Online Payment Fruad Detection using Machine Learning

Project Description:

Online Payments Fraud Detection using Machine Learning is a proactive approach to identify and prevent fraudulent activities during online transactions. By leveraging historical transaction data, customer behavior patterns, and machine learning algorithms, this project aims to detect potential fraud in real time, ensuring secure and trustworthy online payment experiences for users and businesses alike.

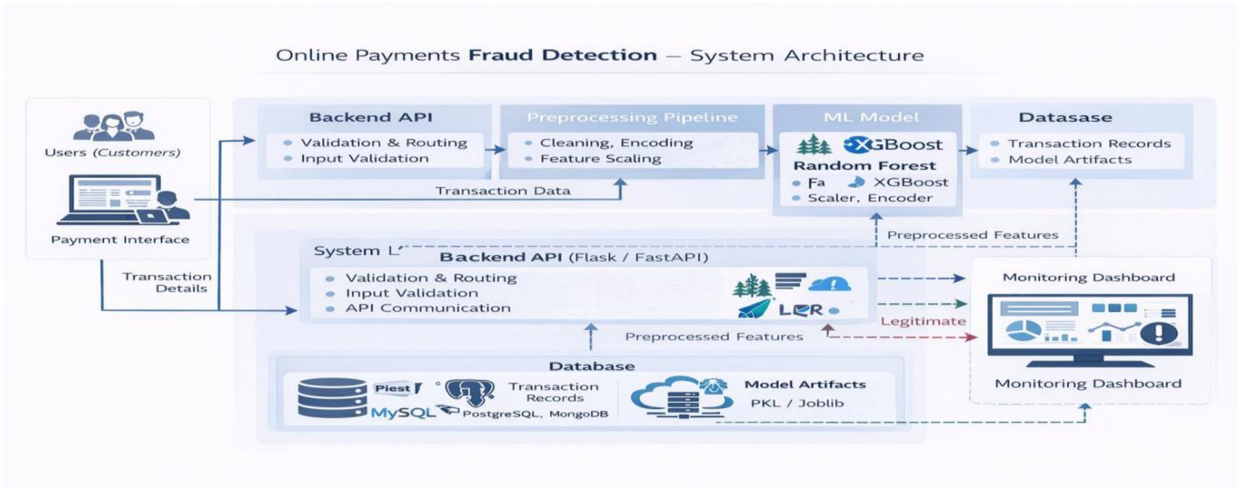
Scenario 1: Real-time Fraud Monitoring The system continuously monitors online payment transactions in real time. By analyzing transaction features such as transaction amount, location, device information, and user behavior, it can flag suspicious transactions for further investigation, preventing fraudulent activities before they occur.

Scenario 2: Fraudulent Account Detection Machine learning models can detect patterns indicative of fraudulent accounts or activities. By analyzing user behavior over time, such as unusual login times, multiple failed login attempts, or sudden changes in spending patterns, the system can identify and block potentially fraudulent accounts, protecting legitimate users and businesses.

Scenario 3: Adaptive Fraud Prevention The system adapts and improves its fraud detection capabilities over time. By continuously learning from new data and adjusting its algorithms, it can stay ahead of evolving fraud techniques and trends, providing ongoing protection against online payment fraud for businesses and their customers

.

Technical Architecture:



Pre requisites:

To complete this project, you must required following software's, concepts and packages

- **Anaconda navigator and Visual Studio Code:**
  - Refer the link below to download anaconda navigator
  - Link : https://youtu.be/1ra4zH2G4o0
  - Refer the link below to download Visual Studio Code
  - Link: https://code.visualstudio.com/
- **Python packages:**
  - Open VS Code terminal or Command Prompt as administrator
  - Type "pip install numpy" and click enter.
  - Type "pip install pandas" and click enter.
  - Type "pip install scikit-learn" and click enter.
  - Type "pip install matplotlib" and click enter.
  - Type "pip install scipy" and click enter.
  - Type "pip install joblib" and click enter.
  - Type "pip install seaborn" and click enter.
  - Type "pip install Flask" and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - Supervised learning: https://youtu.be/QeKshry8pWQ
  - Unsupervised learning: https://youtu.be/D6gtZrsYi6c
  - Metrics : https://youtu.be/aWAnNHXIKww
  - Flask : https://youtu.be/lj4I_CvBnt0

Project Objectives:

By the end of this project :

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding of data.
- Have knowledge of pre-processing the data/transformation techniques and some visualization concepts before building the model
- Learn how to build a machine learning model and tune it for better performance
- Know how to evaluate the model and deploy it using flask

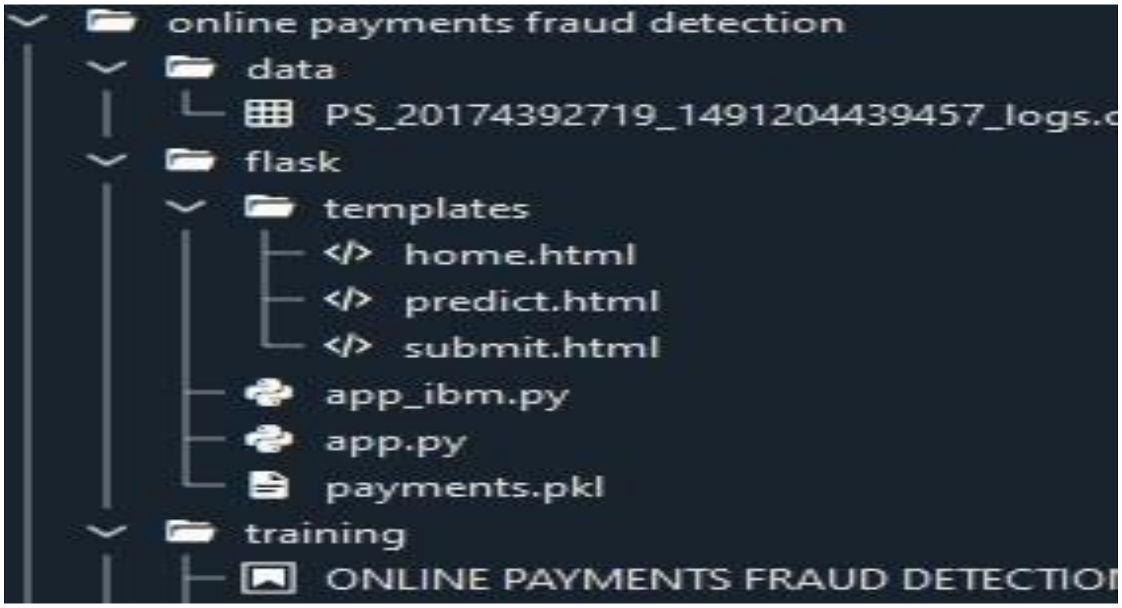
- **Project Flow**:
- The user interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- The predictions made by the model are showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data collection
  - Collect the dataset or create the dataset
- Data pre-processing
  - Removing unnecessary columns
  - Checking for null values
- Visualizing and analyzing data
  - Univariate analysis
  - Bivariate analysis
  - Descriptive analysis
- Model building
  - Handling categorical values
  - Dividing data into train and test sets
  - Import the model building libraries
  - Comparing the accuracy of various models
  - Hyperparameter tuning of the selected model
  - Evaluating the performance of models
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

Project Structure:

Create the Project folder which contains files as shown below



```
online payments fraud detection
  data
    PS_20174392719_1491204439457_logs.c
  flask
    templates
      home.html
      predict.html
      submit.html
    app_ibm.py
    app.py
    payments.pkl
  training
    ONLINE PAYMENTS FRAUD DETECTIO
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains model training files and the training_ibm folder contains IBM deployment files.

Milestone 1: Data Collection

ML depends heavily on data, It is most crucial aspect that makes algorithm training possible.
So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used PS_20174392719_1491204439457_logs.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset

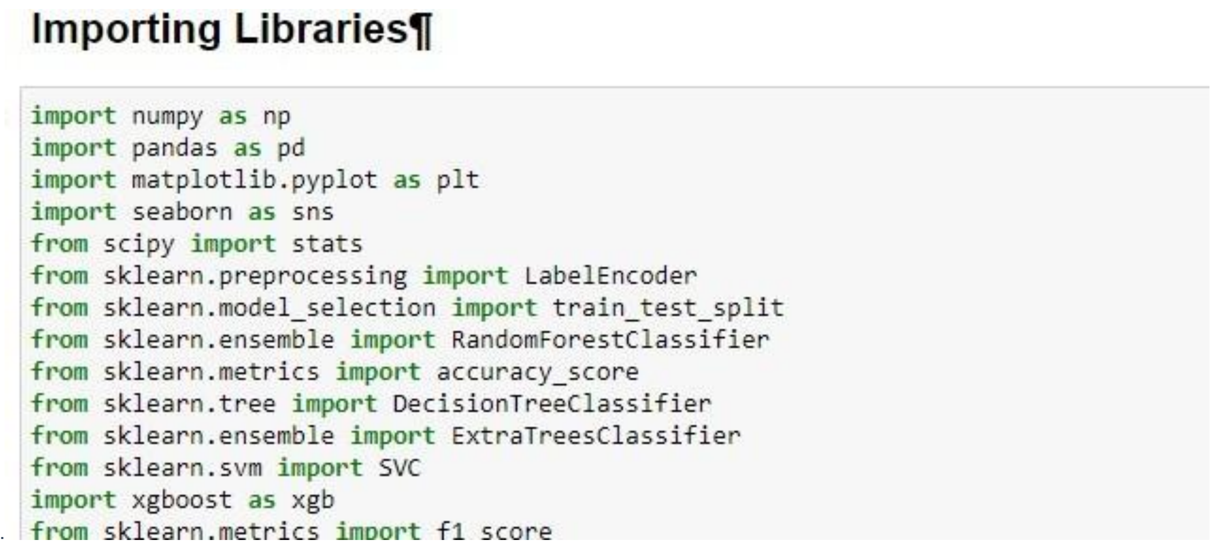
Milestone 2: Visualizing and analysing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight



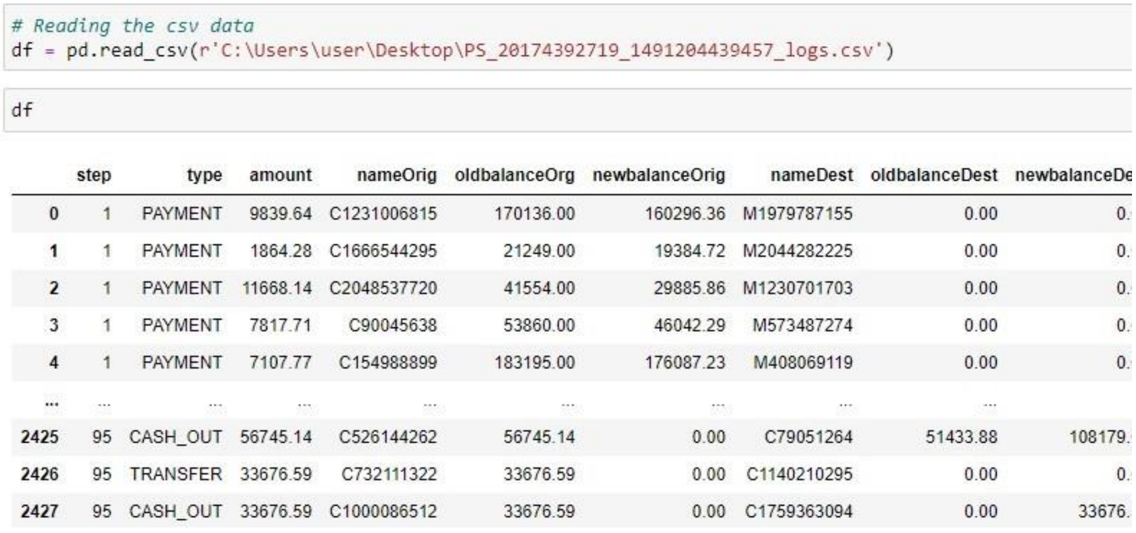
Importing Libraries¶

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.metrics import f1_score
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

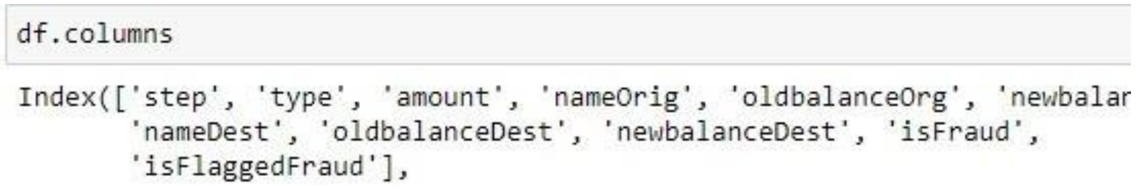
In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.



```python
# Reading the csv data
df = pd.read_csv(r'C:\Users\user\Desktop\PS_20174392719_1491204439457_logs.csv')
```

```python
df
```

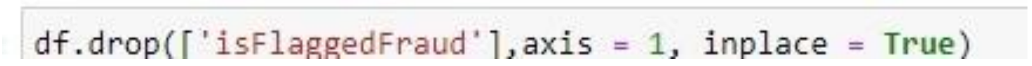
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDe
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.
...	...	...	...	...	...	...	...	...	
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.



```python
df.columns
```

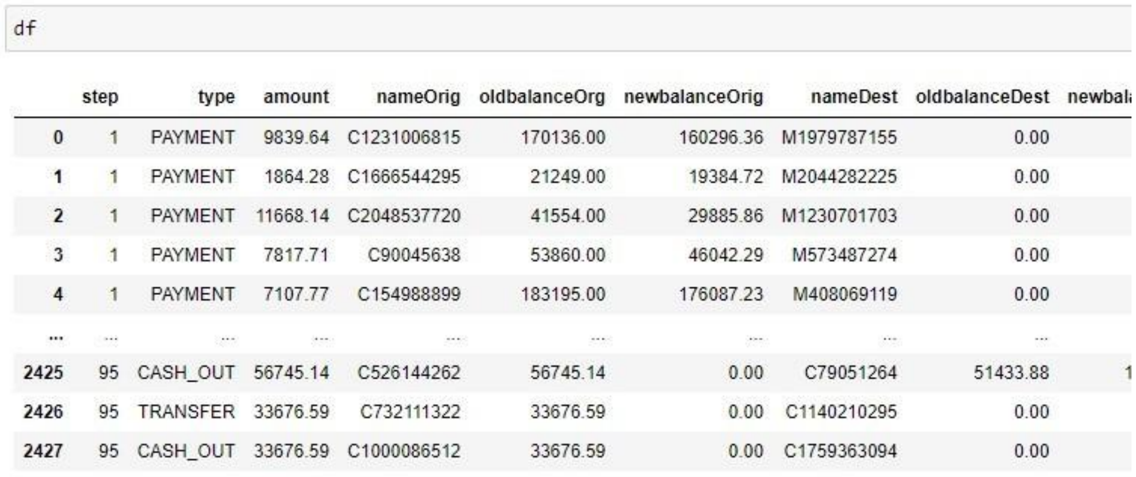
```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalar
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
```

Here, the input features in the dataset are known using the df.columns function.



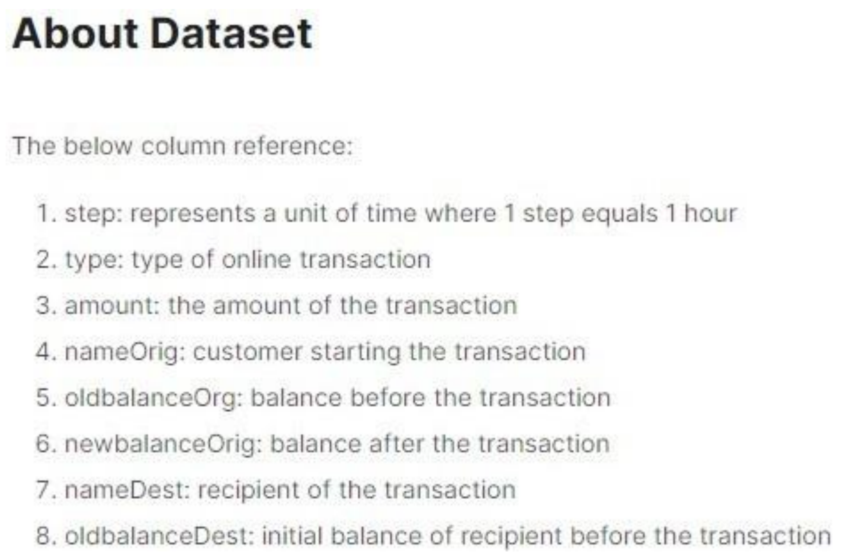
```python
df.drop(['isFlaggedFraud'],axis = 1, inplace = True)
```

here, the dataset's superfluous columns are being removed using the drop method.



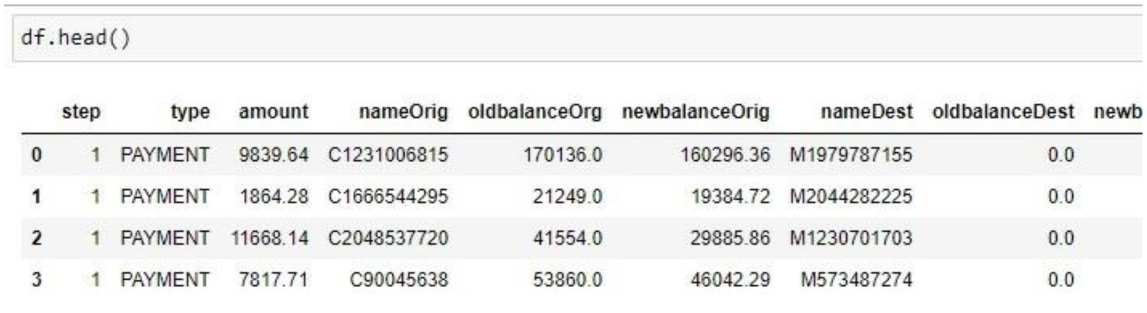
```python
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbal
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	
...	...	...	...	...	...	...	...	...	
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	1
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	


About Dataset

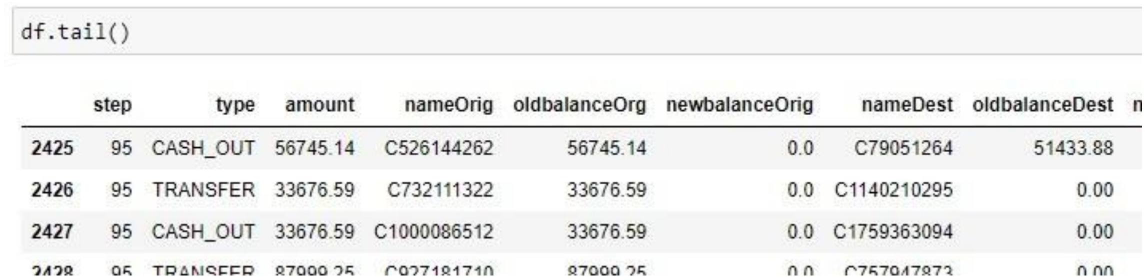
The below column reference:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction


```
df.head()
```

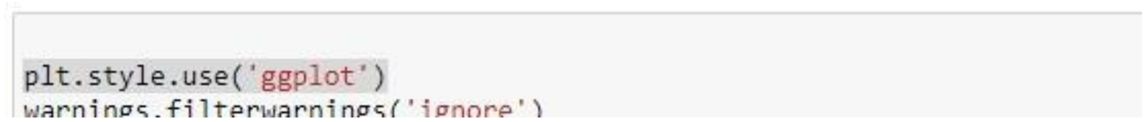
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newb
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	
2	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	
3	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	

above, the dataset's first five values are loaded using the head method.

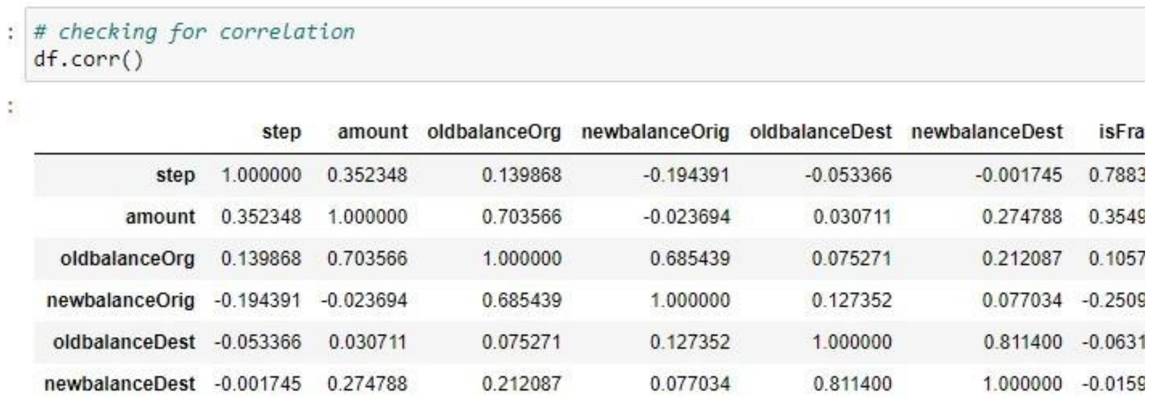

```
df.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	n
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.0	C79051264	51433.88	
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.0	C1140210295	0.00	
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.0	C1759363094	0.00	
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.0	C757947873	0.00	

above, the dataset's last five values are loaded using the tail method.


```
plt.style.use('ggplot')
warnings.filterwarnings('ignore')
```

utilising Style use here The Ggplot approach Setting "styles"—basically stylesheets that resemble matplotlibrc files—is a fundamental feature of mpltools. The "ggplot" style, which modifies the style to resemble ggplot, is demonstrated in this dataset.



```
# checking for correlation
df.corr()
```

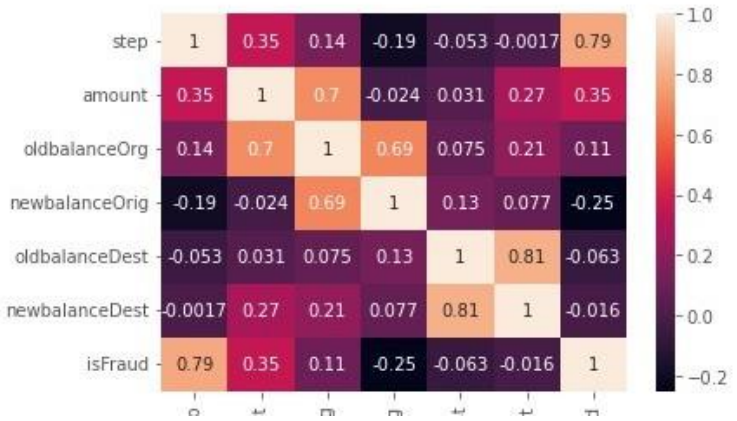
	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFra
step	1.000000	0.352348	0.139868	-0.194391	-0.053366	-0.001745	0.7883
amount	0.352348	1.000000	0.703566	-0.023694	0.030711	0.274788	0.3549
oldbalanceOrg	0.139868	0.703566	1.000000	0.685439	0.075271	0.212087	0.1057
newbalanceOrig	-0.194391	-0.023694	0.685439	1.000000	0.127352	0.077034	-0.2509
oldbalanceDest	-0.053366	0.030711	0.075271	0.127352	1.000000	0.811400	-0.0631
newbalanceDest	-0.001745	0.274788	0.212087	0.077034	0.811400	1.000000	-0.0159

utilising the corr function to examine the dataset's correlation

Heatmap

```
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



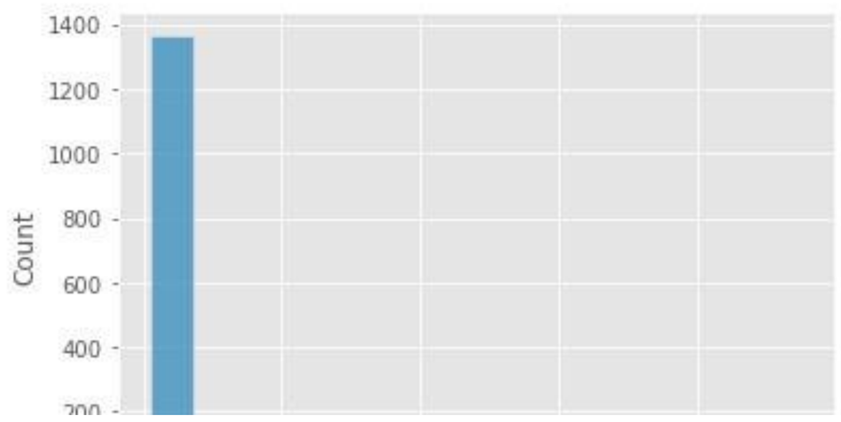
Here, a heatmap is used to understand the relationship between the input attributes and the anticipated goal value.

Activity 3: Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature. Here I have displayed the graph such as histplot .

```
#step
sns.histplot(data=df,x='step')
```

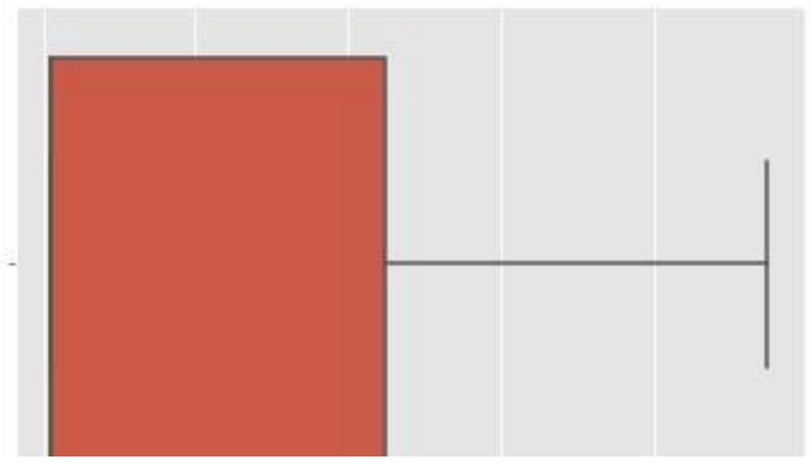
```
<AxesSubplot:xlabel='step', ylabel='Count'>
```



The distribution of one or more variables is represented by a histogram, a traditional visualisation tool, by counting the number of observations that fall within.

```
sns.boxplot(data=df,x='step')
```

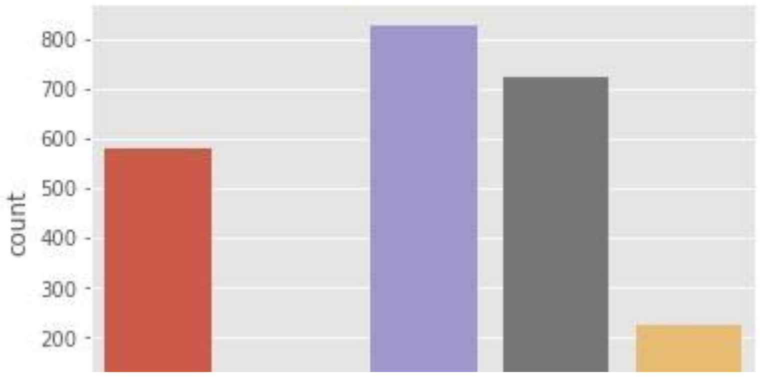
```
<AxesSubplot:xlabel='step'>
```



Here, the relationship between the step attribute and the boxplot is visualised.

```
#type
sns.countplot(data=df,x='type')
```

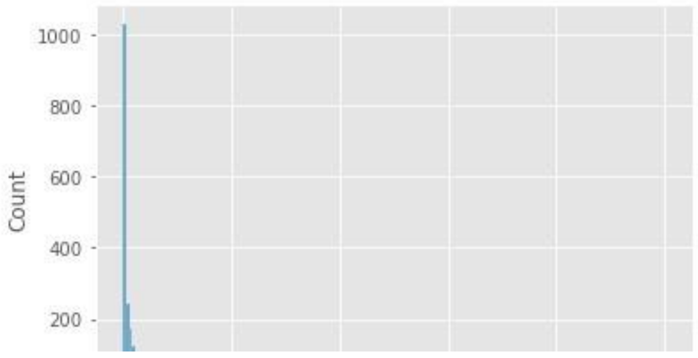
```
<AxesSubplot:xlabel='type', ylabel='count'>
```



Here, the counts of observations in the type attribute of the dataset will be displayed using a countplot.

```
#amount
sns.histplot(data=df,x='amount')
```

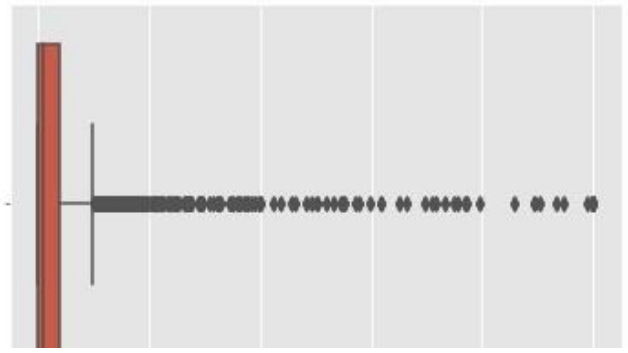
```
<AxesSubplot:xlabel='amount', ylabel='Count'>
```



By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the amount attribute in the dataset.

```
#amount
sns.boxplot(data=df,x='amount')
```

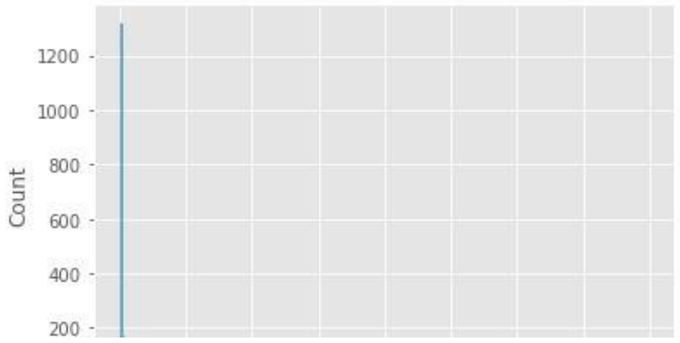
```
<AxesSubplot:xlabel='amount'>
```



Here, the relationship between the amount attribute and the boxplot is visualised.

```
#oldbalanceOrg
sns.histplot(data=df,x='oldbalanceOrg')
```

```
<AxesSubplot:xlabel='oldbalanceOrg', ylabel='Count'>
```

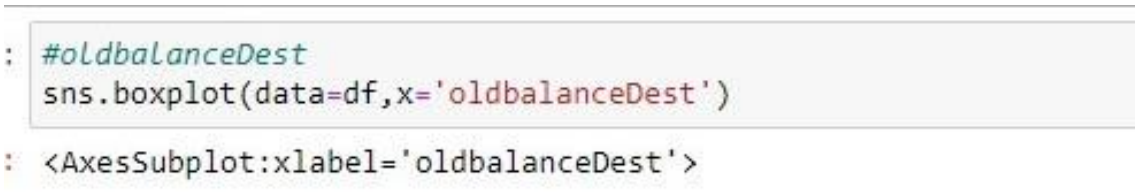


By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the oldbalanceOrg attribute in the dataset.

```
#nameDest
df['nameDest'].value_counts()
```

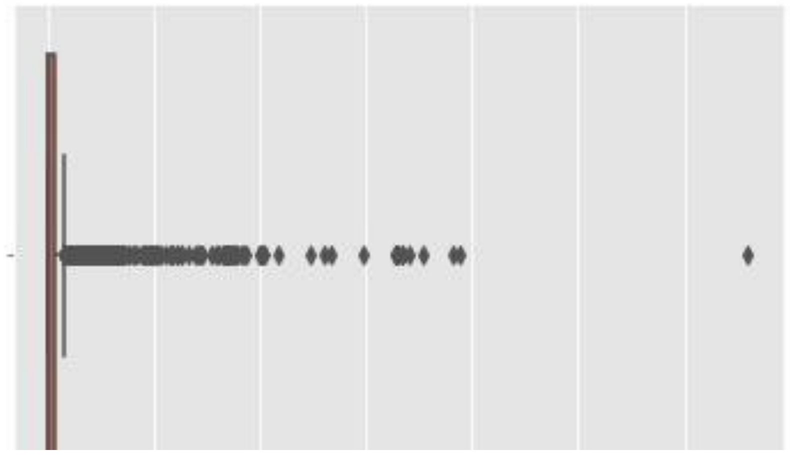
```
C1590550415    25
C985934102     22
C564160838     19
C451111351     17
C1023714065    15
               ..
M1113829504     1
M936219350      1
M178401052      1
```

utilising the value counts() function here to determine how many times the nameDest column appears.

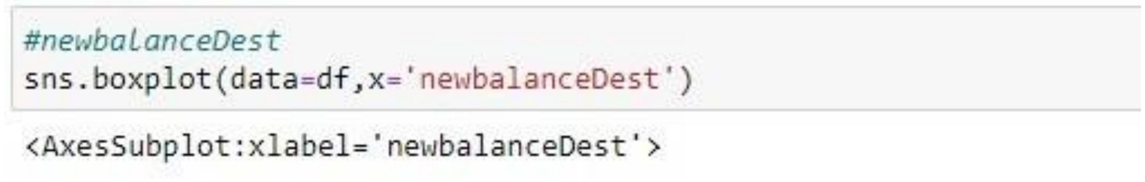


```
: #oldbalanceDest
  sns.boxplot(data=df,x='oldbalanceDest')
```

```
: <AxesSubplot:xlabel='oldbalanceDest'>
```

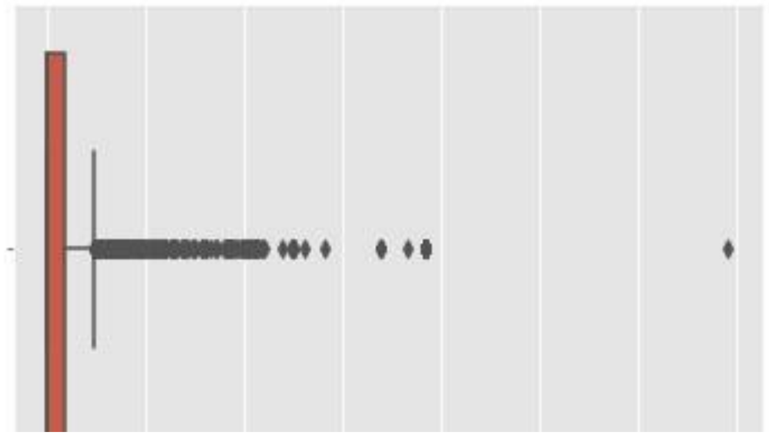


Here, the relationship between the oldbalanceDest attribute and the boxplot is visualised.

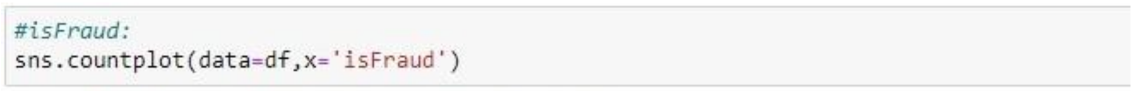


```
#newbalanceDest
sns.boxplot(data=df,x='newbalanceDest')
```

```
<AxesSubplot:xlabel='newbalanceDest'>
```

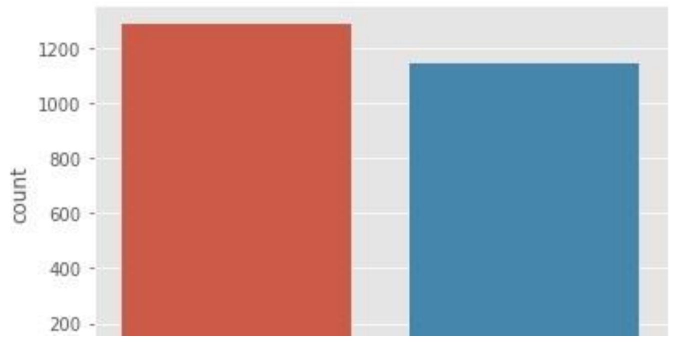


Here, the relationship between the newbalanceDest attribute and the boxplot is visualised.

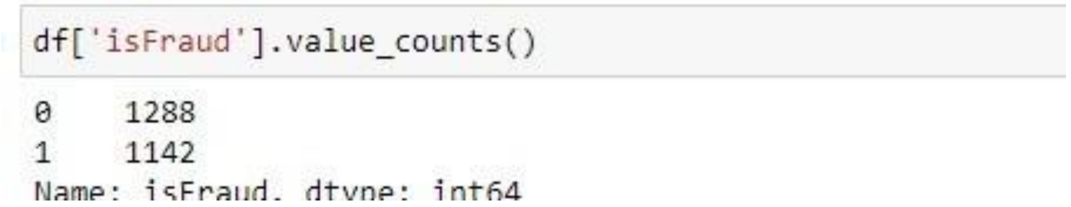


```
#isFraud:
sns.countplot(data=df,x='isFraud')
```

```
<AxesSubplot:xlabel='isFraud', ylabel='count'>
```



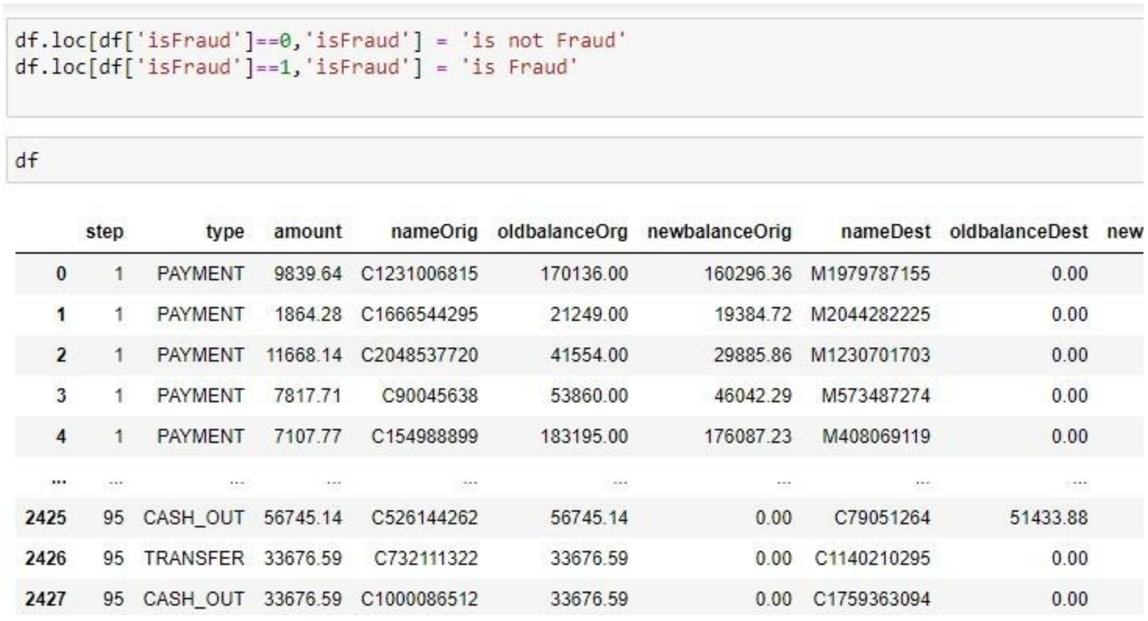
using the countplot approach here to count the number of instances in the dataset's target isFraud column.



```
df['isFraud'].value_counts()
```

```
0    1288
1    1142
Name: isFraud, dtype: int64
```

Here, we're using the value counts method to figure out how many classes there are in the dataset's target isFraud column.



```
df.loc[df['isFraud']==0,'isFraud'] = 'is not Fraud'
df.loc[df['isFraud']==1,'isFraud'] = 'is Fraud'
```

```
df
```

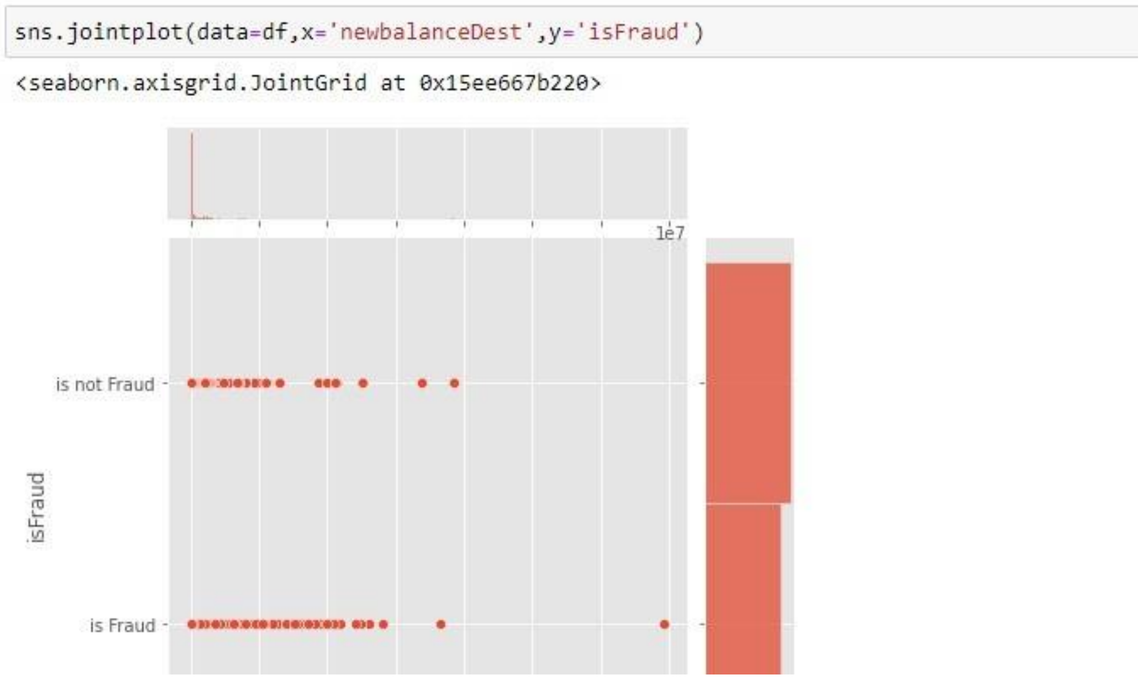
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	new
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	
...	...	...	...	...	...	...	...	...	
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	

converting 0-means: is not fraud and 1-means: is fraud using the loc technique here

Activity 4:Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between newbalanceDest and isFraud.

jointplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.



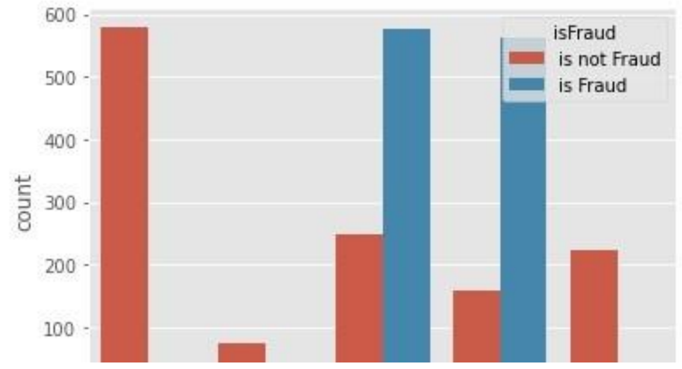
```
sns.jointplot(data=df,x='newbalanceDest',y='isFraud')
```

```
<seaborn.axisgrid.JointGrid at 0x15ee667b220>
```

Here we are visualising the relationship between type and isFraud.countplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.

```
sns.countplot(data=df,x='type',hue='isFraud')
```

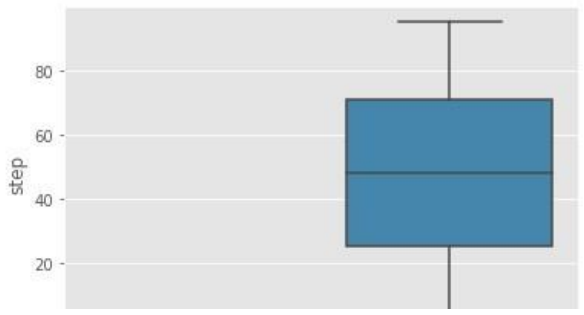
```
<AxesSubplot:xlabel='type', ylabel='count'>
```



Here we are visualising the relationship between isFraud and step.boxtplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.

```
sns.boxplot(data=df,x='isFraud',y='step')
```

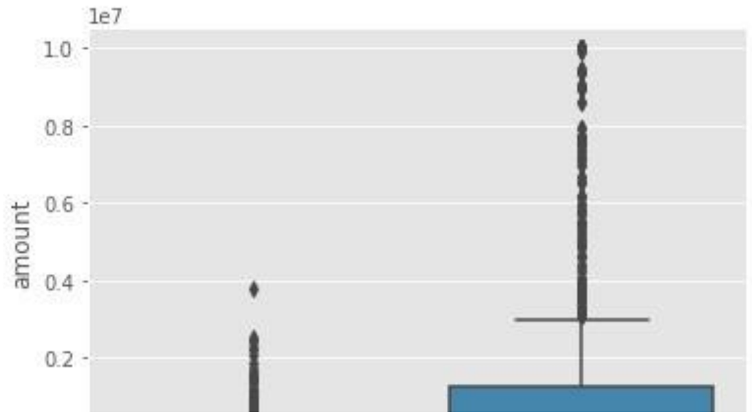
```
<AxesSubplot:xlabel='isFraud', ylabel='step'>
```



Here we are visualising the relationship between isFraud and amount.boxtplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.

```
sns.boxplot(data=df,x='isFraud',y='amount')
```

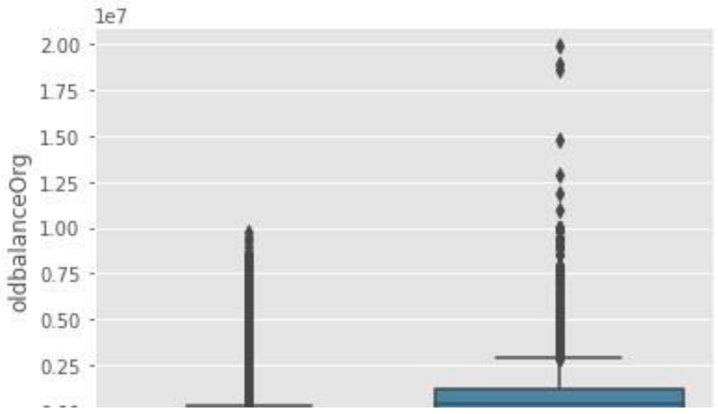
```
<AxesSubplot:xlabel='isFraud', ylabel='amount'>
```



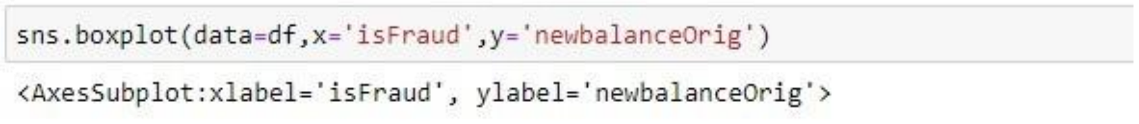
Here we are visualising the relationship between isFraud and oldbalanceOrg. boxtplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.

```
sns.boxplot(data=df,x='isFraud',y='oldbalanceOrg')
```

```
<AxesSubplot:xlabel='isFraud', ylabel='oldbalanceOrg'>
```

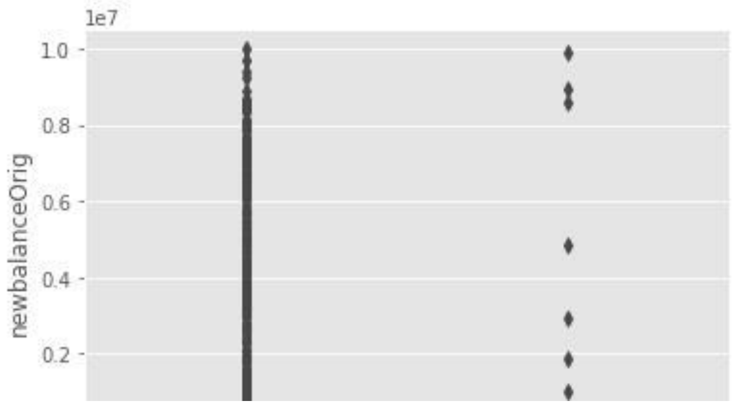


Here we are visualising the relationship between isFraud and newbalanceOrig. boxtplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.

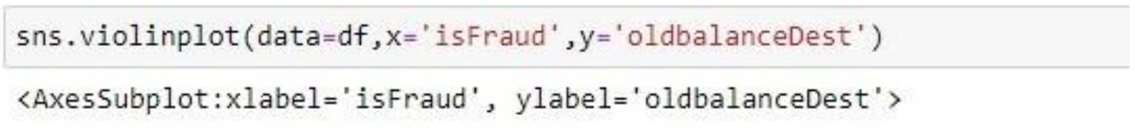


```
sns.boxplot(data=df,x='isFraud',y='newbalanceOrig')
```

```
<AxesSubplot:xlabel='isFraud', ylabel='newbalanceOrig'>
```

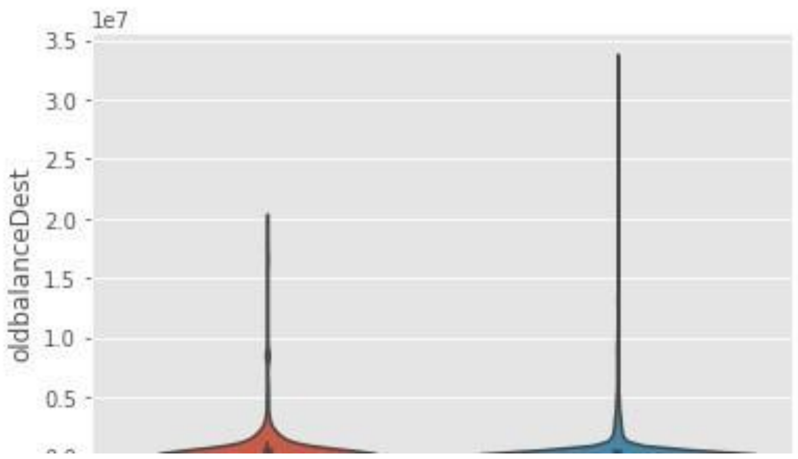


Here we are visualising the relationship between isFraud and oldbalanceDest. violinplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.



```
sns.violinplot(data=df,x='isFraud',y='oldbalanceDest')
```

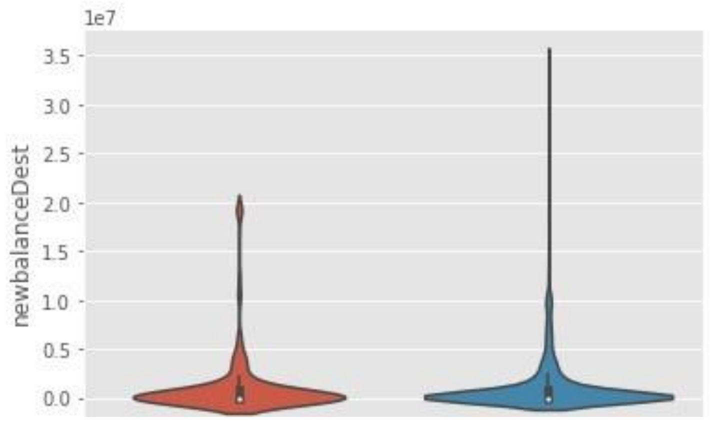
```
<AxesSubplot:xlabel='isFraud', ylabel='oldbalanceDest'>
```



Here we are visualising the relationship between isFraud and newbalanceDest. violinplot is used here. As a 1stparameter we are passing x value and as a 2ndparameter we are passing hue value.

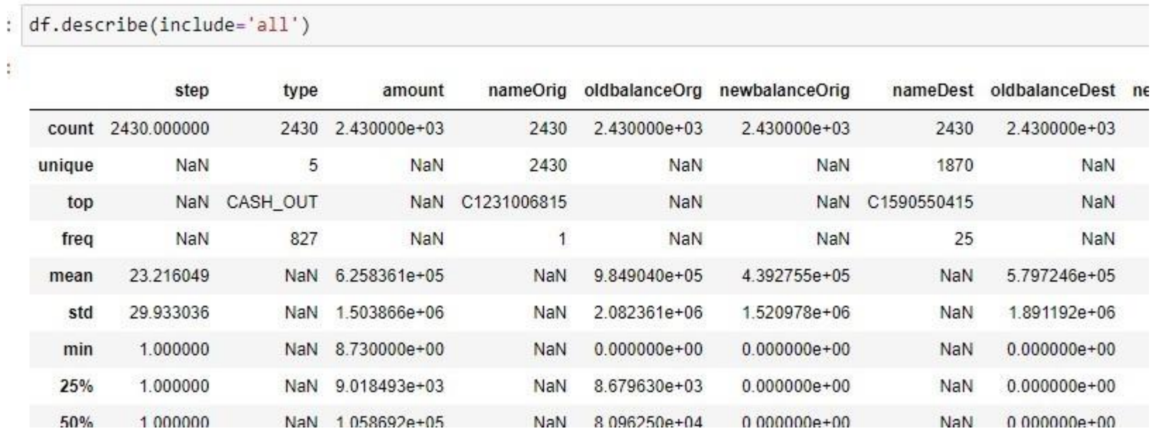
```
sns.violinplot(data=df,x='isFraud',y='newbalanceDest')
```

```
<AxesSubplot:xlabel='isFraud', ylabel='newbalanceDest'>
```



Activity 5: Descriptive analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
df.describe(include='all')
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	ne
count	2430.000000	2430	2.430000e+03	2430	2.430000e+03	2.430000e+03	2430	2.430000e+03	
unique	NaN	5	NaN	2430	NaN	NaN	1870	NaN	
top	NaN	CASH_OUT	NaN	C1231006815	NaN	NaN	C1590550415	NaN	
freq	NaN	827	NaN	1	NaN	NaN	25	NaN	
mean	23.216049	NaN	6.258361e+05	NaN	9.849040e+05	4.392755e+05	NaN	5.797246e+05	
std	29.933036	NaN	1.503866e+06	NaN	2.082361e+06	1.520978e+06	NaN	1.891192e+06	
min	1.000000	NaN	8.730000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	
25%	1.000000	NaN	9.018493e+03	NaN	8.679630e+03	0.000000e+00	NaN	0.000000e+00	
50%	1.000000	NaN	1.058692e+05	NaN	8.096250e+04	0.000000e+00	NaN	0.000000e+00	

Milestone 3: Data Pre-processing

As we have understood how the data is, let's pre-process the collected data.

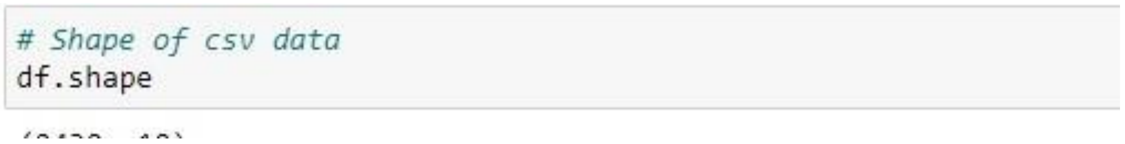
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

Handling missing values

HandlingObject data label encoding

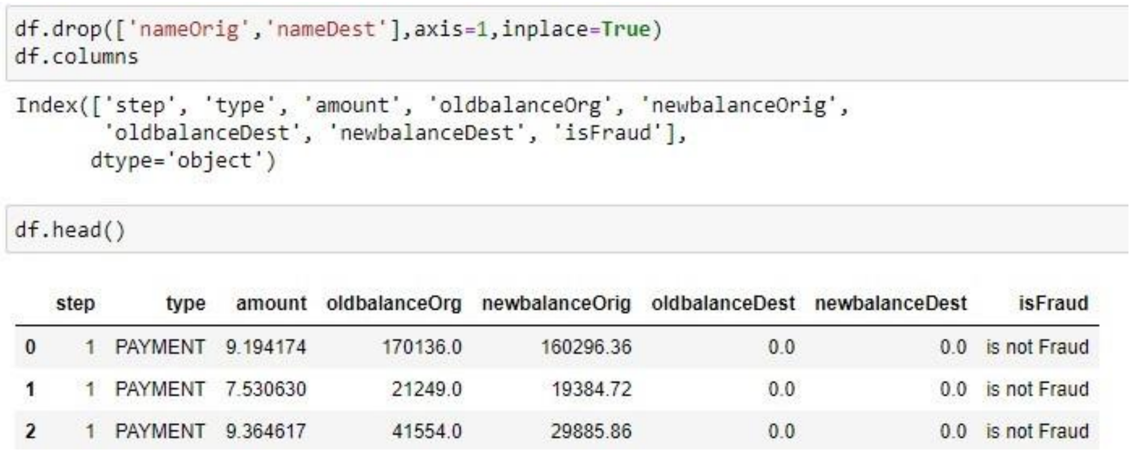
Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.



```
# Shape of csv data
df.shape
```

Here, I'm using the shape approach to figure out how big my dataset is



```
df.drop(['nameOrig','nameDest'],axis=1,inplace=True)
df.columns
```

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

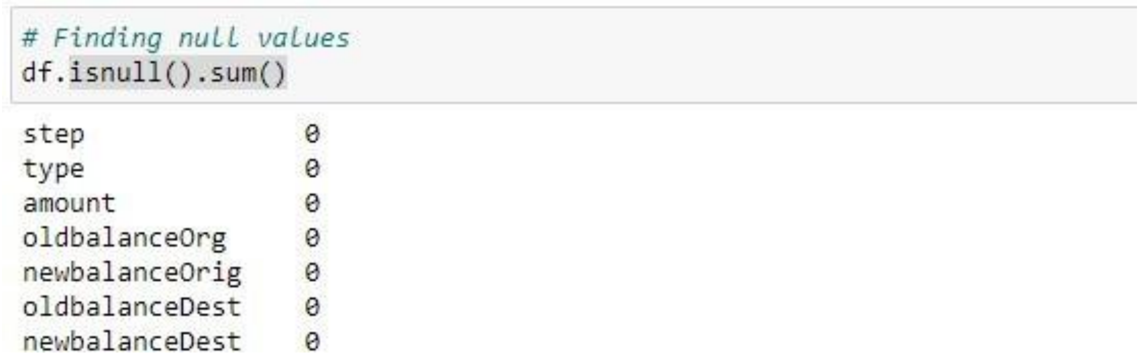
```
df.head()
```

```
   step     type    amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  newbalanceDest      isFraud
0     1  PAYMENT  9.194174       170136.0       160296.36             0.0             0.0  is not Fraud
1     1  PAYMENT  7.530630        21249.0        19384.72             0.0             0.0  is not Fraud
2     1  PAYMENT  9.364617        41554.0        29885.86             0.0             0.0  is not Fraud
```

here, the dataset's superfluous columns (nameOrig,nameDest) are being removed using the drop method.

Activity 1: Checking For Null Vaules

Isnull is used (). sum() to check your database for null values. Using the df.info() function, the data type can be determined.

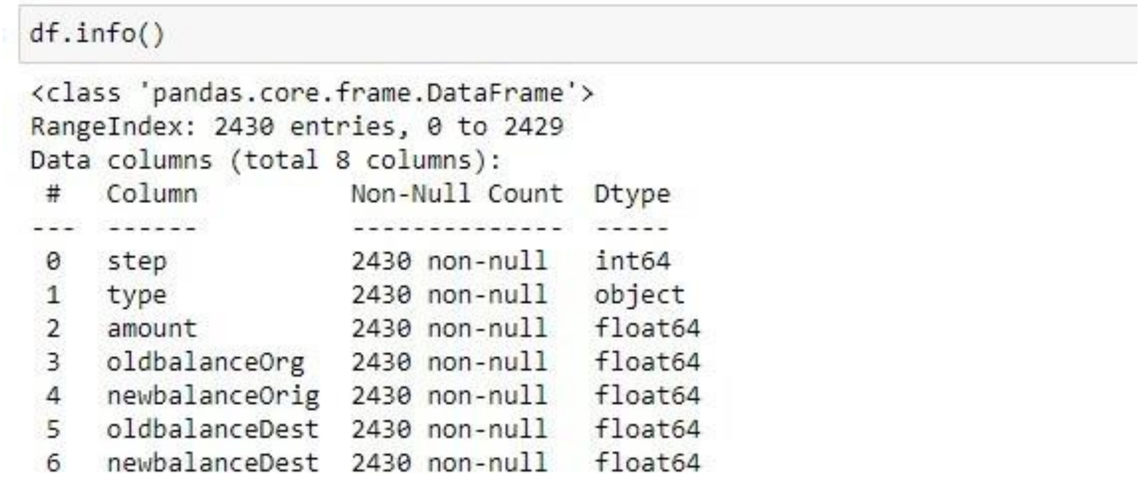


```
# Finding null values
df.isnull().sum()
```

```
step              0
type              0
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
```

For checking the null values, data.isnull() function is used. To sum those null values we use the .sum()

function to it. From the above image we found that there are no null values present in our dataset.So we can skip handling of missing values step.



```
df.info()
```

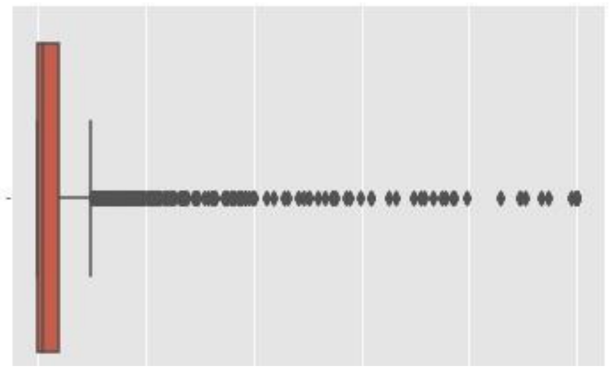
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2430 entries, 0 to 2429
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   step            2430 non-null   int64  
 1   type            2430 non-null   object 
 2   amount          2430 non-null   float64
 3   oldbalanceOrg   2430 non-null   float64
 4   newbalanceOrig  2430 non-null   float64
 5   oldbalanceDest  2430 non-null   float64
 6   newbalanceDest  2430 non-null   float64
```

determining the types of each attribute in the dataset using the info() function

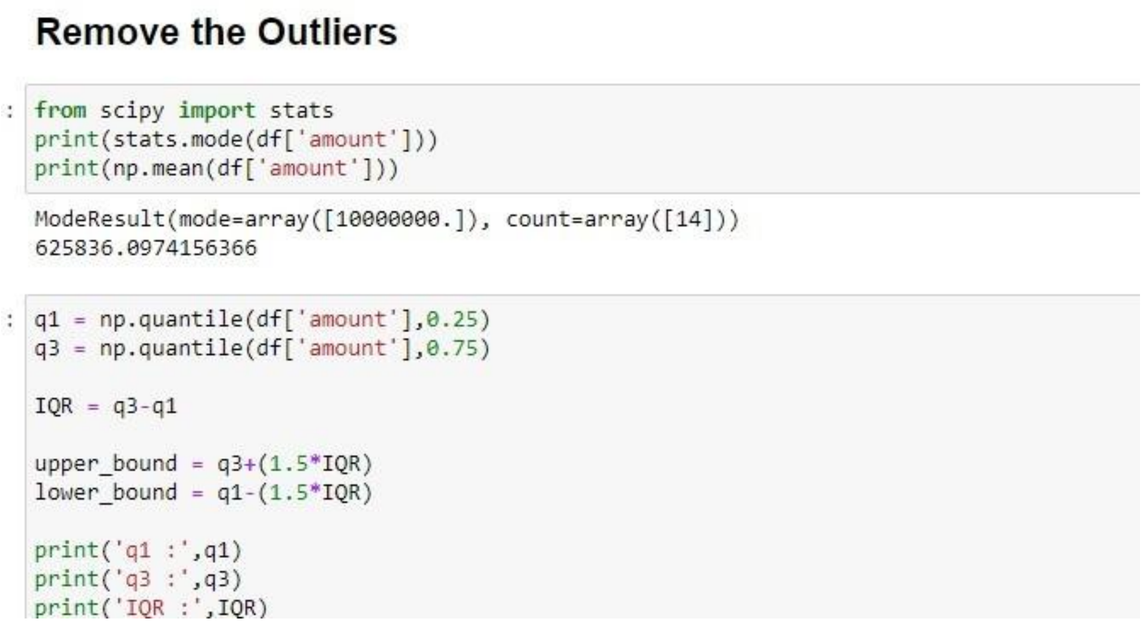
Activity 2: Handling Outliers

```
sns.boxplot(df['amount'])
```

```
<AxesSubplot:xlabel='amount'>
```



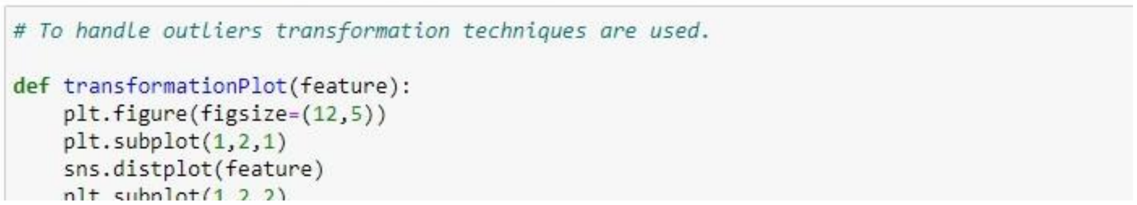
Here, a boxplot is used to identify outliers in the dataset's amount attribute.



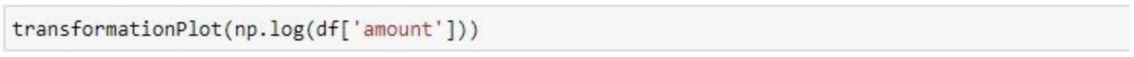
Remove the Outliers

```
: from scipy import stats
  print(stats.mode(df['amount']))
  print(np.mean(df['amount']))

  ModeResult(mode=array([10000000.]), count=array([14]))
  625836.0974156366
```

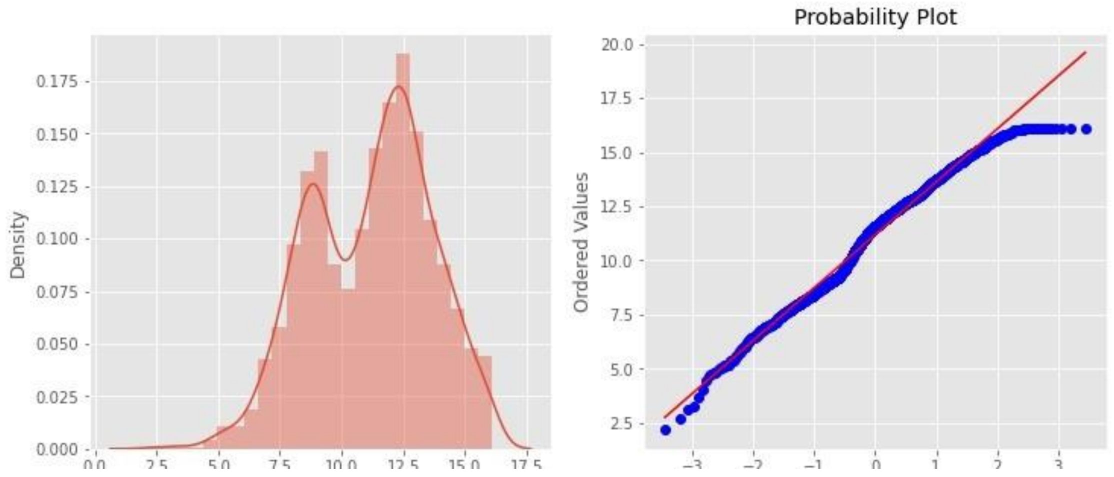
```
: q1 = np.quantile(df['amount'],0.25)
  q3 = np.quantile(df['amount'],0.75)

  IQR = q3-q1

  upper_bound = q3+(1.5*IQR)
  lower_bound = q1-(1.5*IQR)

  print('q1 :',q1)
  print('q3 :',q3)
  print('IQR :',IQR)
```



```
# To handle outliers transformation techniques are used.

def transformationPlot(feature):
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    sns.distplot(feature)
    plt.subplot(1,2,2)
```

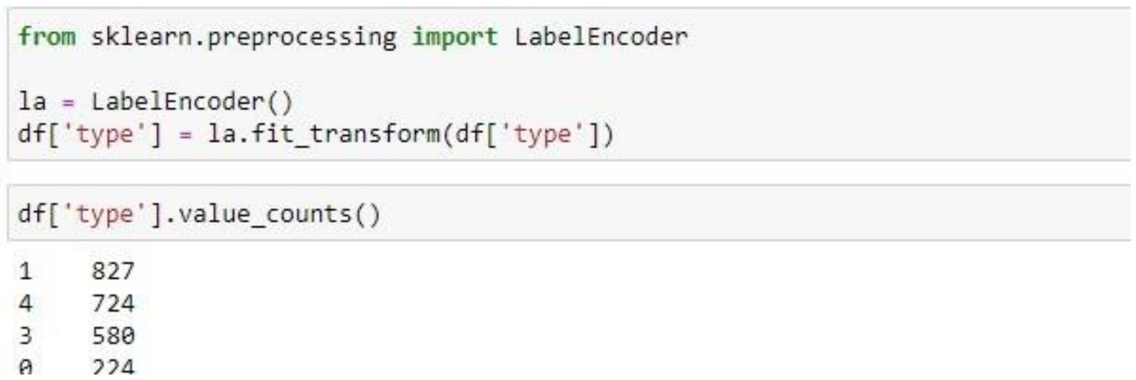


```
transformationPlot(np.log(df['amount']))
```



Here, transformationPlot is used to plot the dataset's outliers for the amount property.

Activity 3: Object Data Labelencoding

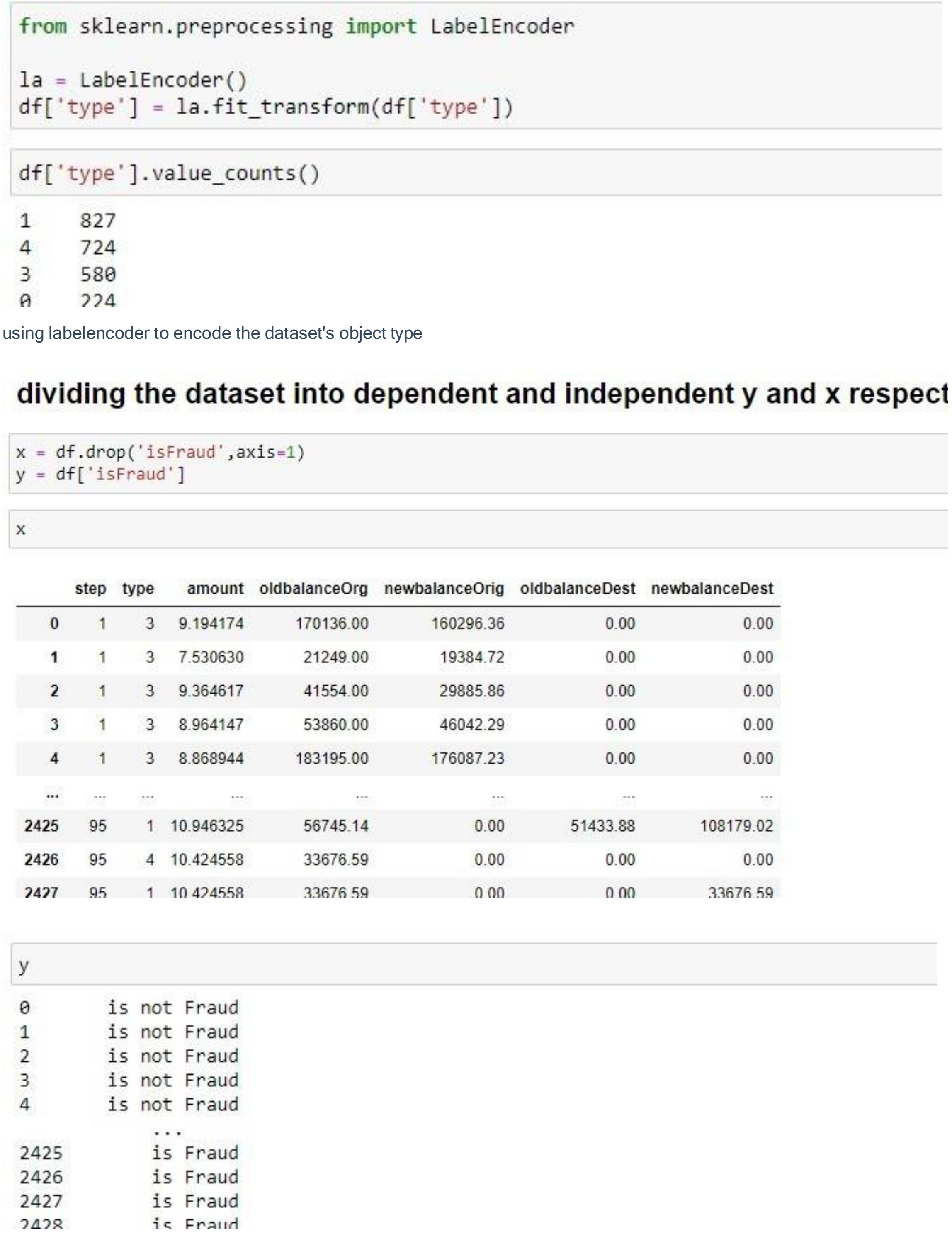


```
from sklearn.preprocessing import LabelEncoder

la = LabelEncoder()
df['type'] = la.fit_transform(df['type'])
```

```
df['type'].value_counts()
```

```
1    827
4    724
3    580
0    224
```

using labelencoder to encode the dataset's object type



dividing the dataset into dependent and independent y and x respect

```
x = df.drop('isFraud',axis=1)
y = df['isFraud']
```

```
x
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	1	3	9.194174	170136.00	160296.36	0.00	0.00
1	1	3	7.530630	21249.00	19384.72	0.00	0.00
2	1	3	9.364617	41554.00	29885.86	0.00	0.00
3	1	3	8.964147	53860.00	46042.29	0.00	0.00
4	1	3	8.868944	183195.00	176087.23	0.00	0.00
...	...	...	...	...	...	...	...
2425	95	1	10.946325	56745.14	0.00	51433.88	108179.02
2426	95	4	10.424558	33676.59	0.00	0.00	0.00
2427	95	1	10.424558	33676.59	0.00	0.00	33676.59

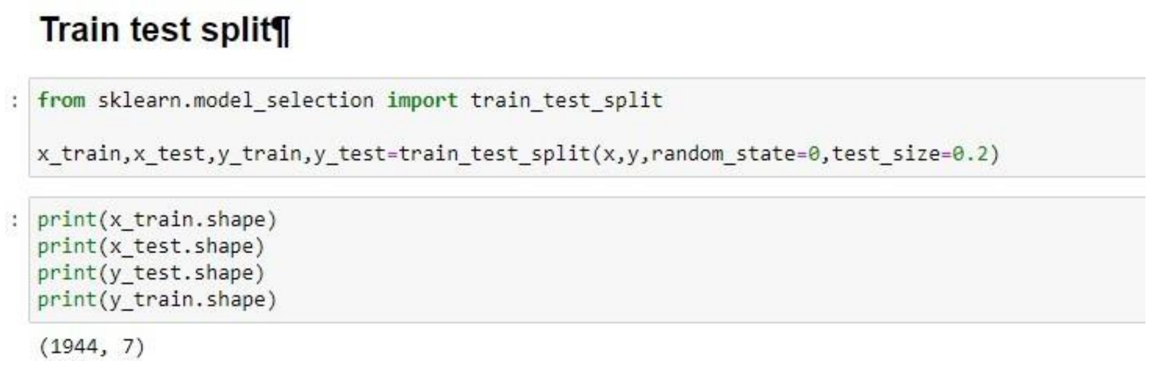
```
y
```

```
0        is not Fraud
1        is not Fraud
2        is not Fraud
3        is not Fraud
4        is not Fraud
             ...
2425         is Fraud
2426         is Fraud
2427         is Fraud
2428         is Fraud
```

Activity 4: Splitting Data Into Train And Test

Now let's split the Dataset into train and test setsChanges: first split the dataset into x and y and then split the data set.

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And my target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



Train test split¶

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_test.shape)
print(y_train.shape)
```

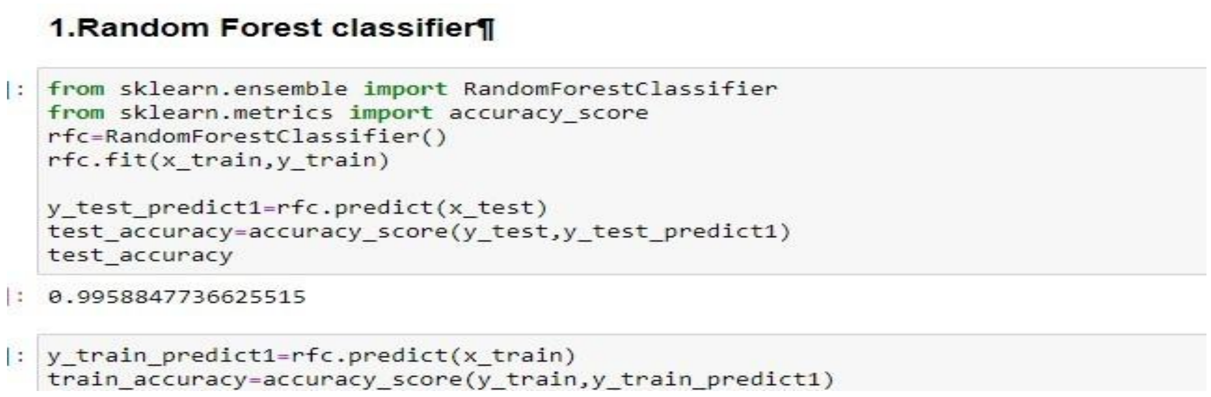
```
(1944, 7)
```

Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1: Random Forest Classifier

A function named RandomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

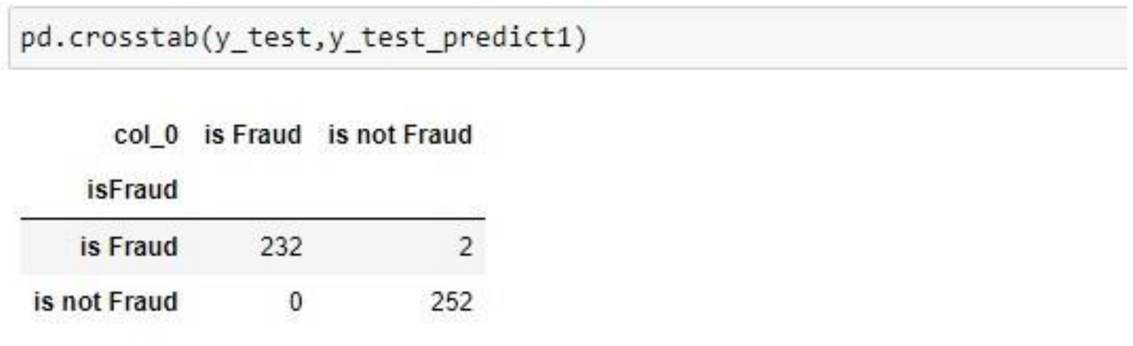


1.Random Forest classifier¶

```
[ ]: from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     rfc=RandomForestClassifier()
     rfc.fit(x_train,y_train)

     y_test_predict1=rfc.predict(x_test)
     test_accuracy=accuracy_score(y_test,y_test_predict1)
     test_accuracy
```

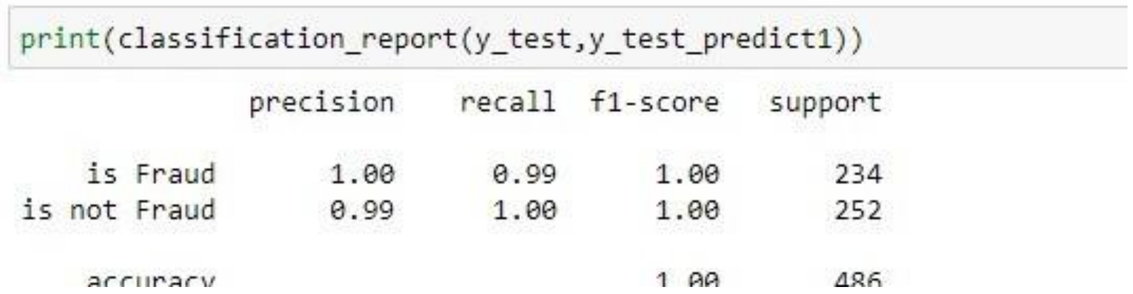
```
[ ]: 0.9958847736625515
```

```
[ ]: y_train_predict1=rfc.predict(x_train)
     train_accuracy=accuracy_score(y_train,y_train_predict1)
```



```
pd.crosstab(y_test,y_test_predict1)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	232	2
is not Fraud	0	252

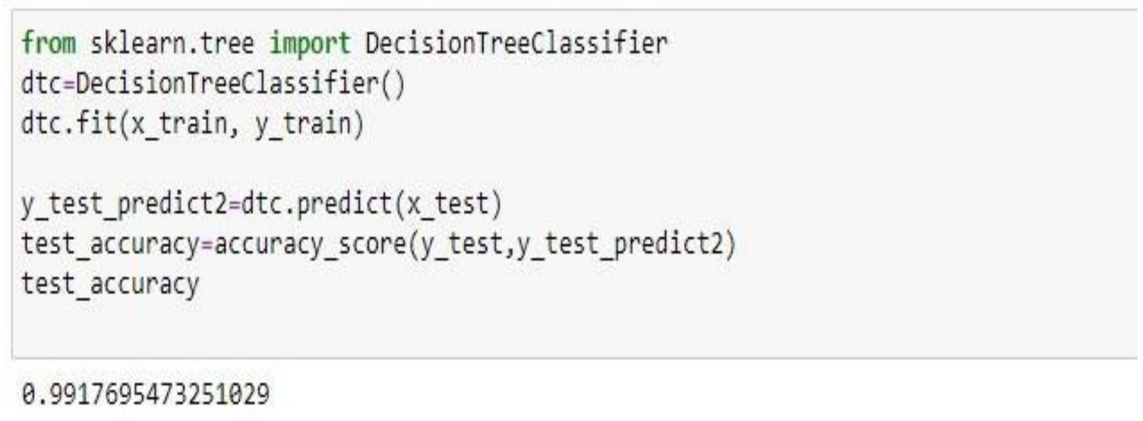


```
print(classification_report(y_test,y_test_predict1))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.99      1.00       234
is not Fraud       0.99      1.00      1.00       252

    accuracy                           1.00       486
```

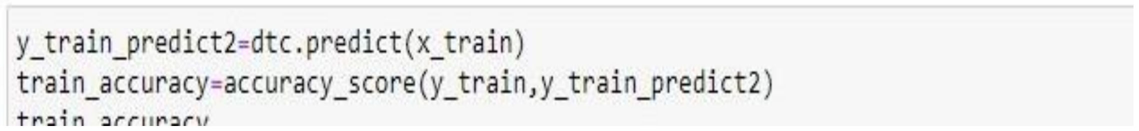
Activity 2: Decision Tree Classifier

A function namedDecisiontreeis created and train and test data are passed as the parameters. Inside the function, the DecisiontreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

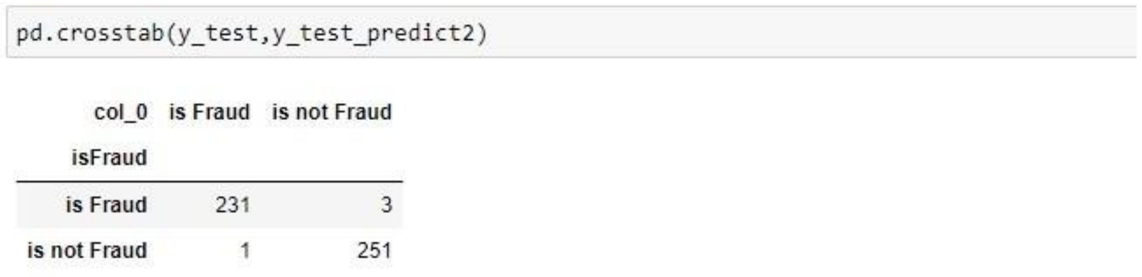


```python
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train, y_train)

y_test_predict2=dtc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict2)
test_accuracy
```

```
0.9917695473251029
```

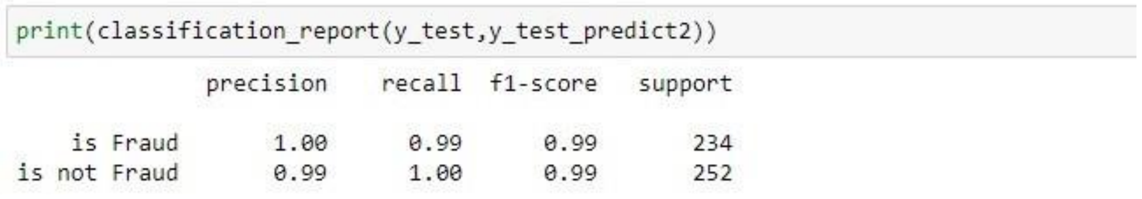


```python
y_train_predict2=dtc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict2)
train_accuracy
```



```python
pd.crosstab(y_test,y_test_predict2)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	231	3
is not Fraud	1	251

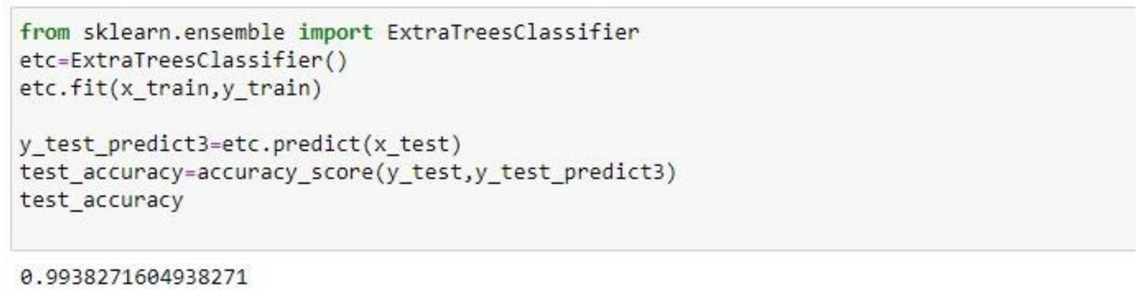


```python
print(classification_report(y_test,y_test_predict2))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.99      0.99       234
is not Fraud       0.99      1.00      0.99       252
```

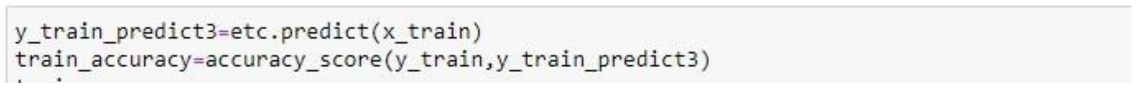
Activity 3: ExtraTrees Classifier

A function named ExtraTree is created and train and test data are passed as the parameters. Inside the function, ExtraTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

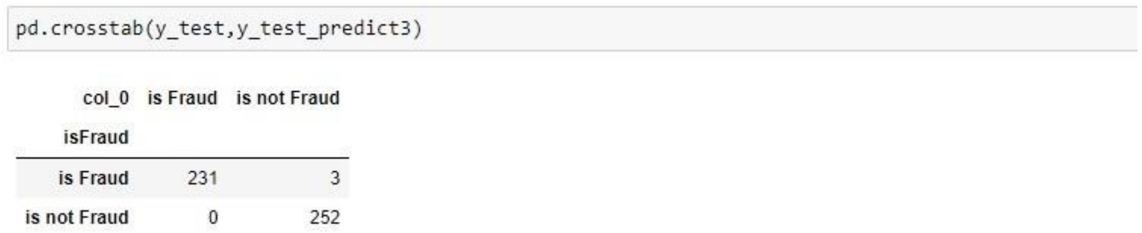


```
from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train,y_train)

y_test_predict3=etc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict3)
test_accuracy
```

```
0.9938271604938271
```

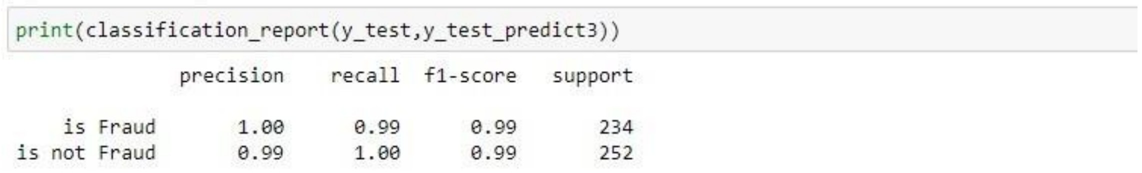


```
y_train_predict3=etc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict3)
```



```
pd.crosstab(y_test,y_test_predict3)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	231	3
is not Fraud	0	252

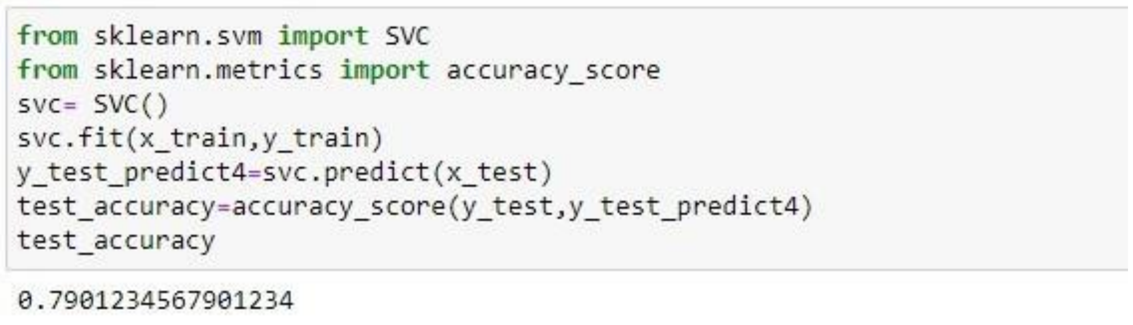


```
print(classification_report(y_test,y_test_predict3))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.99      0.99       234
is not Fraud       0.99      1.00      0.99       252
```

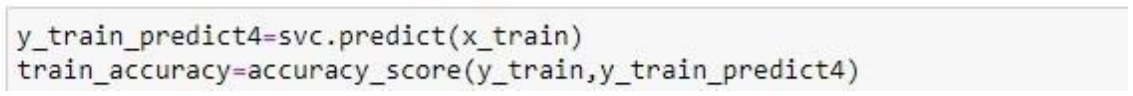
Activity 4: Support Vector Machine Classifier

A function named SupportVector is created and train and test data are passed as the parameters. Inside the function, the SupportVectorClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, confusion matrix and classification report is done



```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC()
svc.fit(x_train,y_train)
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict4)
test_accuracy
```

```
0.7901234567901234
```



```
y_train_predict4=svc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict4)
```

```
pd.crosstab(y_test,y_test_predict4)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	132	102
is not Fraud	0	252

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_test_predict4))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.56      0.72       234
is not Fraud       0.71      1.00      0.83       252
```

```
df.columns
```

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

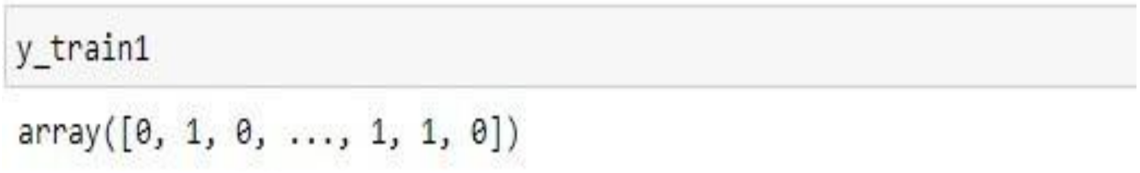
```
from sklearn.preprocessing import LabelEncoder

la = LabelEncoder()
y_train1 = la.fit_transform(y_train)
```

preprocessing class of sklearn. LabelEncoder[source] 0 to n classes-1 as the range for the target labels to be encoded. Instead of encoding the input X, the target values, i.e. y, should be encoded using this transformer.

```
y_test1=la.transform(y_test)
```

```
y_test1
```

```
array([0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
```

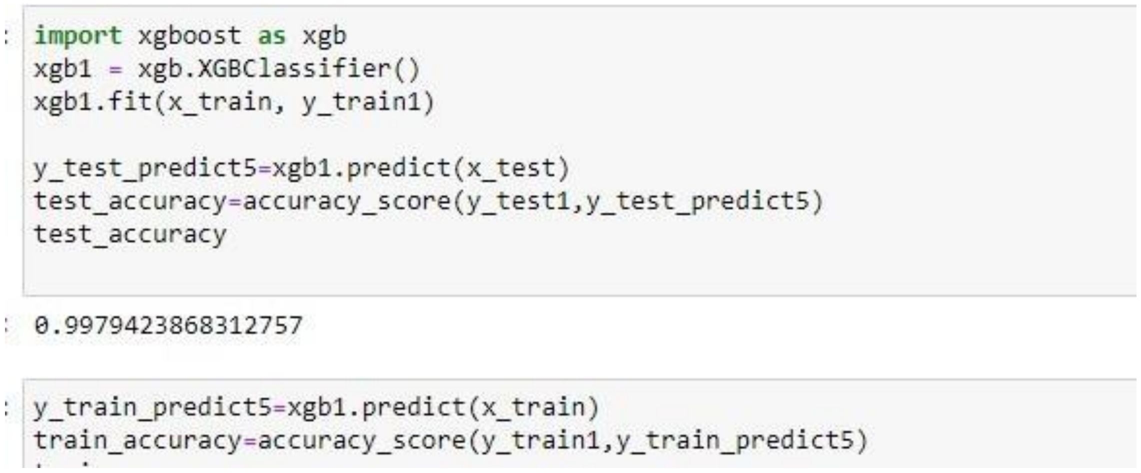


```
y_train1
```

```
array([0, 1, 0, ..., 1, 1, 0])
```

Activity 5: Xgboost Classifier

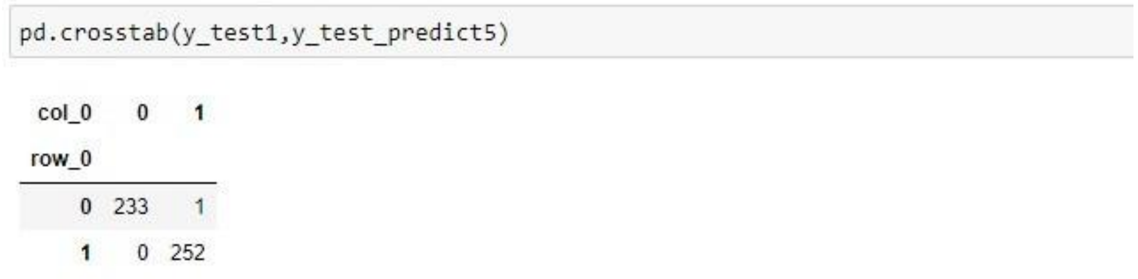
A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the xgboostClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, confusion matrix and classification report is done



```
import xgboost as xgb
xgb1 = xgb.XGBClassifier()
xgb1.fit(x_train, y_train1)

y_test_predict5=xgb1.predict(x_test)
test_accuracy=accuracy_score(y_test1,y_test_predict5)
test_accuracy
```

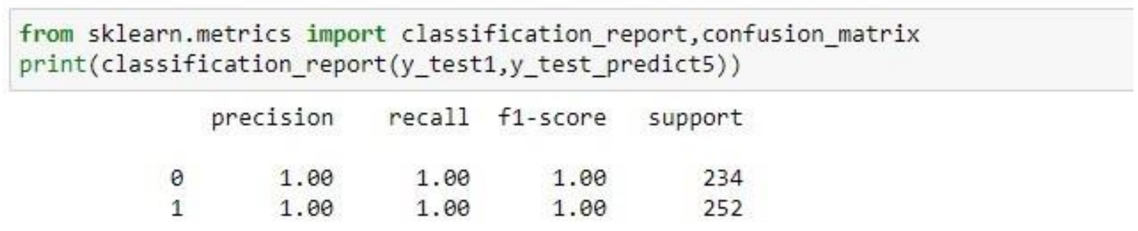
```
0.9979423868312757
```

```
y_train_predict5=xgb1.predict(x_train)
train_accuracy=accuracy_score(y_train1,y_train_predict5)
```



```
pd.crosstab(y_test1,y_test_predict5)
```

```
col_0    0    1
row_0
    0  233    1
    1    0  252
```



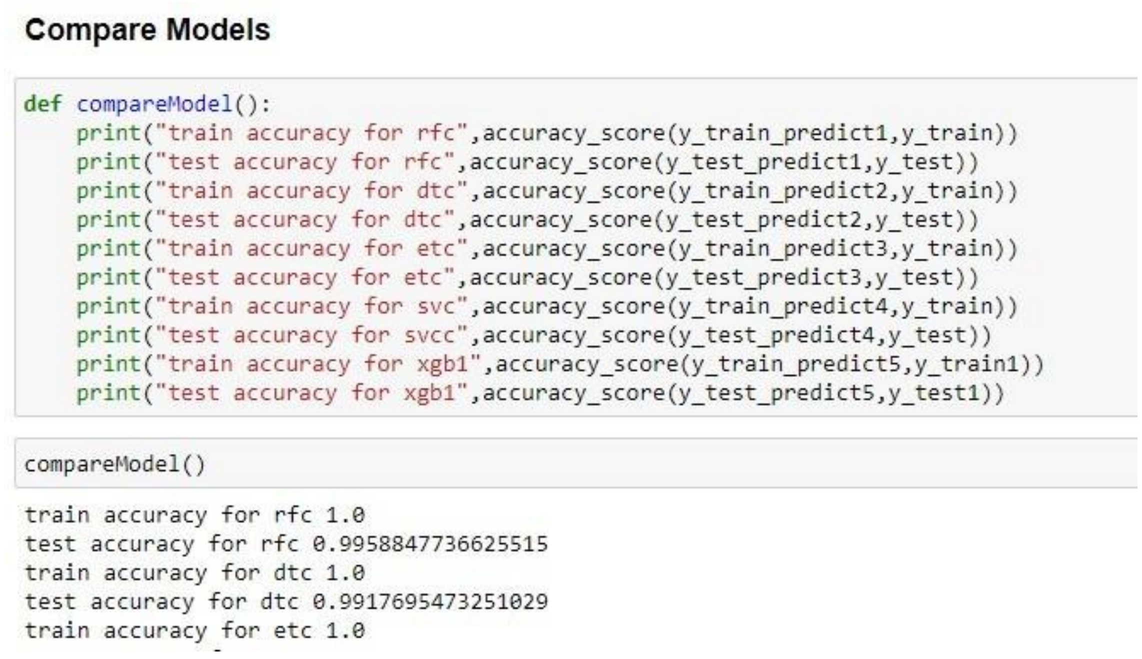
```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test1,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       234
           1       1.00      1.00      1.00       252
```

Activity 6:Compare the models

For comparing the above four models, the compareModel function is defined.

After calling the function, the results of models are displayed as output. From the five models, the svc is performing well. From the below image, We can see the accuracy of the model is 79% accuracy. .



Compare Models

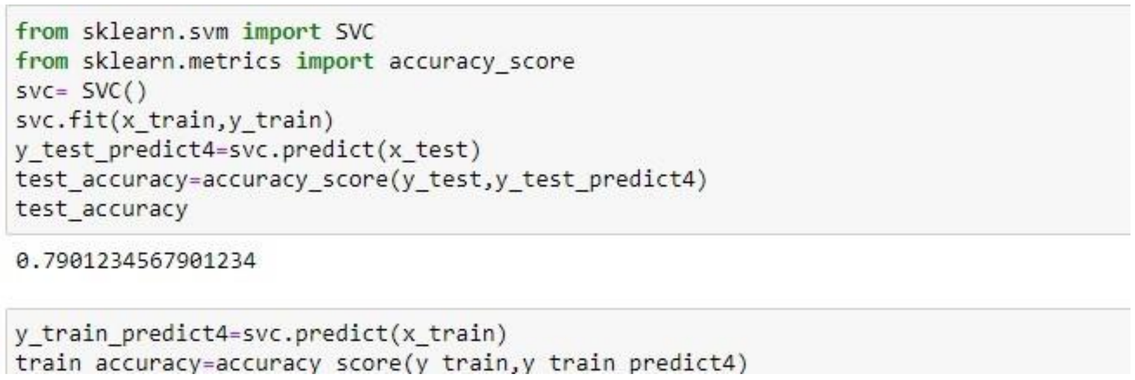
```
def compareModel():
    print("train accuracy for rfc",accuracy_score(y_train_predict1,y_train))
    print("test accuracy for rfc",accuracy_score(y_test_predict1,y_test))
    print("train accuracy for dtc",accuracy_score(y_train_predict2,y_train))
    print("test accuracy for dtc",accuracy_score(y_test_predict2,y_test))
    print("train accuracy for etc",accuracy_score(y_train_predict3,y_train))
    print("test accuracy for etc",accuracy_score(y_test_predict3,y_test))
    print("train accuracy for svc",accuracy_score(y_train_predict4,y_train))
    print("test accuracy for svcc",accuracy_score(y_test_predict4,y_test))
    print("train accuracy for xgb1",accuracy_score(y_train_predict5,y_train1))
    print("test accuracy for xgb1",accuracy_score(y_test_predict5,y_test1))
```

```
compareModel()
```

```
train accuracy for rfc 1.0
test accuracy for rfc 0.9958847736625515
train accuracy for dtc 1.0
test accuracy for dtc 0.9917695473251029
train accuracy for etc 1.0
```

Activity 7: Evaluating Performance of The Model And Saving The Model

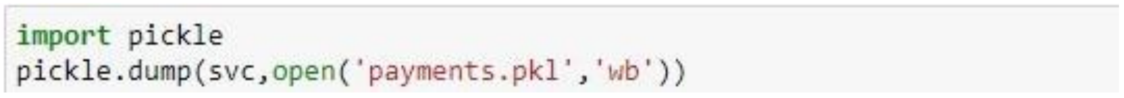
From sklearn, accuracy_score is used to evaluate the score of the model. On the parameters, we have given svc (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model is svc by pickle.dump().



```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC()
svc.fit(x_train,y_train)
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict4)
test_accuracy
```

```
0.7901234567901234
```

```
y_train_predict4=svc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict4)
```



```
import pickle
pickle.dump(svc,open('payments.pkl','wb'))
```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

Building HTML Pages

Building server side script

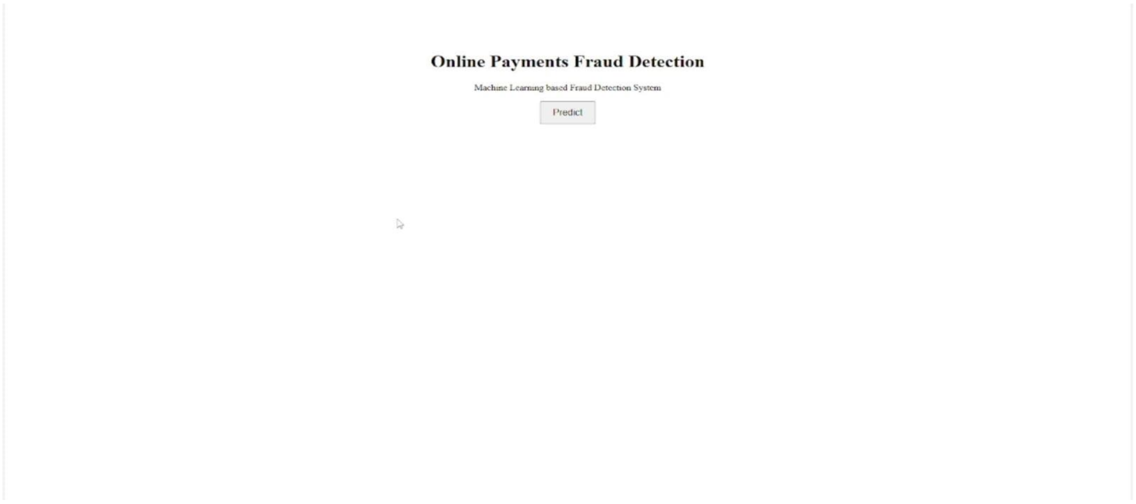
Activity1: Build Html Pages

For this project create three HTML files namely

- home.html
- predict.html
- submit.html

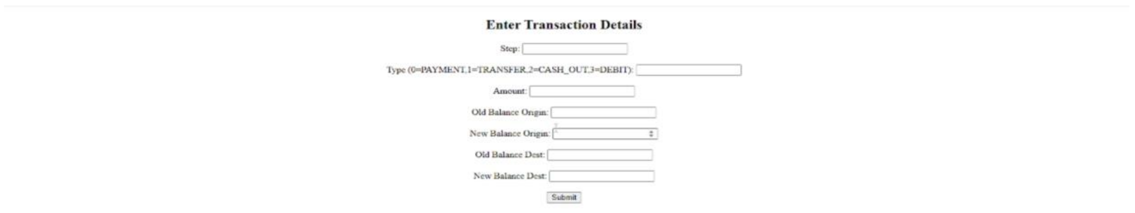
and save them in the templates folder.

Let’s see how our home.html page looks like:


Online Payments Fraud Detection
Machine Learning based Fraud Detection System
Predict

Now when you click on predict button from top right corner you will get redirected to predict.html

Let's look how our predict.html file looks like:



Enter Transaction Details

Step:

Type (0=PAYMENT,1=TRANSFER,2=CASH_OUT,3=DEBIT):

Amount:

Old Balance Origin:

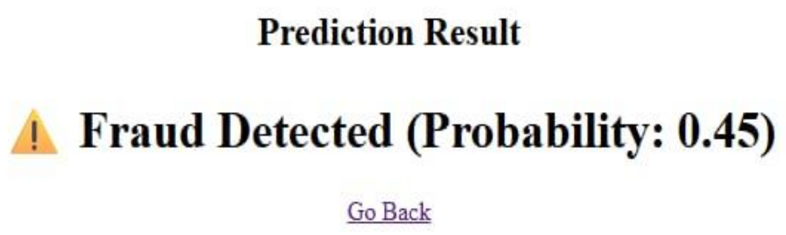
New Balance Origin:

Old Balance Dest:

New Balance Dest:

Now when you click on submit button from left bottom corner you will get redirected to submit.html

Let's look how our submit.html file looks like:

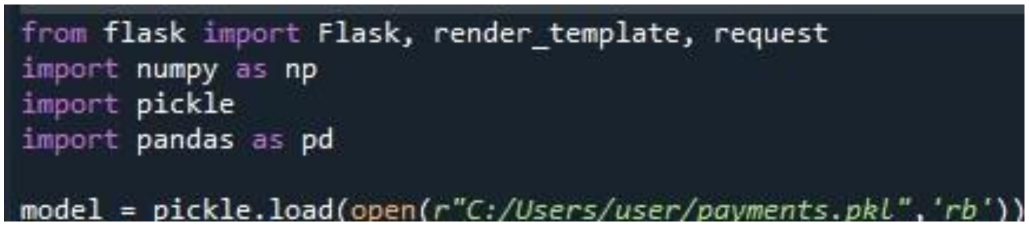


Prediction Result

⚠️ Fraud Detected (Probability: 0.45)

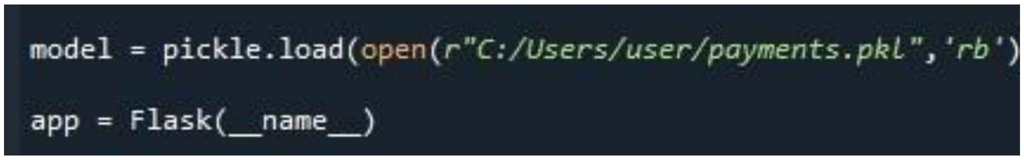
Activity 2 :Build Python Code

Import the libraries



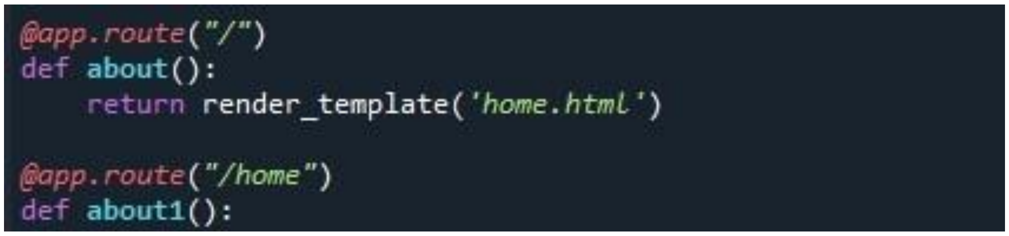
```
from flask import Flask, render_template, request
import numpy as np
import pickle
import pandas as pd

model = pickle.load(open(r"C:/Users/user/payments.pkl",'rb'))
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



```
model = pickle.load(open(r"C:/Users/user/payments.pkl",'rb')

app = Flask(__name__)
```

Render HTML page:

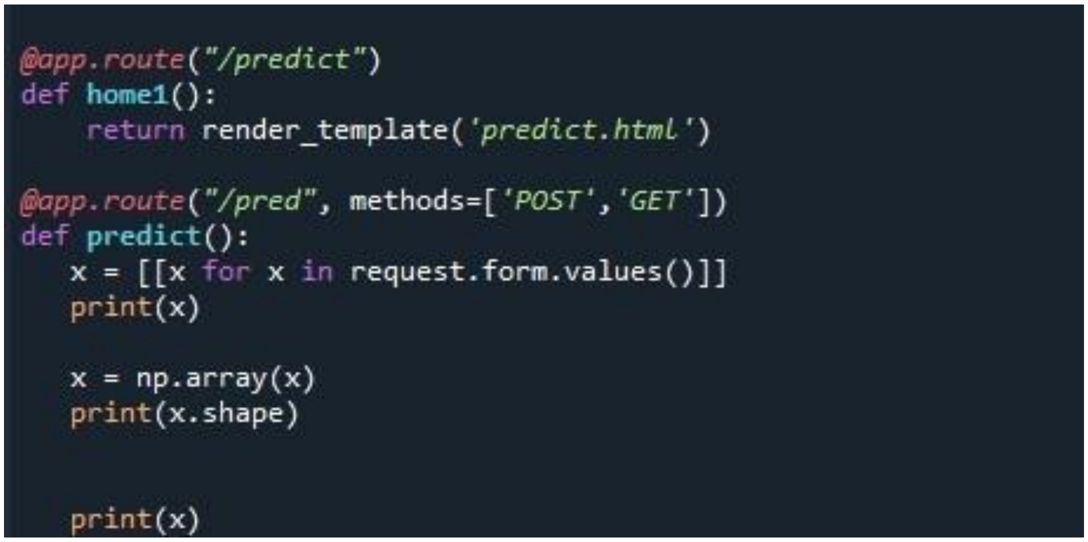


```
@app.route("/")
def about():
    return render_template('home.html')

@app.route("/home")
def about1():
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

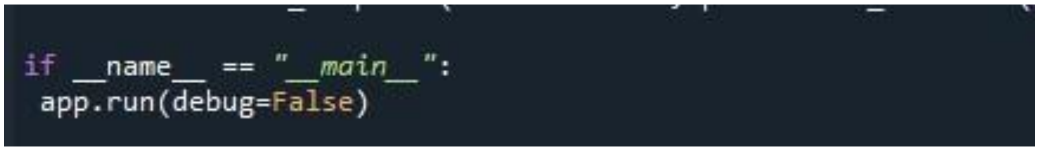
Retrieves the value from UI:



```
@app.route("/predict")
def home1():
    return render_template('predict.html')

@app.route("/pred", methods=['POST','GET'])
def predict():
   x = [[x for x in request.form.values()]]
   print(x)

   x = np.array(x)
   print(x.shape)


   print(x)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

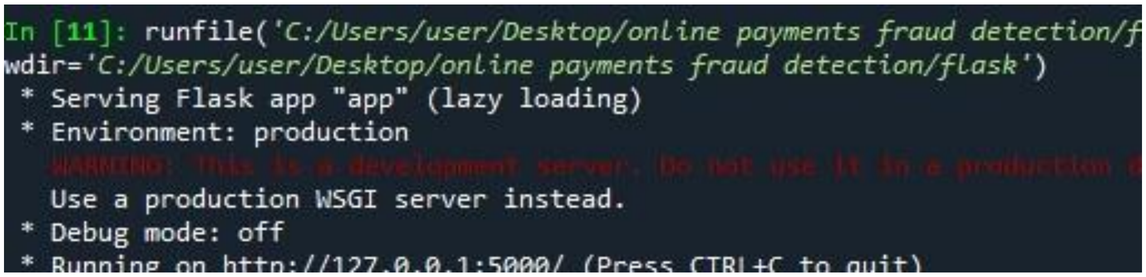
Main Function:



```
if __name__ == "__main__":
 app.run(debug=False)
```

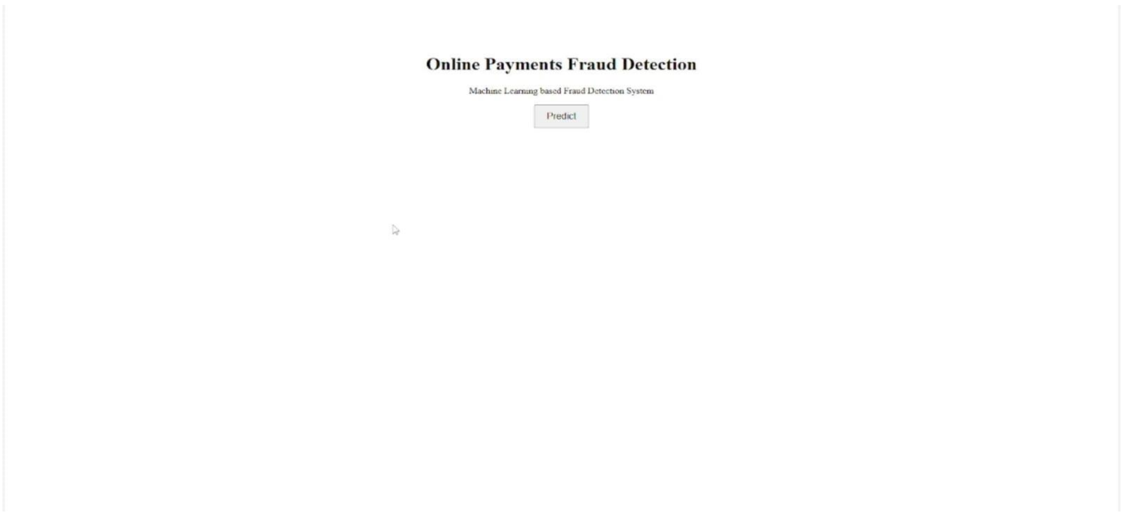
Activity 3: Run the application

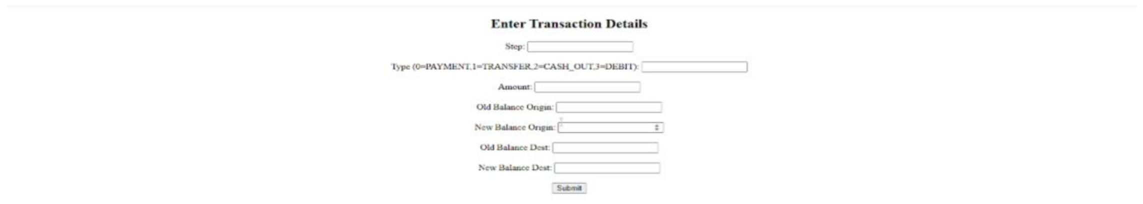
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
In [11]: runfile('C:/Users/user/Desktop/online payments fraud detection/f
wdir='C:/Users/user/Desktop/online payments fraud detection/flask')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production d
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Output screensorts:


Online Payments Fraud Detection
Machine Learning based Fraud Detection System
Predict


Enter Transaction Details

Step:

Type (0=PAYMENT,1=TRANSFER,2=CASH_OUT,3=DEBIT):

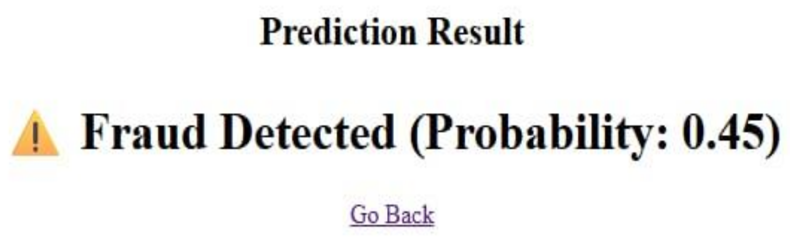
Amount:

Old Balance Origin:

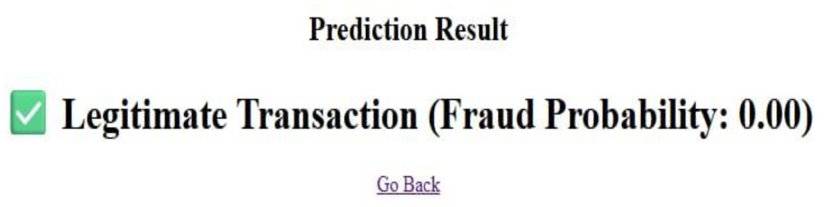
New Balance Origin:

Old Balance Dest:

New Balance Dest:


Prediction Result

⚠️ Fraud Detected (Probability: 0.45)


Prediction Result

✅ Legitimate Transaction (Fraud Probability: 0.00)