```
!pip install -q kaggle
```

Upload your kaggle.json file using the kaggle API in Google Colab

```
from google.colab import files

files.upload()
```

This cell creates a .kaggle hidden folder in our root directory and copies the kaggle.json file to that folder.

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 /root/.kaggle/kaggle.json
```

This cell will download the dataset used for this project to the /content directory in your Google Colab instance and also unzip the csv file.

```
!kaggle datasets download -d ealaxi/paysim1
!unzip /content/paysim1.zip
```

After we've installed the requirements and loaded the data, we can then do some analysis on the dataset.

Data Analysis

```
# Importing some important libraries
import pandas as pd
import numpy as np

df = pd.read_csv('PS_20174392719_1491204439457_log.csv')
pd.set_option('display.max.columns', None)
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

```
df.columns
```

```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

Screenshot by Author: Loaded Data

We can see above that the dataset contains a total of 11 columns and more than 6 million rows.

```
len_not_fraud =
len(df['isFraud'][df.isFraud
== 0])
                            len_is_fraud = len(df['isFraud'][df.isFraud == 1])

                            arr = np.array([len_not_fraud, len_is_fraud])
                            labels = ['Not Fraudulent', 'Fraudulent']
                            print(f"Total No. of Non-Fraudulent Cases: {len_not_fraud}")
                            print(f"Total No. Fraudulent Cases: {len_is_fraud}")
```

```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

We can see that the number of fraudulent cases in the dataset is really small with only 8213 transactions out of over 6 million transactions is tagged as fraud. Therefore the dataset is highly imbalanced and some workarounds we have to be applied to fix this problem.

Screenshot by Author: Number of fraudulent cases.

The dataset is also highly imbalanced with only 0.12% of transactions tagged as fraudulent cases. Some workarounds can be done when working with imbalanced data such as:

- Synthetic Minority Oversampling Technique (SMOTE)
- Undersampling

But we won't be applying any technique here for the sake of simplicity. You can read about SMOTE, undersampling and working with imbalanced data by checking out the links below;

Credit Card Fraud Detection : How to handle Imbalanced Dataset
This post will be focused on the stey-by-step project and the result, you can view my code in my Github.
medium.com

Imbalanced Classification with the Fraudulent Credit Card Transactions Dataset - Machine Learning…
Fraud is a major problem for credit card companies, both because of the large volume of transactions that are completed…
machinelearningmastery.com

Feature Engineering

```
df.drop(['nameOrig', 'nameDest'], axis=1, inplace=True)
data = df.copy(deep = True)
# get all categorical columns in the dataframe
catCols = [col for col in data.columns if data[col].dtype=="O"]

from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()

for item in catCols:
    data[item] = lb_make.fit_transform(data[item])
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Screenshot by Author: Feature Selection

The image above shows the columns in the dataset. This is the description of each column:

- step column: Number of hours it took for a transaction to complete.
- type column: Type of transaction that took place. There are 5 categories in this column namely; ‘PAYMENT’, ‘TRANSFER’, ‘CASH_OUT’, ‘DEBIT’, ‘CASH_IN’ .
- nameOrig: Name/ID of the Sender.
- oldbalanceOrg: Sender balance before the transaction took place.
- newbalanceOrg: Sender balance after the transaction took place.
- nameDest: Name/ID of the Recipient.
- oldbalanceDest: Recipient balance before the transaction took place.
- newbalanceDest: Recipient balance after the transaction took place.

- isFraud: This is the transactions made by the fraudulent agents inside the simulation.
- isFlaggedFraud: The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

We will be dropping the ‘nameOrig’ and ’nameDest’ columns and also converting the ‘type’ column from categorical data to numerical data using label encoding.

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	3	9839.64	170136.0	160296.36	0.0	0.0	0	0
1	1	3	1864.28	21249.0	19384.72	0.0	0.0	0	0
2	1	4	181.00	181.0	0.00	0.0	0.0	1	0
3	1	1	181.00	181.0	0.00	21182.0	0.0	1	0
4	1	3	11668.14	41554.0	29885.86	0.0	0.0	0	0

The type column has now been converted from a categorical datatype to numerical datatype using label encoding.

- PAYMENT = 3
- TRANSFER = 4
- CASH_OUT = 1
- DEBIT = 2
- CASH_IN = 0

Screenshot by Author: Label Encoding

We will also be creating an “evaluate_model” function that will be used to get the metric scores of our models for evaluation. The metrics used are:

- Accuracy Score: Accuracy is a metric used in classification problems used to tell the percentage of

accurate predictions. We calculate it by dividing the number of correct predictions by the total number of predictions.

- Precision Score: Precision quantifies the number of positive class predictions that actually belong to the positive class.
- Recall Score: Recall quantifies the number of positive class predictions made out of all positive examples in the dataset.
- F1 Score: F-Measure or F1 Score provides a single score that balances both the concerns of precision and recall in one number.
- Confusion Matrix: A confusion matrix is **a technique for summarizing the performance of a classification algorithm**. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

Performance Metrics in Machine Learning [Complete Guide] — neptune.ai

Model Building

Seeing as this project tries to solve a classification problem(detecting whether or not a transaction was fraudulent), we will be building

and evaluating a classification model for our credit card detection app.

You can read more about classification machine learning techniques with the links below:

Classification In Machine Learning | Classification Algorithms | Edureka

4 Types of Classification Tasks in Machine Learning (machinelearningmastery.com)

Logistic Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X = data.drop('isFraud', axis=1)
y = data.isFraud



# setting up testing and training sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=27)

lr = LogisticRegression()
model1 = lr.fit(X_train, y_train)

# Predict on training set
lr_pred = model1.predict(X_test)
evaluate_model(y_test, lr_pred)
```

Logistic regression **estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables**. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

We will split our dataset into training and testing sets and use the LogisticRegression function from Sci-Kit Learn library to predict the outcome of a credit card transaction.

In the code snippet above;

- We set our X variable by dropping the 'isFraud' column because that is our target column.
- The 'isFraud' column is subsequently assigned to our y variable.
- The data was then divided into a training and testing set, and logistic regression was then applied to the training set.
- Using the 'evaluate model' function we previously declared, we then evaluated the model.

```
Accuracy Score:  0.9983421923672953
Precision Score:  0.3843906510851419
Recall Score:  0.442150744119059
F1 Score:  0.4112525117213664
Confusion Matrix:  [[1587097    1475]
 [   1162     921]]
```