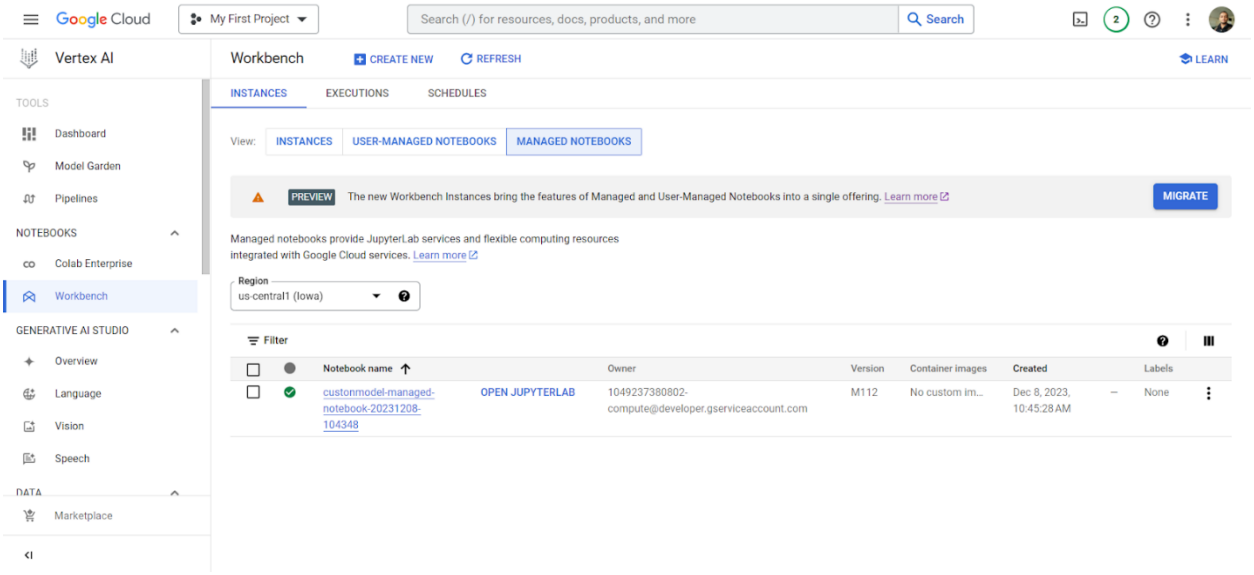
1. First we create a workbench instance with a name and region.



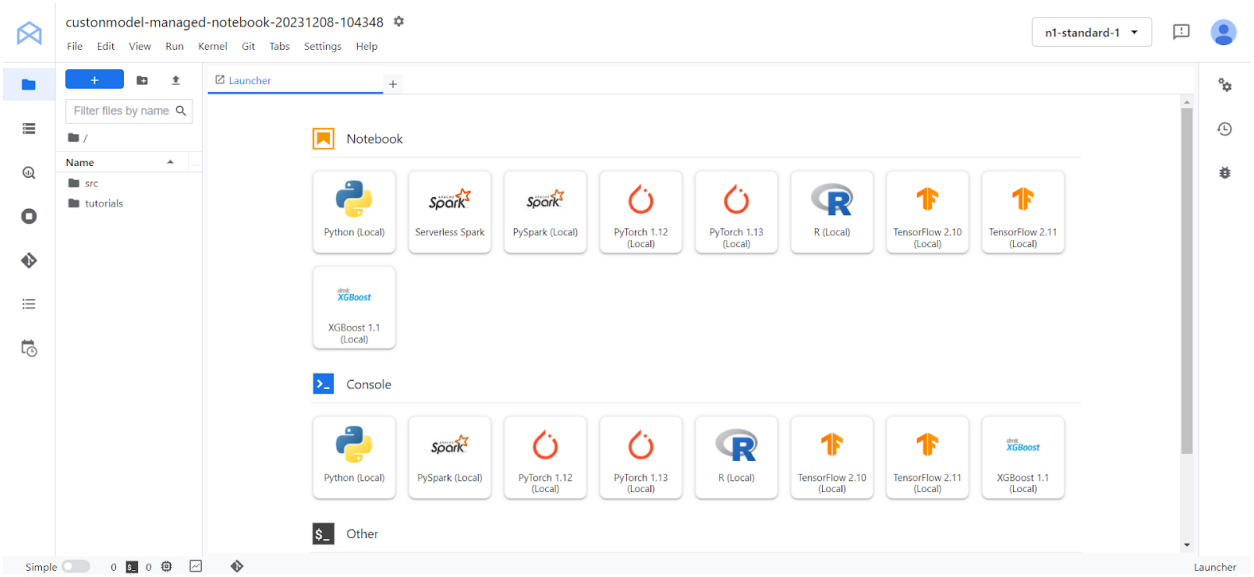
2. Then we select the machine type. We selected the lowest configuration machine for cost-saving purposes.



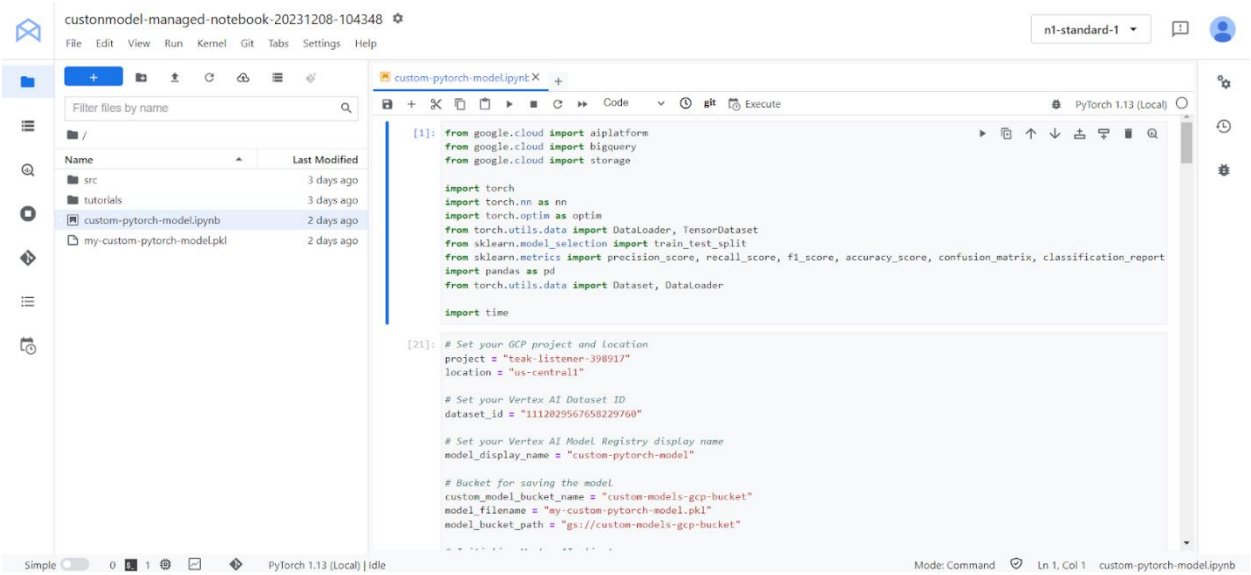
3. Now we can create our instance.



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



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



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, :, :]  # Take the hidden state from the last time step
        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.

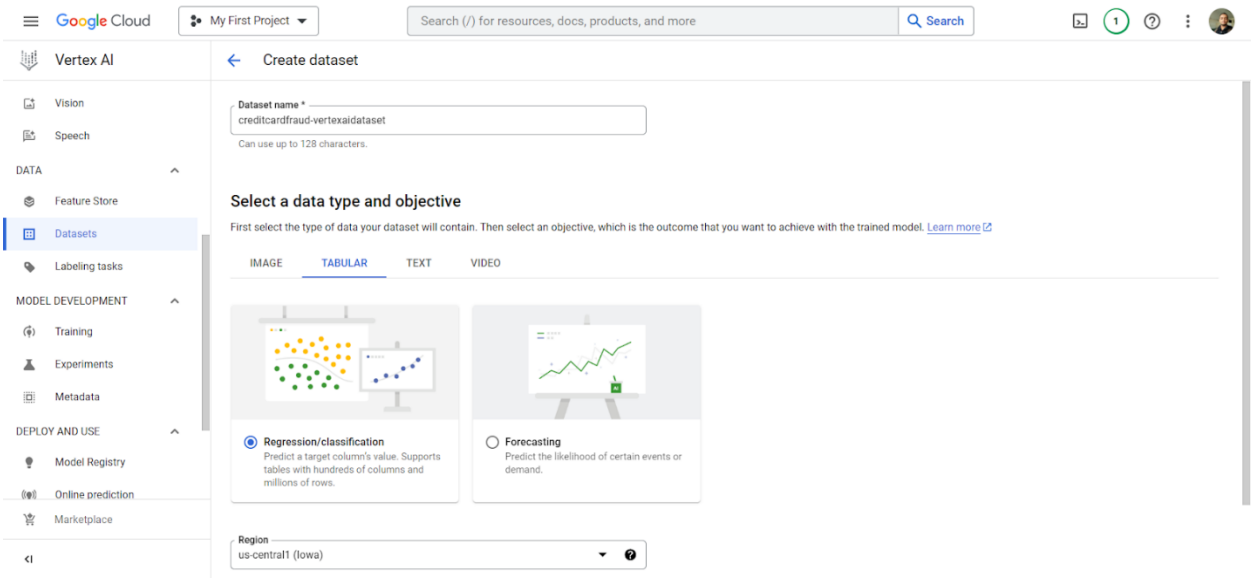
```
Epoch 1/10 | Loss: 0.6034 | Accuracy: 100.00%
Epoch 2/10 | Loss: 0.3966 | Accuracy: 100.00%
Epoch 3/10 | Loss: 0.3168 | Accuracy: 100.00%
Epoch 4/10 | Loss: 0.2945 | Accuracy: 100.00%
Epoch 5/10 | Loss: 0.2809 | Accuracy: 100.00%
Epoch 6/10 | Loss: 0.2958 | Accuracy: 100.00%
Epoch 7/10 | Loss: 0.2664 | Accuracy: 100.00%
Epoch 8/10 | Loss: 0.2666 | Accuracy: 100.00%
Epoch 9/10 | Loss: 0.2536 | Accuracy: 100.00%
Epoch 10/10 | Loss: 0.2363 | Accuracy: 100.00%


Total training time (s): 1508.1906161308289
```

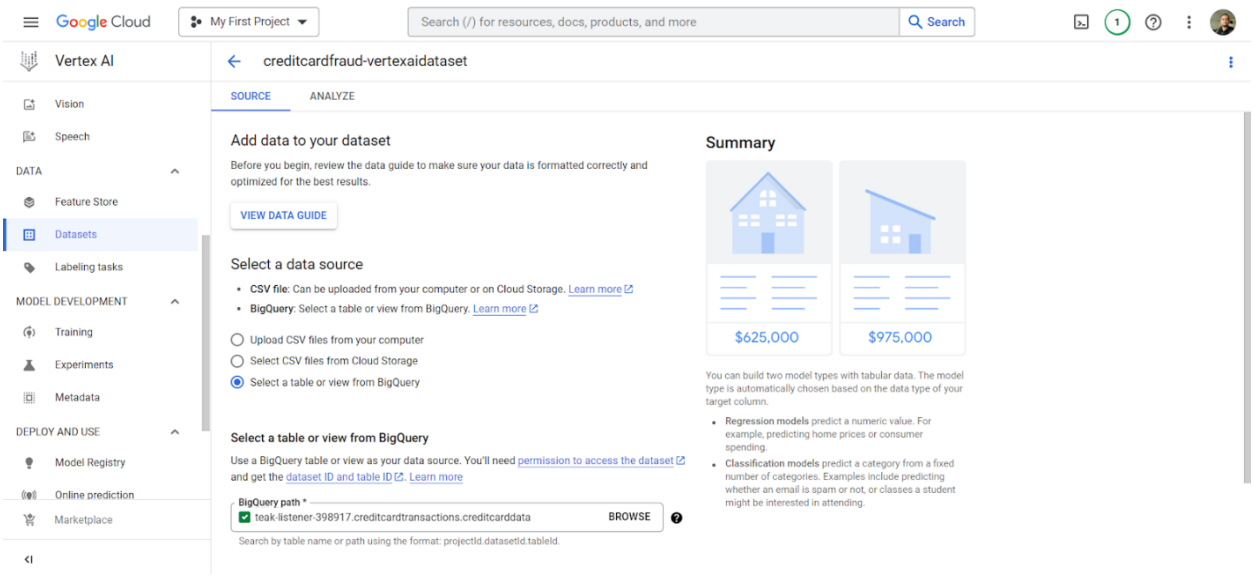
```
Test Accuracy: 94.11%
Confusion Matrix:
[[56618   245]
 [ 6451 50412]]
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.90      1.00      0.94     56863
     Class 1       1.00      0.89      0.94     56863

    accuracy                           0.94    113726
   macro avg       0.95      0.94      0.94    113726
weighted avg       0.95      0.94      0.94    113726
```

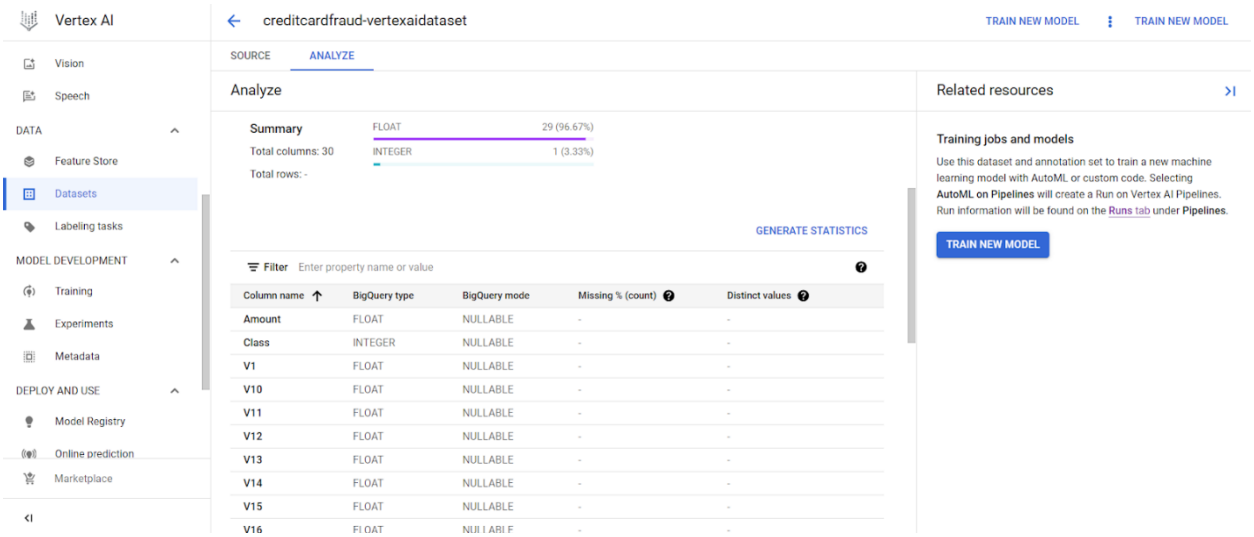
5.2 Vertex AI – AutoML:

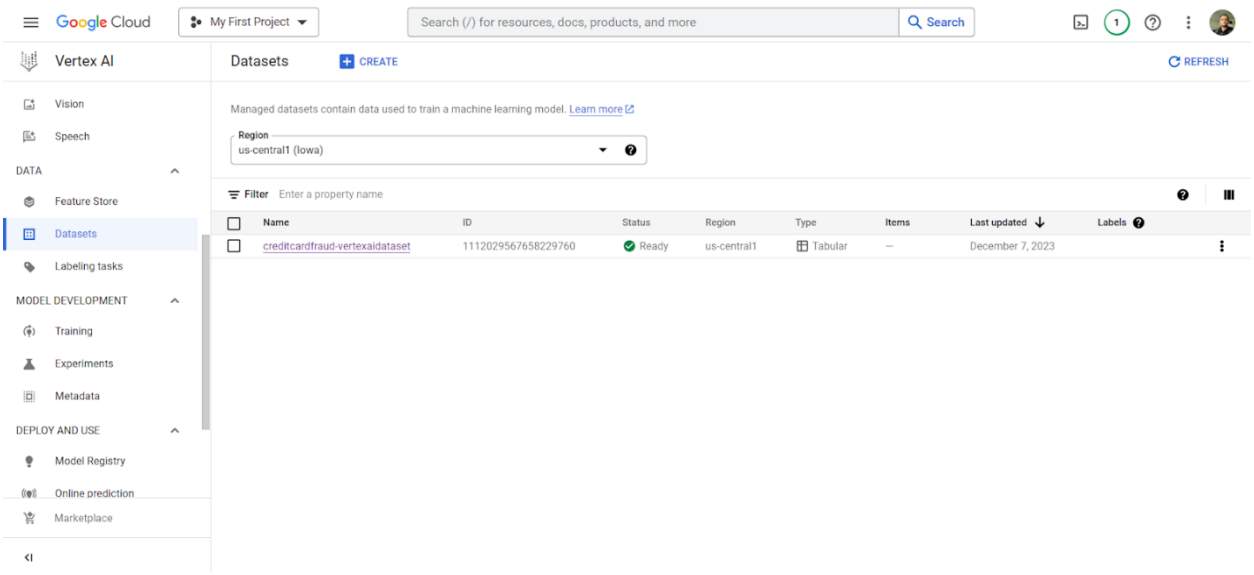
AutoML is a part of Vertex AI that makes it easy to create a classification model. Vertex AI AutoML simplifies the process of building machine learning models by automating various tasks that would typically require expertise in machine learning and data science. To use AutoML, we just need to specify 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.

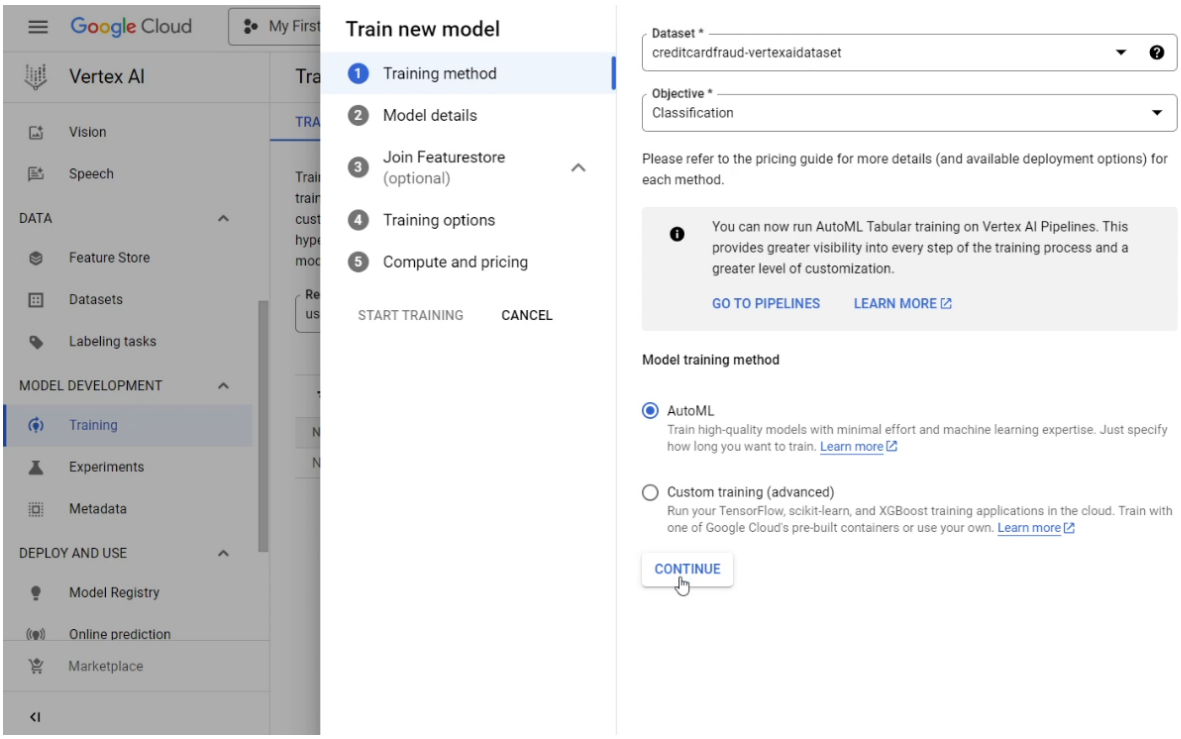



Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
Vertex AI
creditcardfraud-vertexaidataset
SOURCE
ANALYZE
Vision
Speech
DATA
Feature Store
Datasets
Labeling tasks
MODEL DEVELOPMENT
Training
Experiments
Metadata
DEPLOY AND USE
Model Registry
Online prediction
Marketplace
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
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.
Summary
$625,000
$975,000
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.

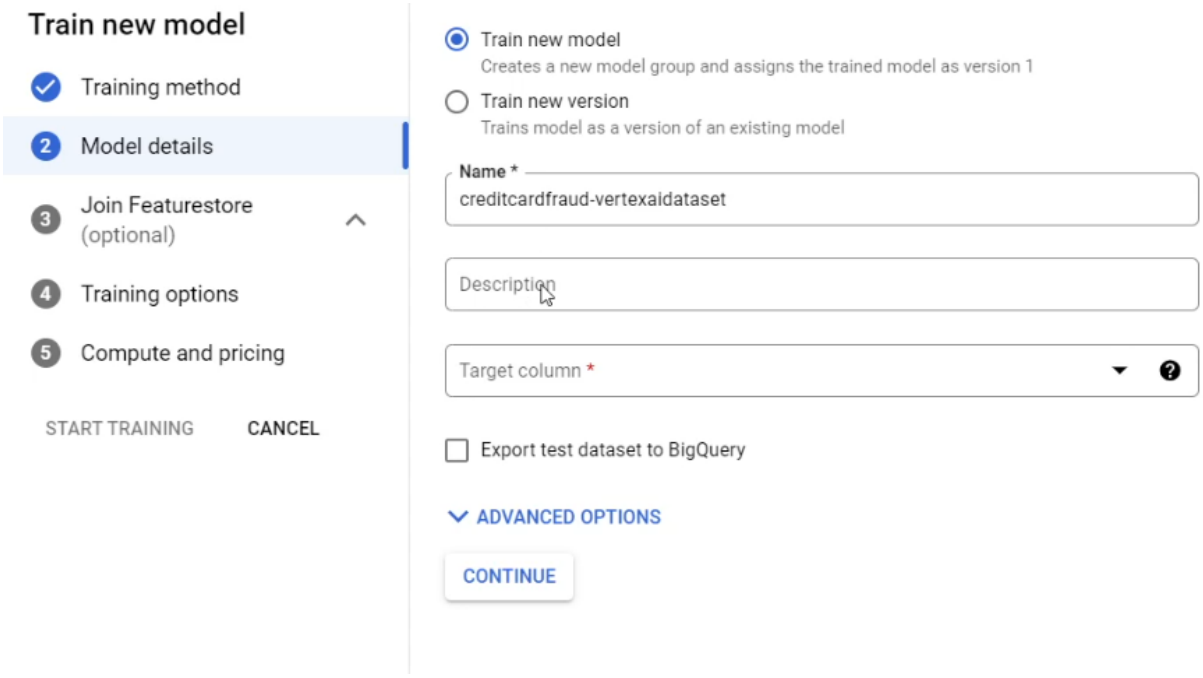

Vertex AI
creditcardfraud-vertexaidataset
TRAIN NEW MODEL
TRAIN NEW MODEL
SOURCE
ANALYZE
Vision
Speech
DATA
Feature Store
Datasets
Labeling tasks
MODEL DEVELOPMENT
Training
Experiments
Metadata
DEPLOY AND USE
Model Registry
Online prediction
Marketplace
Analyze
Summary
Total columns: 30
Total rows: -
FLOAT
29 (96.67%)
INTEGER
1 (3.33%)
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



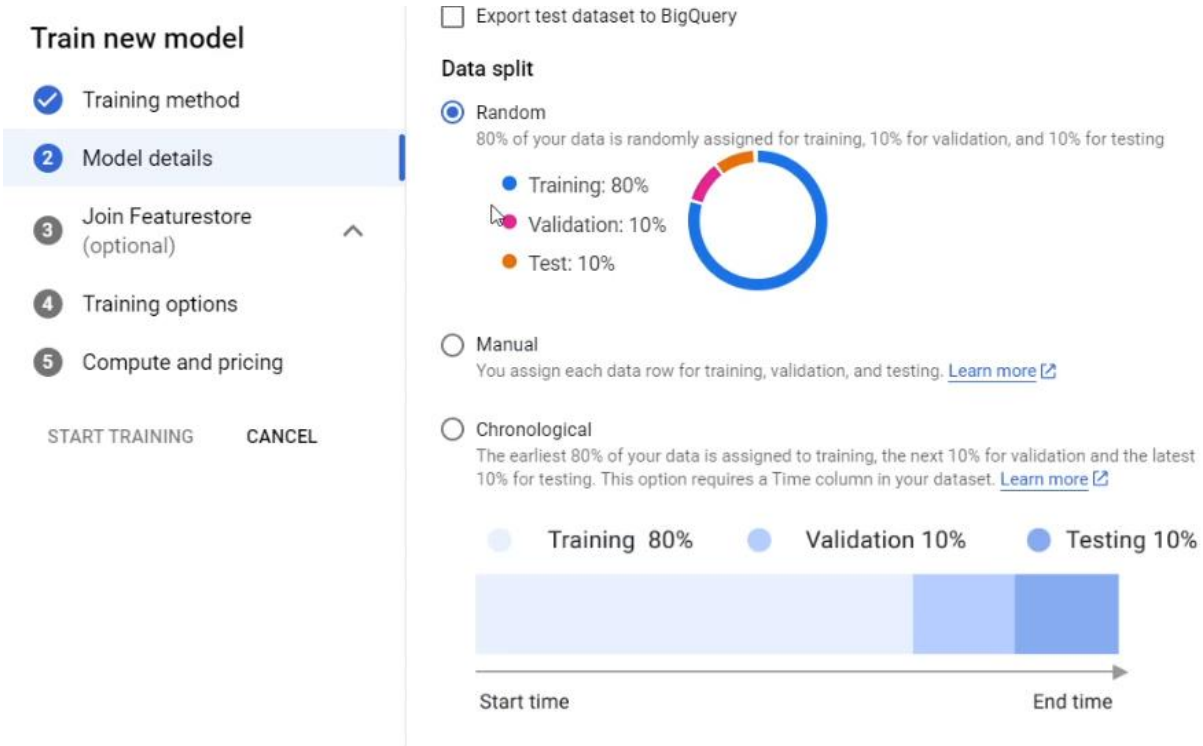
Then with the Vertex AI Dataset ready, we can start using AutoML. First, we need to select the dataset and tell it that we want a classification model.



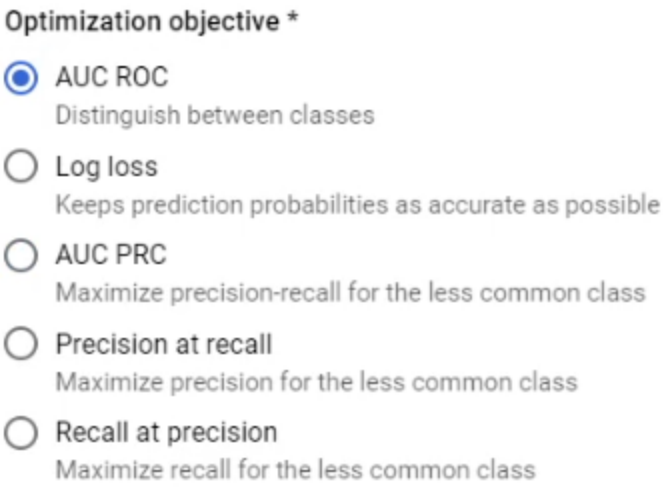
Then we can give it a name and create a new model.



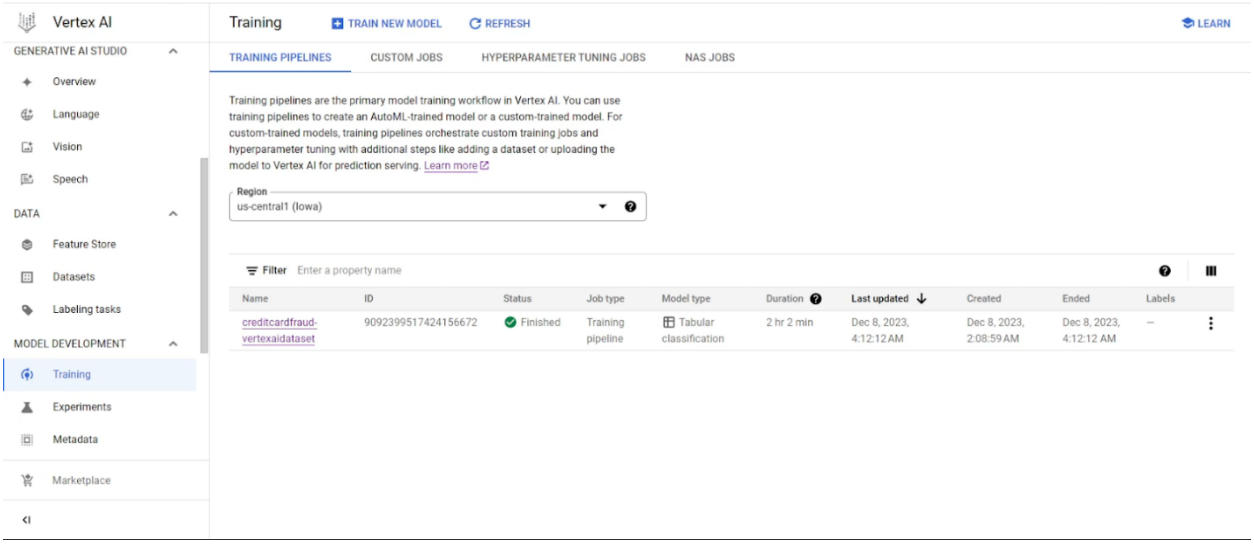
After that we can decide how we want to split our dataset, which we have as an 80-10-10 split for training, evaluation and testing.



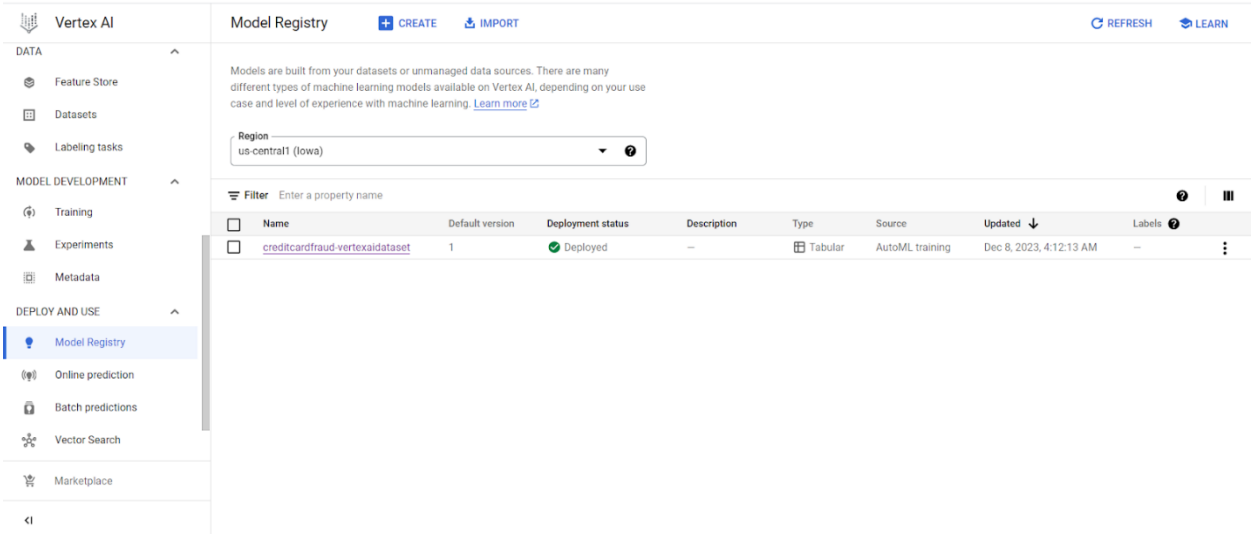
Then we decide how we want our model to be optimized for, which we selected as AUC-ROC.



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

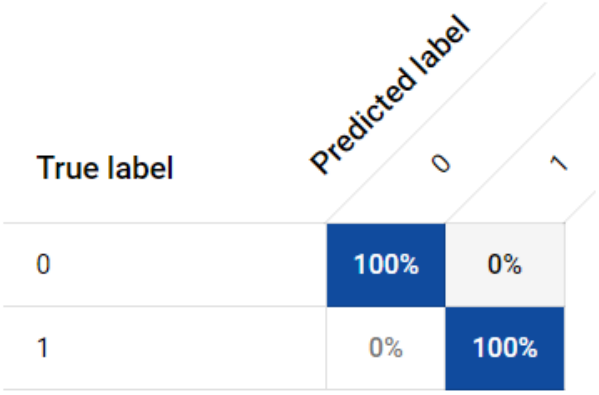


After training we can see the status of our model which is saved in vertex AI Model Registry.



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

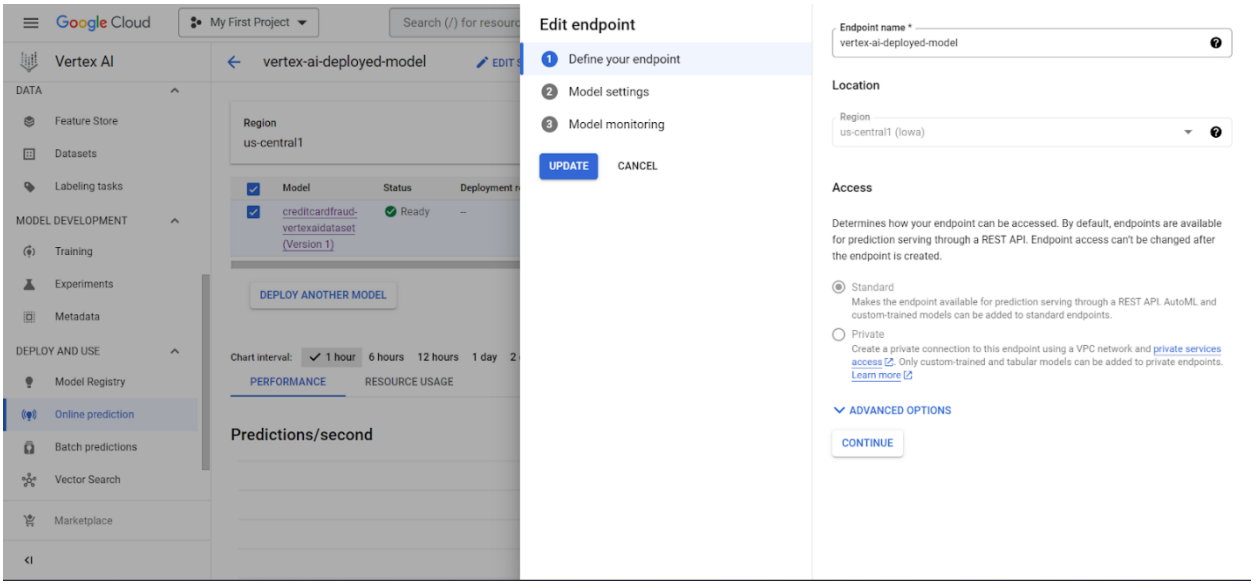
PR AUC	1
ROC AUC	1
Log loss	0.002
F1 score	0.99959564
Micro-F1	0.99959564
Macro-F1	0.99959564
Precision	100%
Recall	100%

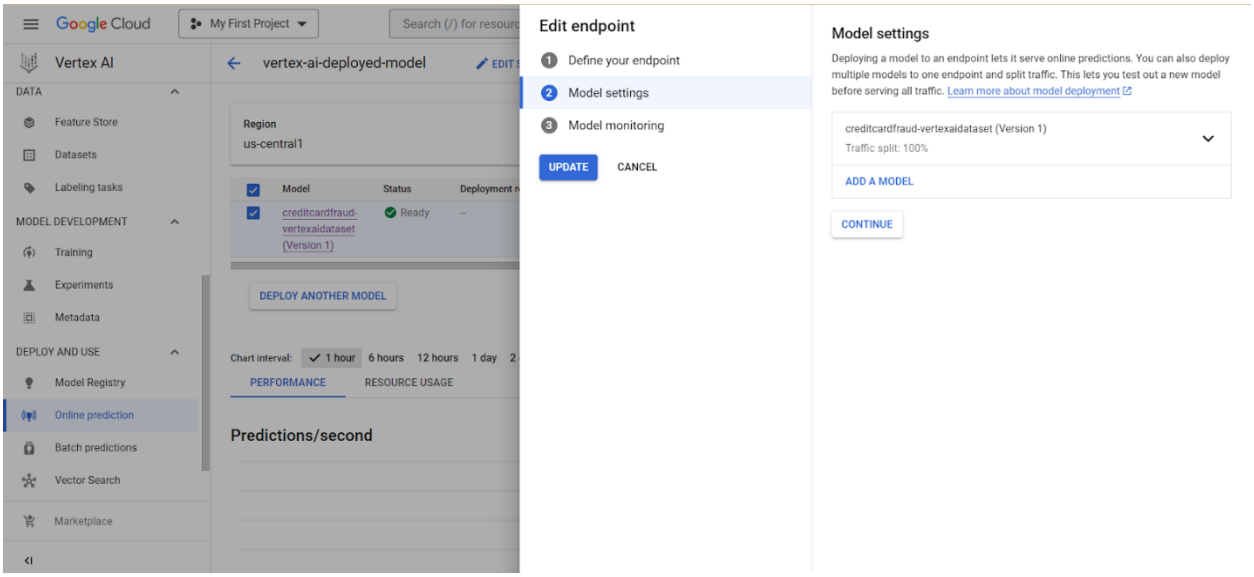


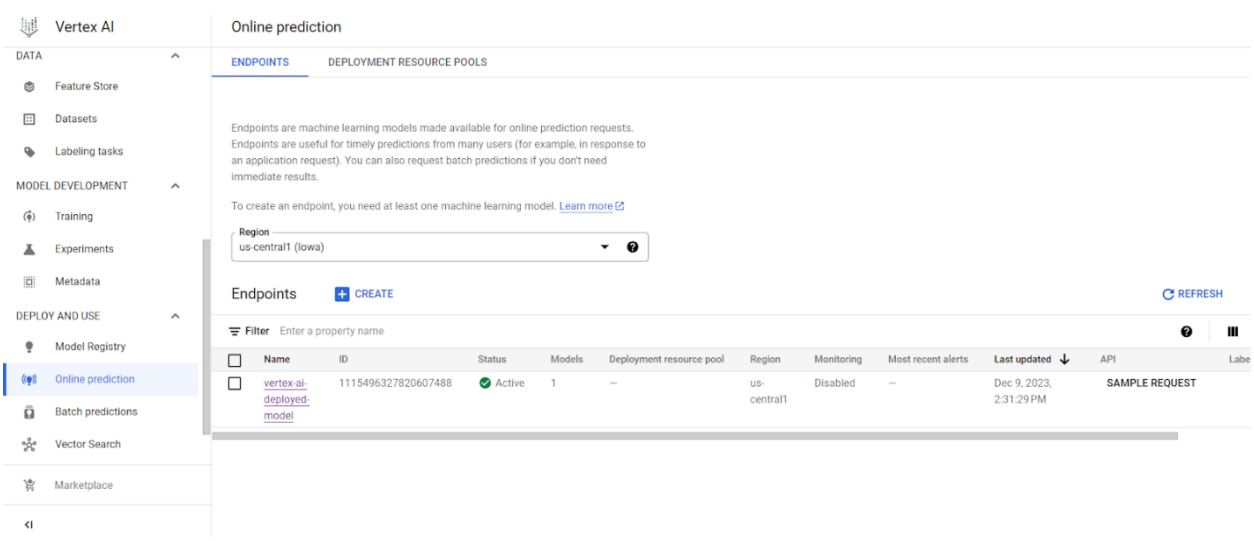
For most common training tasks, such as image and text classification, and classification on tabular data, Google recommends using AutoML and custom training for specialized use cases. The results of the AutoML compared with our custom model are proof of why we should use AutoML for training models for 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:

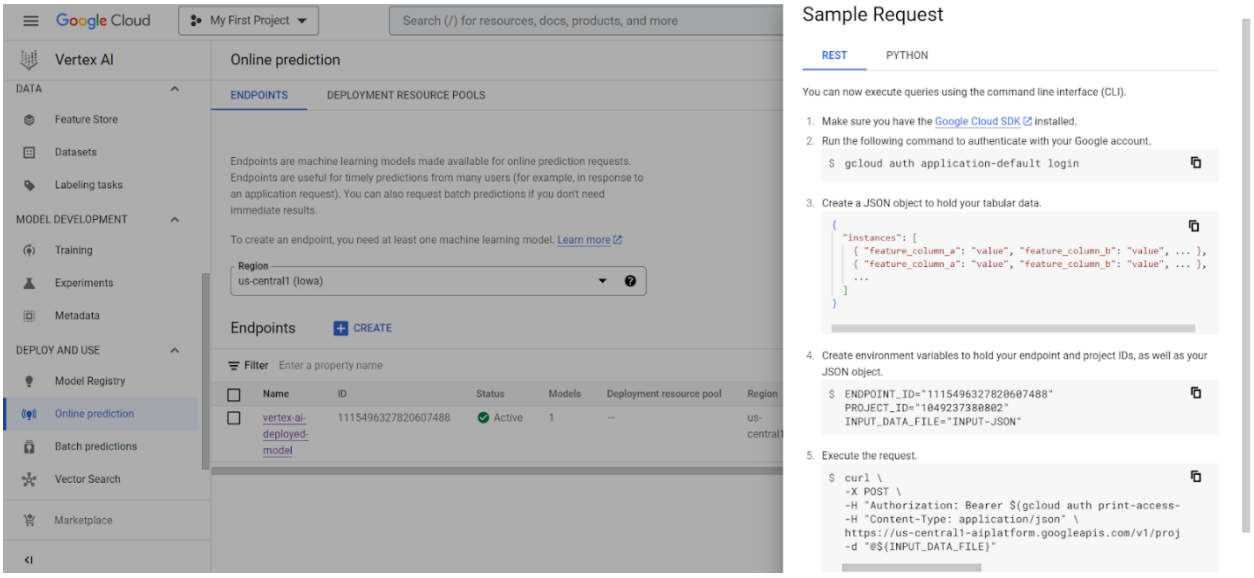
Vertex AI Endpoint is yet another useful part of Vertex AI where we can easily deploy machine learning models in a few clicks. All we need to do is to select the model that have been deployed to the Vertex AI Model Registry and press deploy.

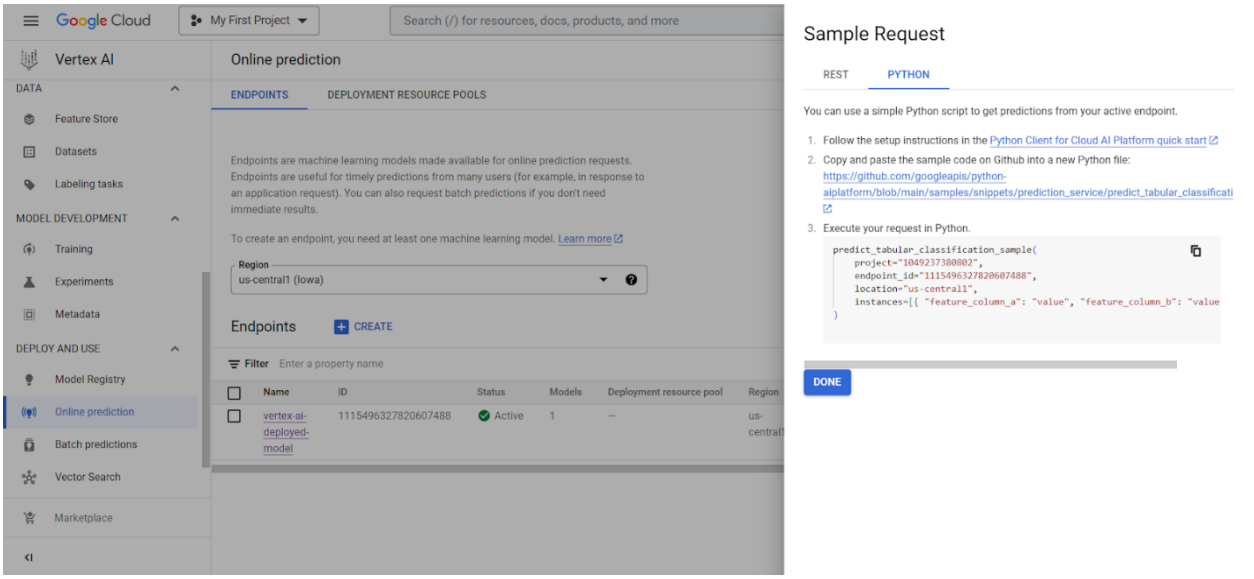

Google Cloud
My First Project
Search (/) for resources
Vertex AI
vertex-ai-deployed-model
EDIT
DATA
Feature Store
Datasets
Labeling tasks
MODEL DEVELOPMENT
Training
Experiments
Metadata
DEPLOY AND USE
Model Registry
Online prediction
Batch predictions
Vector Search
Marketplace
Region
us-central1
Model
Status
creditcardfraud-vertexaidataset (Version 1)
Ready
DEPLOY ANOTHER MODEL
Chart interval:
1 hour
6 hours
12 hours
1 day
PERFORMANCE
RESOURCE USAGE
Predictions/second
Edit endpoint
1 Define your endpoint
2 Model settings
3 Model monitoring
UPDATE
CANCEL
Endpoint name *
vertex-ai-deployed-model
Location
Region
us-central1 (Iowa)
Access
Determines how your endpoint can be accessed. By default, endpoints are available for prediction serving through a REST API. Endpoint access can't be changed after the endpoint is created.
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


Google Cloud
My First Project
Search (/) for resources
Vertex AI
vertex-ai-deployed-model
EDIT
DATA
Feature Store
Datasets
Labeling tasks
MODEL DEVELOPMENT
Training
Experiments
Metadata
DEPLOY AND USE
Model Registry
Online prediction
Batch predictions
Vector Search
Marketplace
Region
us-central1
Model
Status
creditcardfraud-vertexaidataset (Version 1)
Ready
DEPLOY ANOTHER MODEL
Chart interval:
1 hour
6 hours
12 hours
1 day
PERFORMANCE
RESOURCE USAGE
Predictions/second
Edit endpoint
1 Define your endpoint
2 Model settings
3 Model monitoring
UPDATE
CANCEL
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
creditcardfraud-vertexaidataset (Version 1)
Traffic split: 100%
ADD A MODEL
CONTINUE


Vertex AI
Online prediction
DATA
Feature Store
Datasets
Labeling tasks
MODEL DEVELOPMENT
Training
Experiments
Metadata
DEPLOY AND USE
Model Registry
Online prediction
Batch predictions
Vector Search
Marketplace
ENDPOINTS
DEPLOYMENT RESOURCE POOLS
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
Filter
Enter a property name
Name
ID
Status
Models
Deployment resource pool
Region
Monitoring
Most recent alerts
Last updated
API
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.





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/nelgiriyewithana/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-. Cai/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