Online Payments Fraud Detection using ML

Team ID : Team-591594

Team Size : 4

Aniket Saxena

Tanvita Mandava

Kizhakke Vadasseril Varsha

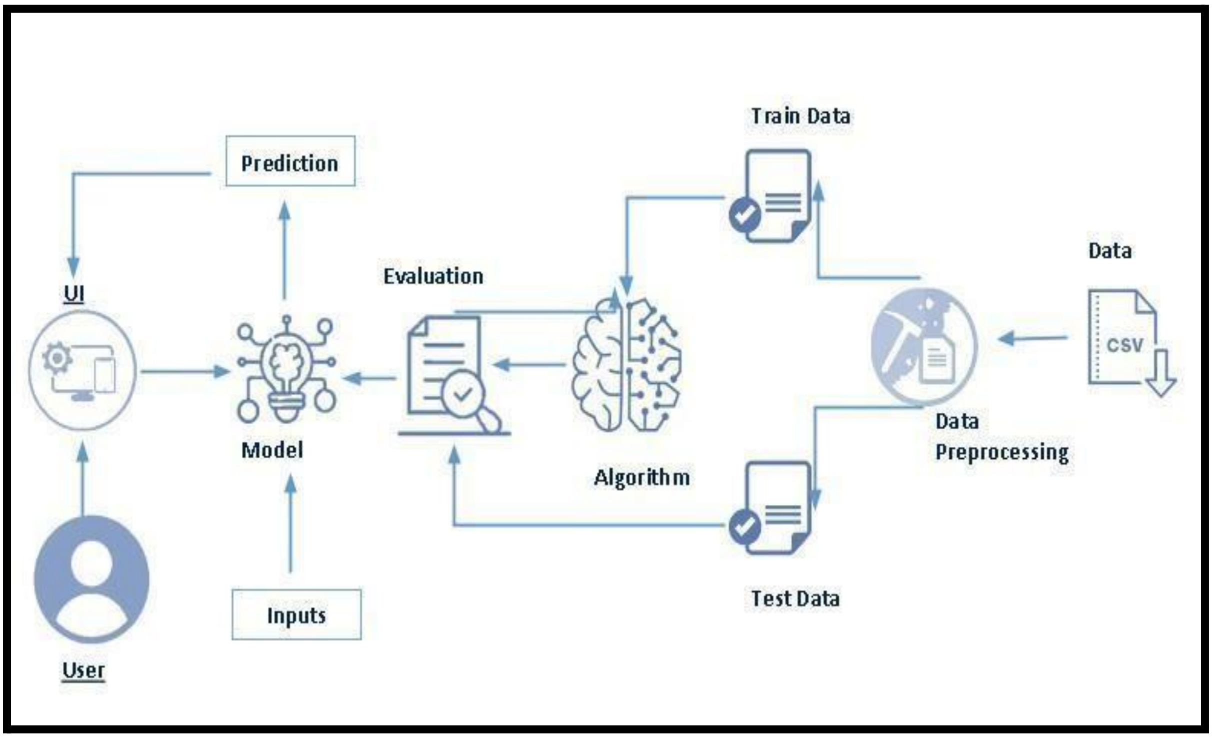
Dinkar Yadav

Project Description:

The growth in internet and e-commerce appears to involve the use of online credit/debit card transactions. The increase in the use of credit / debit cards is causing an increase in fraud. The frauds can be detected through various approaches, yet they lag in their accuracy and its own specific drawbacks. If there are any changes in the conduct of the transaction, the frauds are predicted and taken for further process. Due to large amount of data credit / debit card fraud detection problem is rectified by the proposed method

We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier, xgboost Classifier**.**We will train and test the data with these algorithms. From this the best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

Technical Architecture:



Pre requisites:

For this project, one must required following software's, concepts and packages

- **Google Collab and pycharm:** o Refer the link below to download
- Link : https://youtu.be/1ra4zH2G4o0
- Google Collab: https://colab.research.google.com/
- **Python packages:**
  - Open anaconda 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 pickle-mixin ”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://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning

  Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

  - **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some visualisation concepts.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.

- Once model analyses the input the prediction is 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
- Visualising and analysing data

Importing the

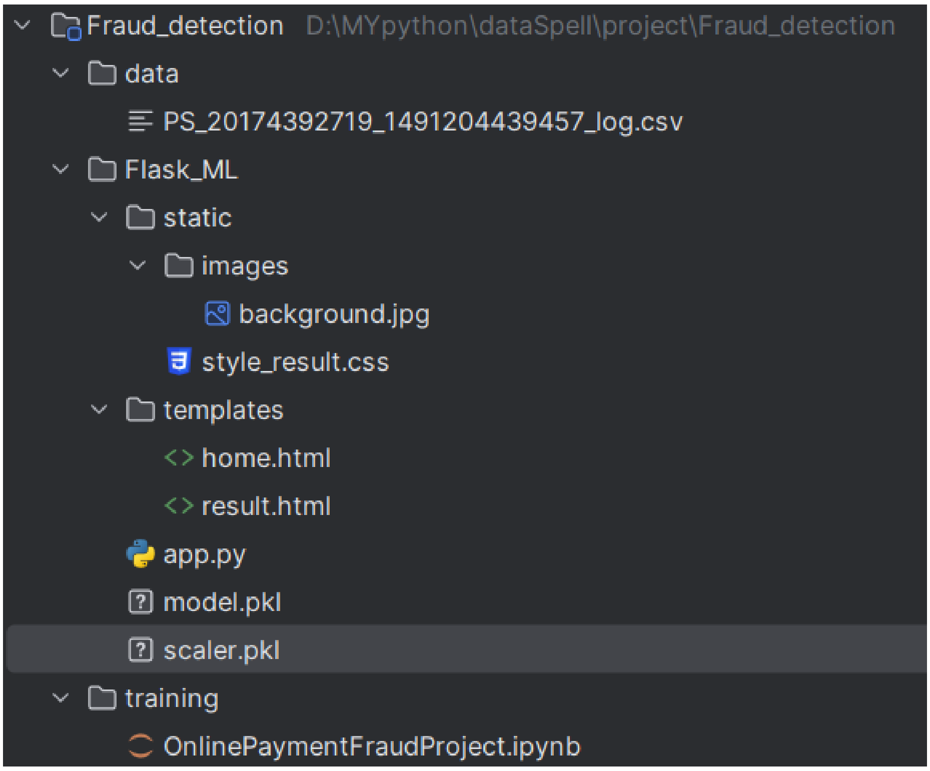
libraries

  - Read the Dataset
    - Univariate analysis
    - Bivariate analysis
    - Descriptive analysis
- Data pre-processing
- Checking for null values
- Handling outlier
- Handling categorical(object) data
- Splitting data into train and test
- Model building
- Import the model building libraries
- Initialising the model
- Training and testing the model
- Evaluating performance of model
- Save the model
- Application Building

- Create an HTML file
- Build python code

Project Structure:

Create the Project folder which contains files as shown below



- 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 the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Collect the dataset or create the dataset or 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.

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

Milestone 2: Visualising and analysing data

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

Note: There are a 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.

Import the Libraries

```
[10] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

We will be using more libraries but we can import them as we require them

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.

```
[11] df=pd.read_csv("/content/PS_20174392719_1491204439457_log.csv")
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1	0

6362620 rows × 11 columns

In this step we will print number of the columns

```
df.columns

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

We will be removing the non-required columns 'isFlaggedFraud' by using the drop function

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

The next step we will print the dataset ‘df’ to see the changes

```python
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0
...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1

6362620 rows × 10 columns

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
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

df.head command loads the first five records of the dataset named as df

```
df.head()
```

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

tail() function loads the last five records of the dataset

```
df.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C776919290	0.00	339682.13	1
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C1881841831	0.00	0.00	1
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C1365125890	68488.84	6379898.11	1
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C2080388513	0.00	0.00	1
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C873221189	6510099.11	7360101.63	1

We will be using corr() function to view the correlation between Integer columns of the dataset

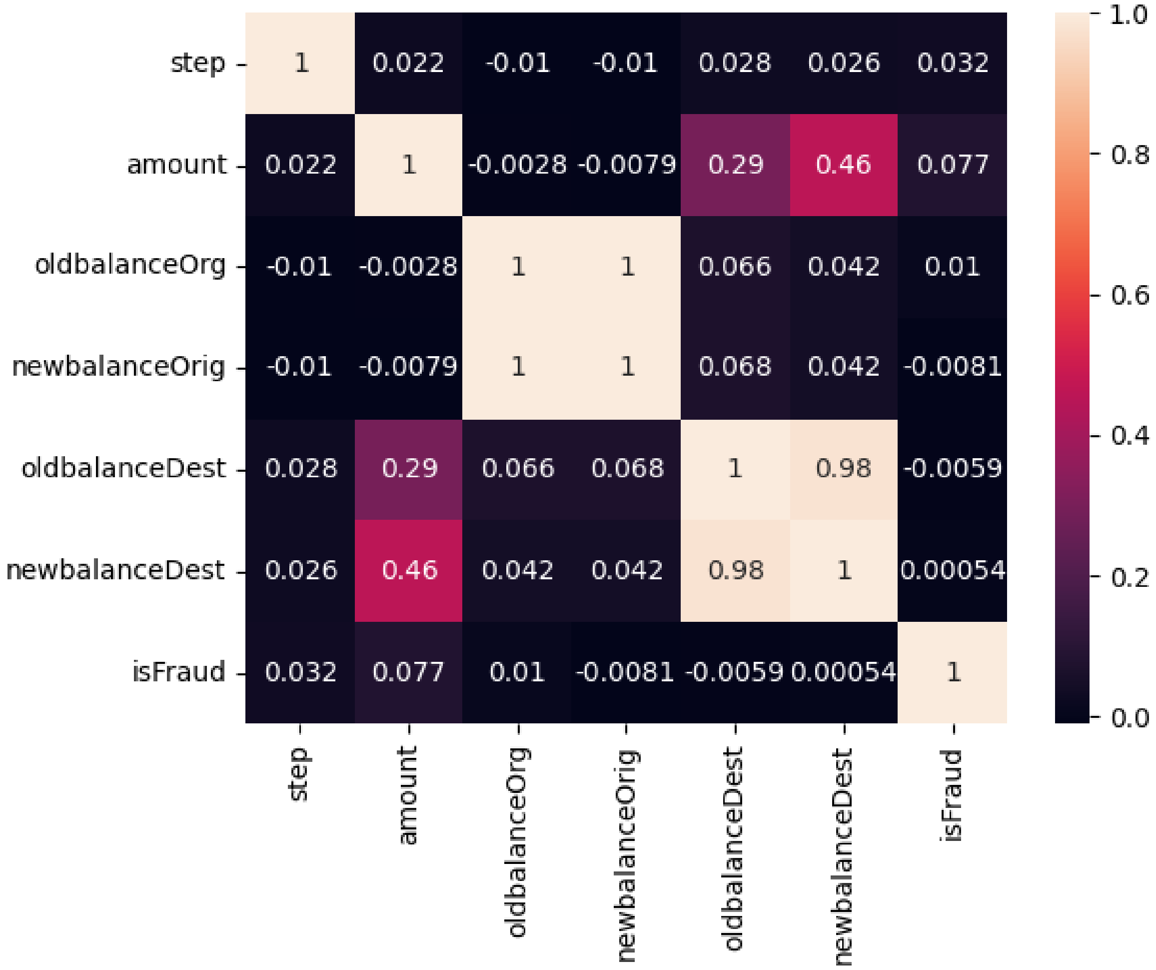
```python
df.corr()
```

```
<ipython-input-18-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated.
  df.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.031578
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.076688
oldbalanceOrg	-0.010058	-0.002762	1.000000	0.998803	0.066243	0.042029	0.010154
newbalanceOrig	-0.010299	-0.007861	0.998803	1.000000	0.067812	0.041837	-0.008148
oldbalanceDest	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.005885
newbalanceDest	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	0.000535
isFraud	0.031578	0.076688	0.010154	-0.008148	-0.005885	0.000535	1.000000

We can visualize the correlation as heatmap for our better understanding if we want the number also we can give annot=True . This will give the correlation number on top of the colour.

```python
import seaborn as sns
sns.heatmap(df.corr(),annot=True)
```



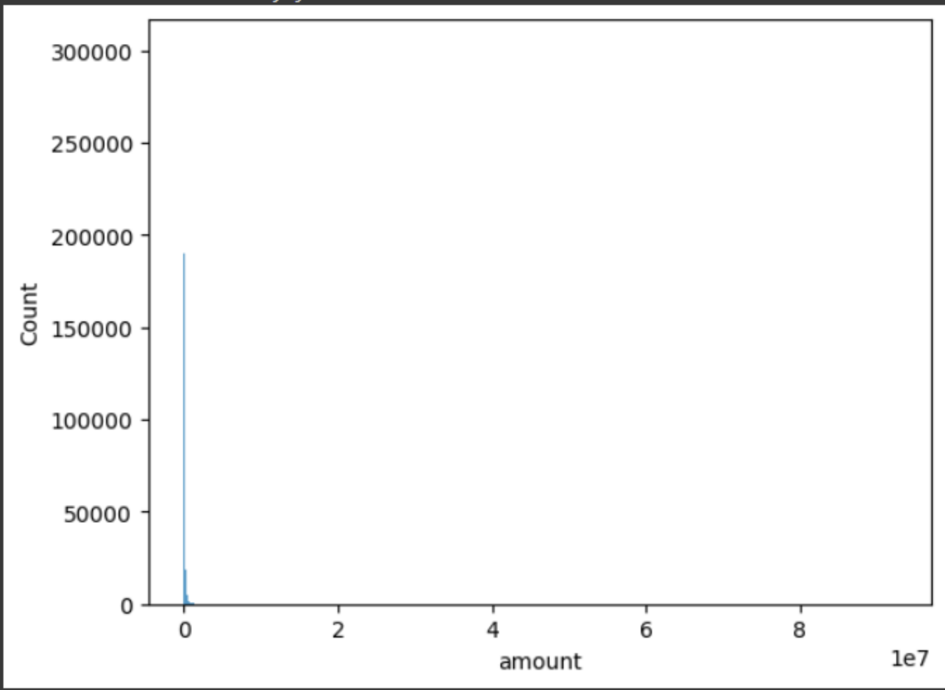
Here, a heatmap is used to understand the relationship between the input attributes and the anticipated goal value.

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 .of feature amount

```
[21] sns.histplot(data=df,x='amount')
```

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

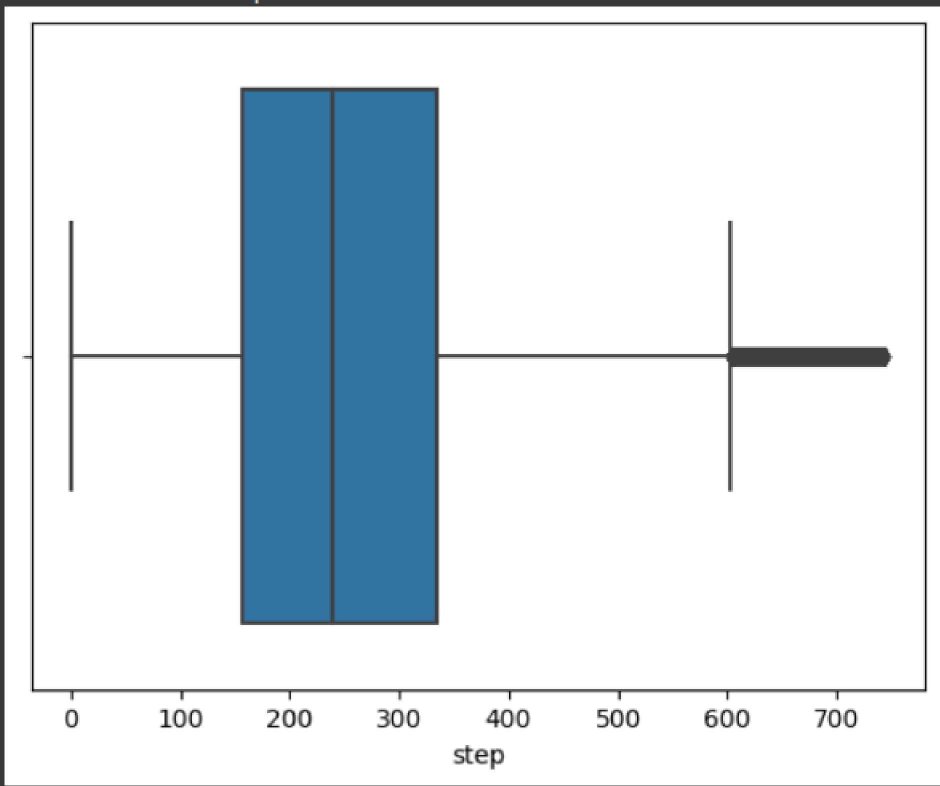


The distribution of one or more variables is represented by a histogram, a traditional visualisation tool, by counting the number of observations that fall within.

Next we will view the relationship between amount attribute as a boxplot

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

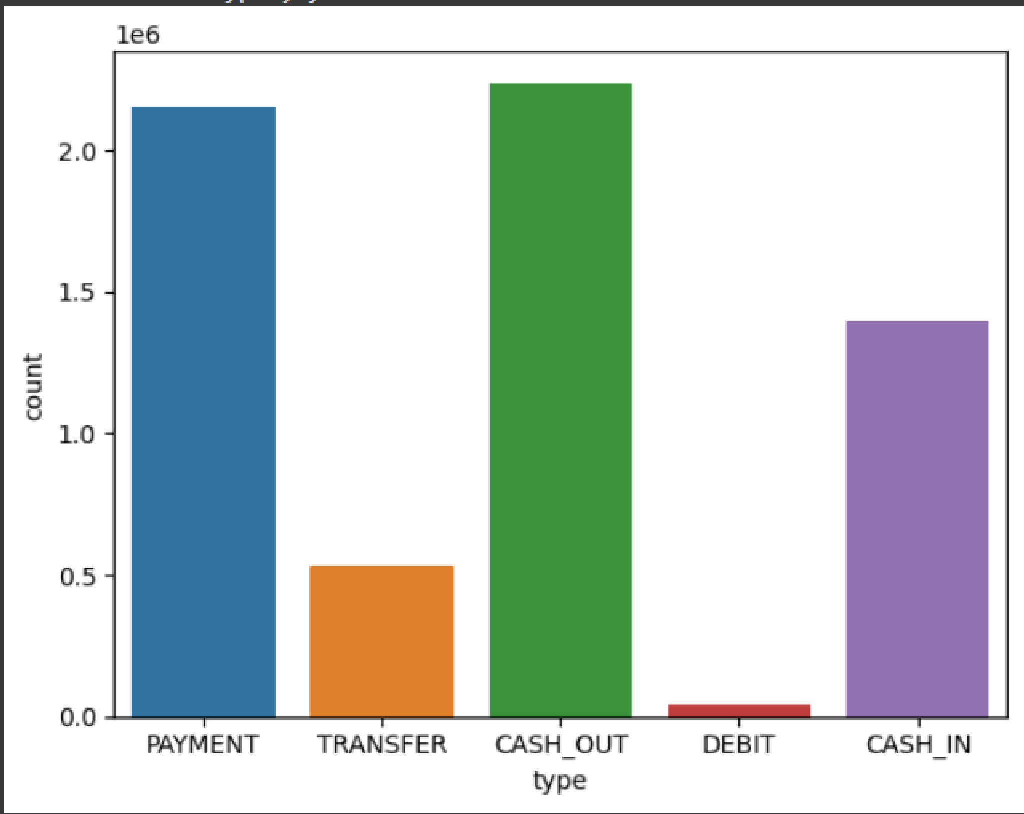
<Axes: xlabel='step'>



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

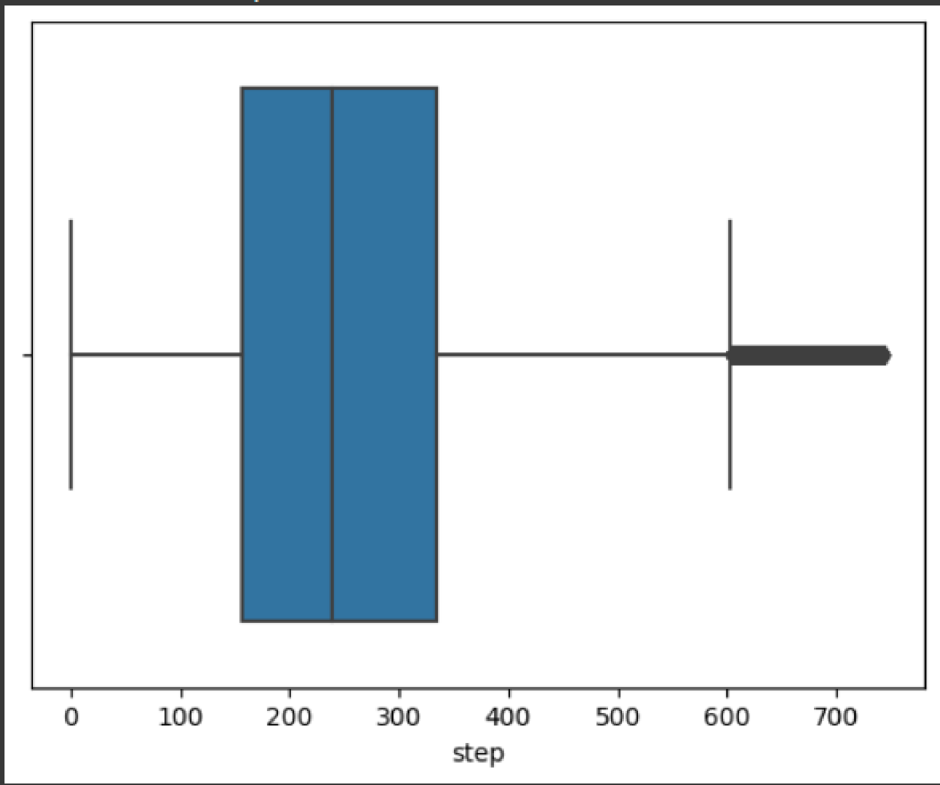
```
sns.countplot(data=df,x='type')
```

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



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

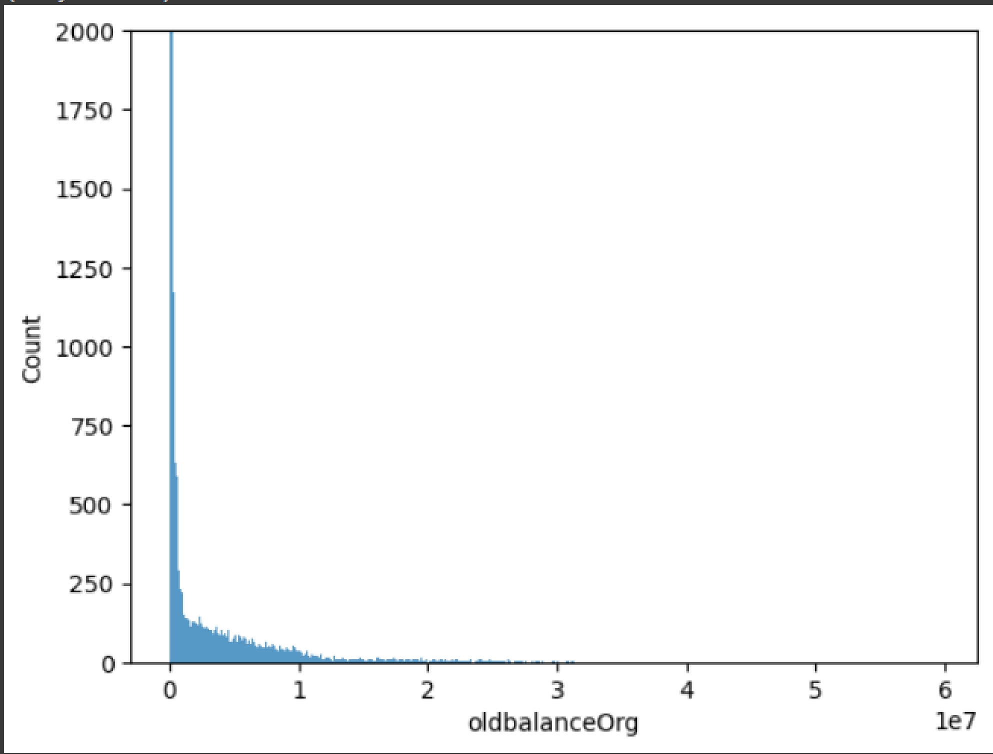
```
<Axes: xlabel='step'>
```



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

```
import matplotlib.pyplot as plt
sns.histplot(data=df,x='oldbalanceOrg')
plt.ylim(0,2000)
```

```
(0.0, 2000.0)
```

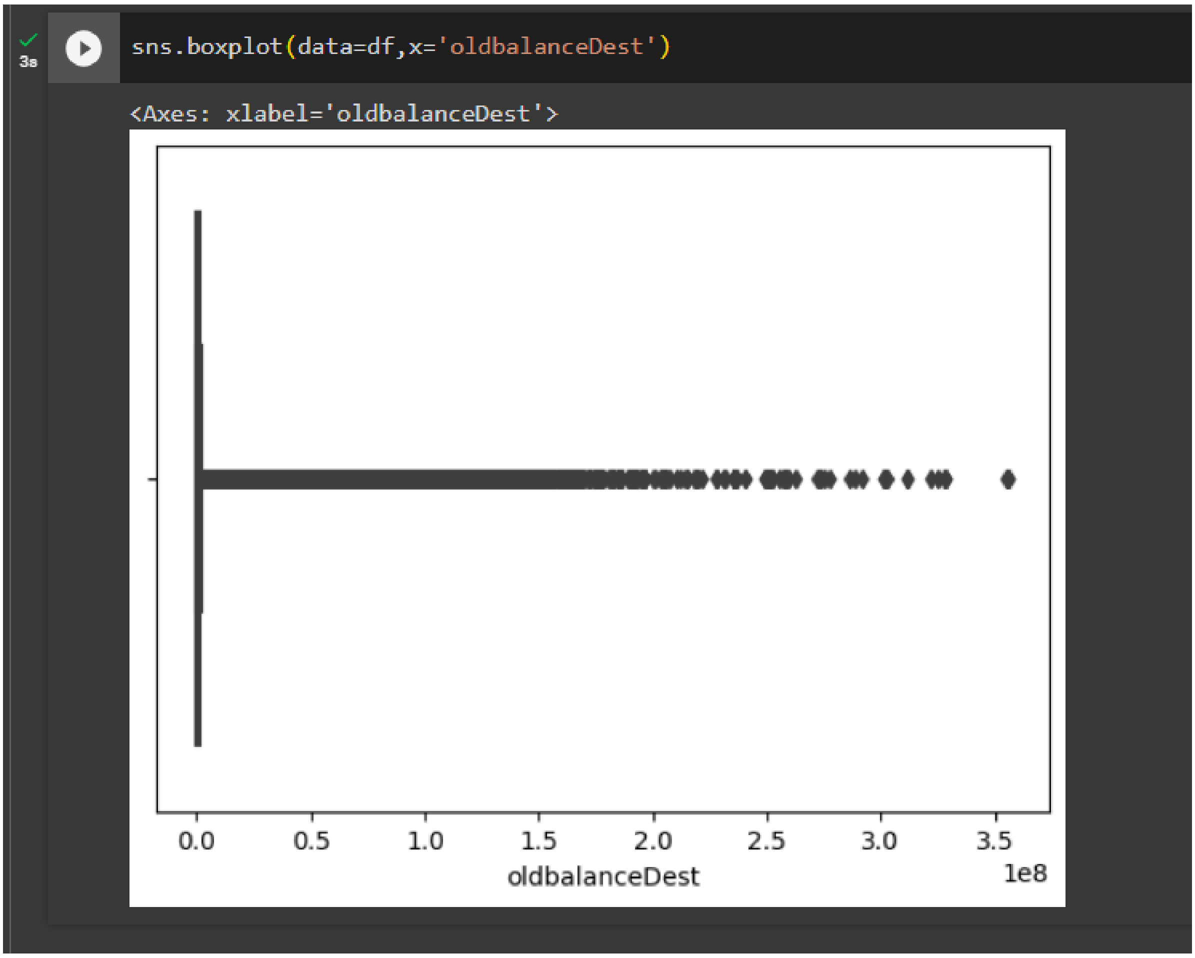


By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the oldbalanceOrg attribute in the dataset. We set the limit as 2000 so that we can view on the required scale

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

```
C1286084959    113
C985934102     109
C665576141     105
C2083562754    102
C248609774     101
              ...
M1470027725      1
M1330329251      1
M1784358659      1
M2081431099      1
C2080388513      1
Name: nameDest, Length: 2722362, dtype: int64
```

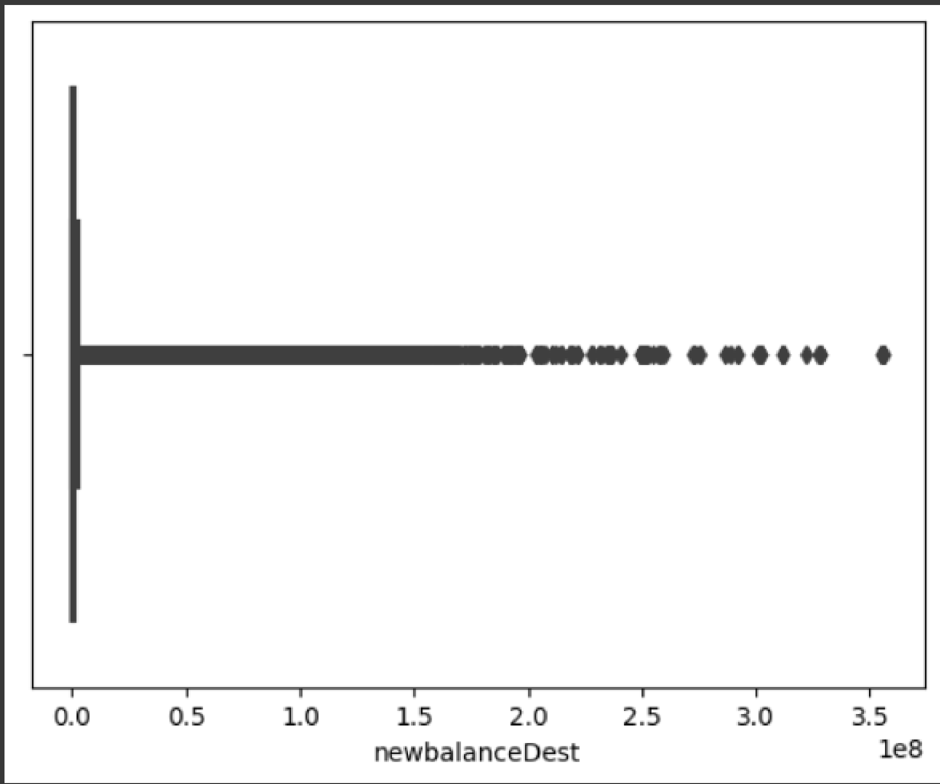
utilising the value counts() function here to determine how many times the nameDest column appears.



Here, the relationship between the oldbalanceDest visulaized is visualised using boxplot

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

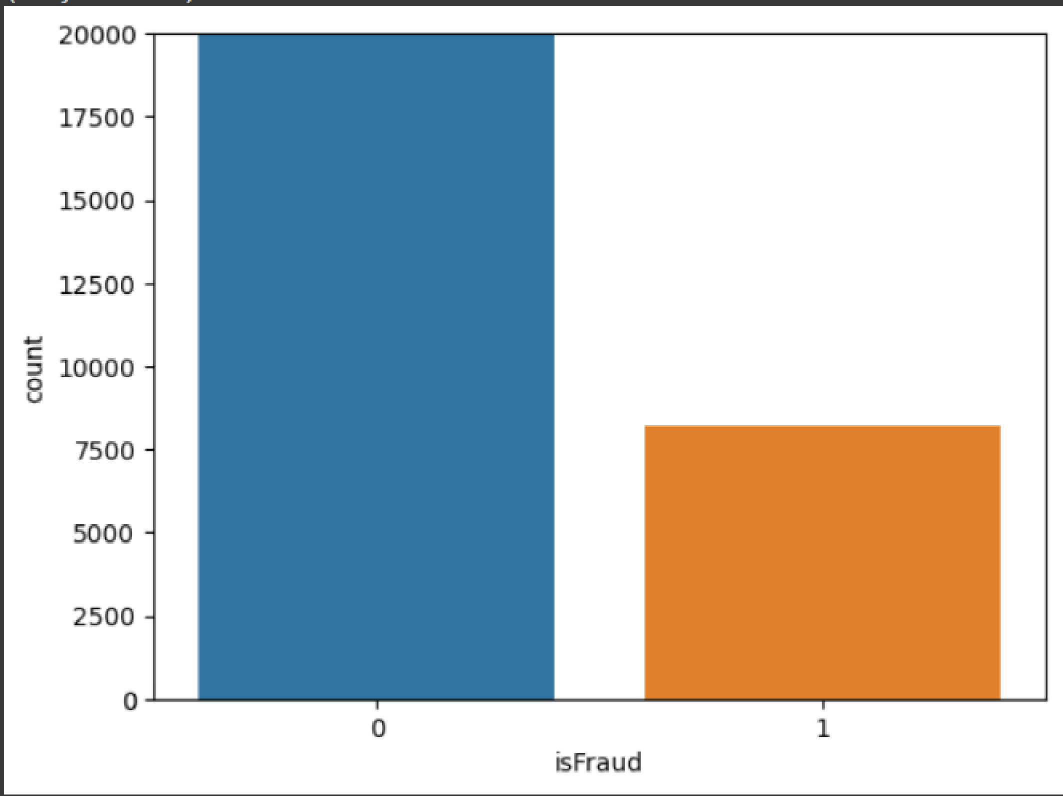
```
<Axes: xlabel='newbalanceDest'>
```



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

```
[50] sns.countplot(data=df,x='isFraud')
     plt.ylim(0,20000)
```

```
(0.0, 20000.0)
```



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

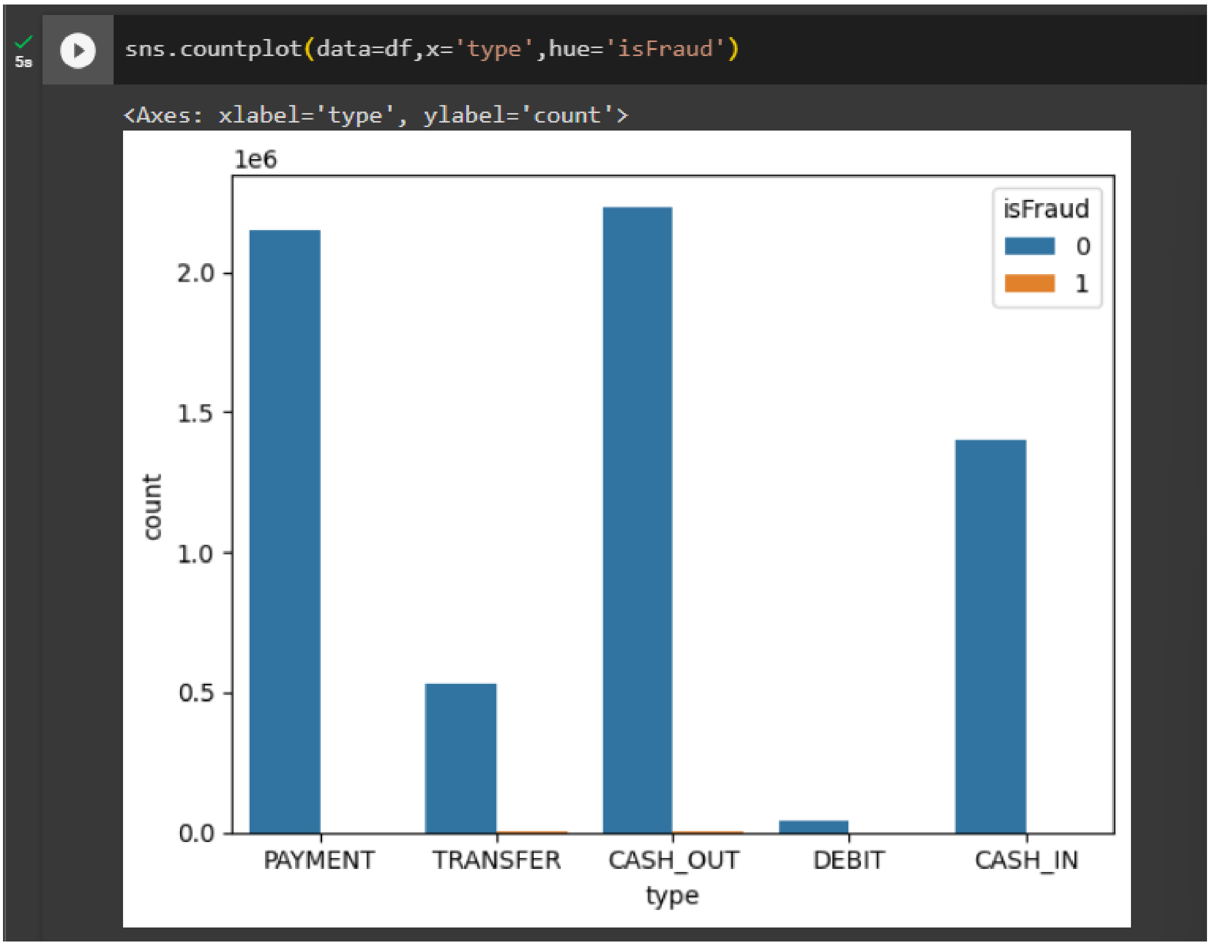
```
df['isFraud'].value_counts()
```

```
0    6354407
1       8213
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.

Activity 4: Bivariate analysis

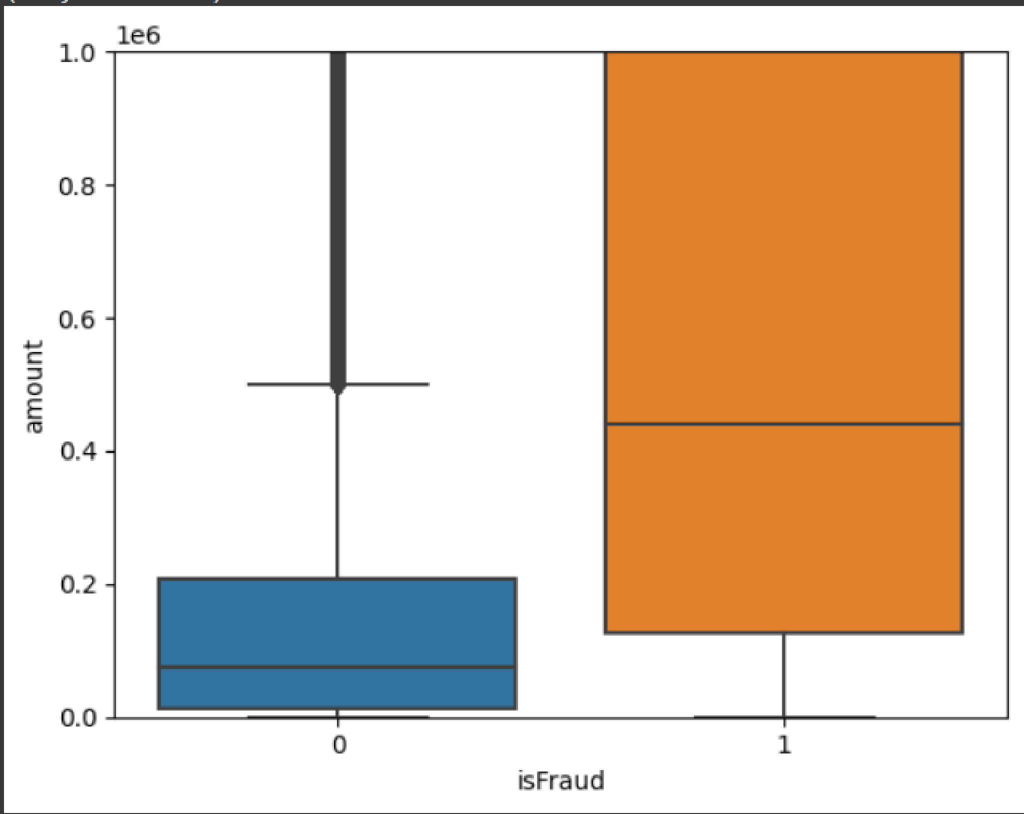
To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between type and isFraud using count Plot.



Here we are visualising the relationship between isFraud and amount using boxplot

```
[94] sns.boxplot(data=df,x='isFraud',y='amount')
     plt.ylim(0,1000000)
```

```
(0.0, 1000000.0)
```



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.

```
[58] df.describe()
```

step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03	2.514687e-06
423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02	1.585775e-03
000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00	0.000000e+00
350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00	0.000000e+00
430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00	1.000000e+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

Removing outlier

Splitting dependent and Independent attributes

Handling Object data label encoding

Feature Scaling

Splitting dataset into training and test

```
[56] df.shape
(6362620, 11)
```

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

```
[57] 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
```

df.info is used to figure out the attribute information

```
#removing unnecessary attributes
df = df[['type','amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newbalanceDest',"isFraud"]]
```

here, by using above code I have removed unnecessary attributes

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 7 columns):
 #   Column          Dtype
---  ------          -----
 0   type            object
 1   amount          float64
 2   oldbalanceOrg   float64
 3   newbalanceOrig  float64
 4   oldbalanceDest  float64
 5   newbalanceDest  float64
 6   isFraud         int64
dtypes: float64(5), int64(1), object(1)
memory usage: 339.8+ MB
```

Again using df.info after removing unnecessary attribute

1: Checking for null values

Checking for Null Values

```
[59] df.isnull().any()

step              False
type              False
amount            False
nameOrig          False
oldbalanceOrg     False
newbalanceOrig    False
nameDest          False
oldbalanceDest    False
newbalanceDest    False
isFraud           False
isFlaggedFraud    False
dtype: bool
```

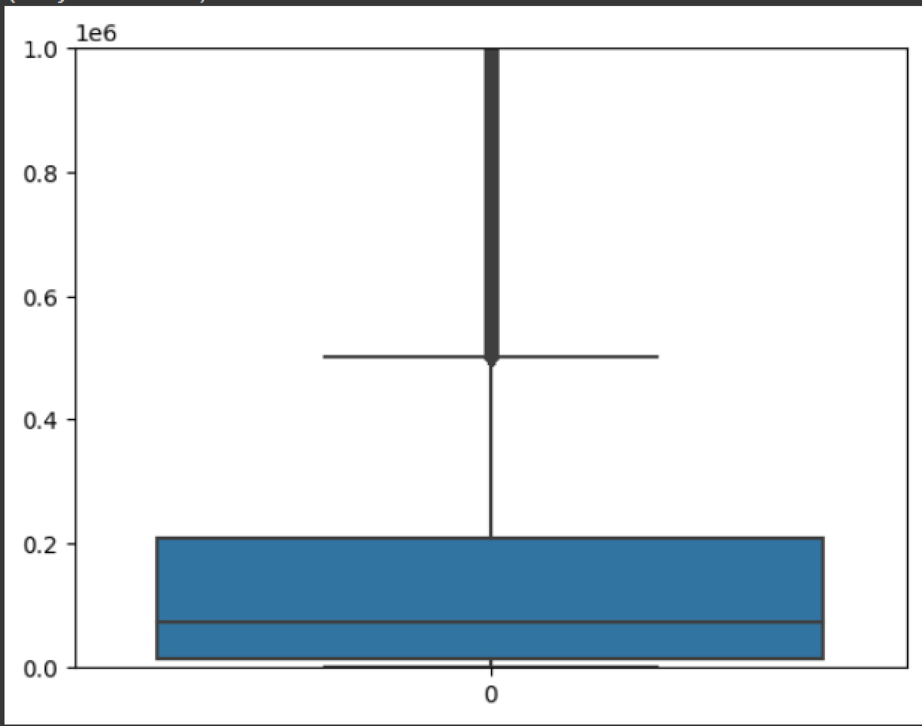
```
[60] df.isnull().sum()

step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

Checking for null value by using isnull and using function .sum() and .any() on top of it. From our observation there are no null values

2: Handling outliers

```
sns.boxplot(df['amount'])
plt.ylim(0,1000000)
```

```
(0.0, 1000000.0)
```

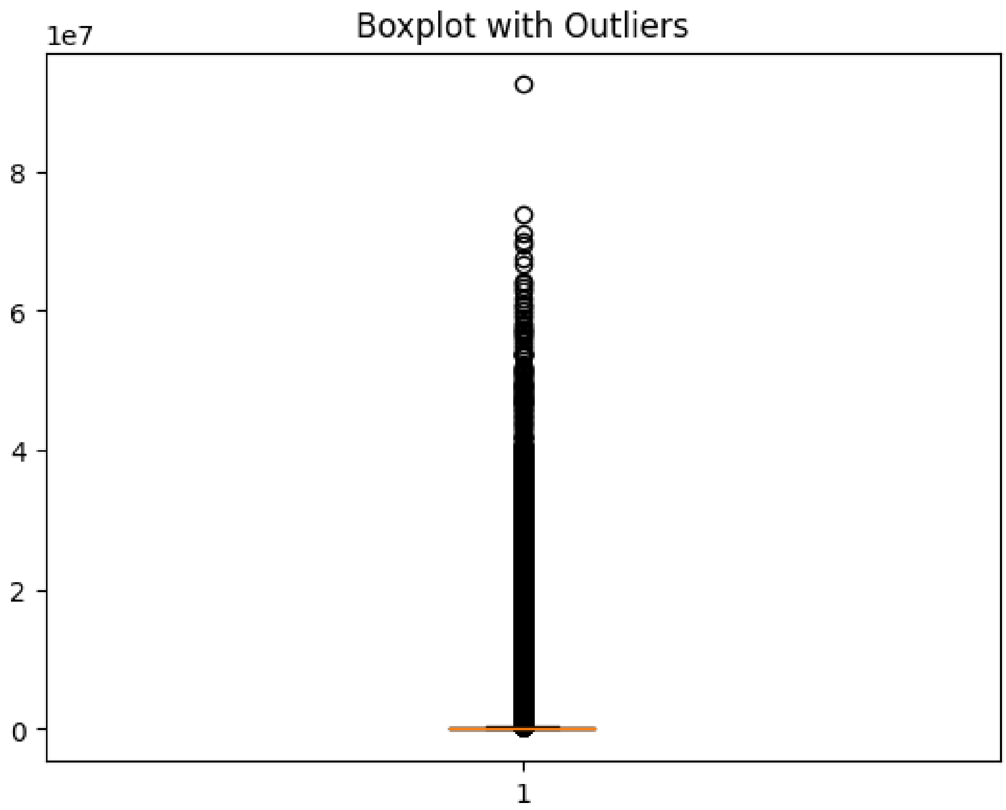


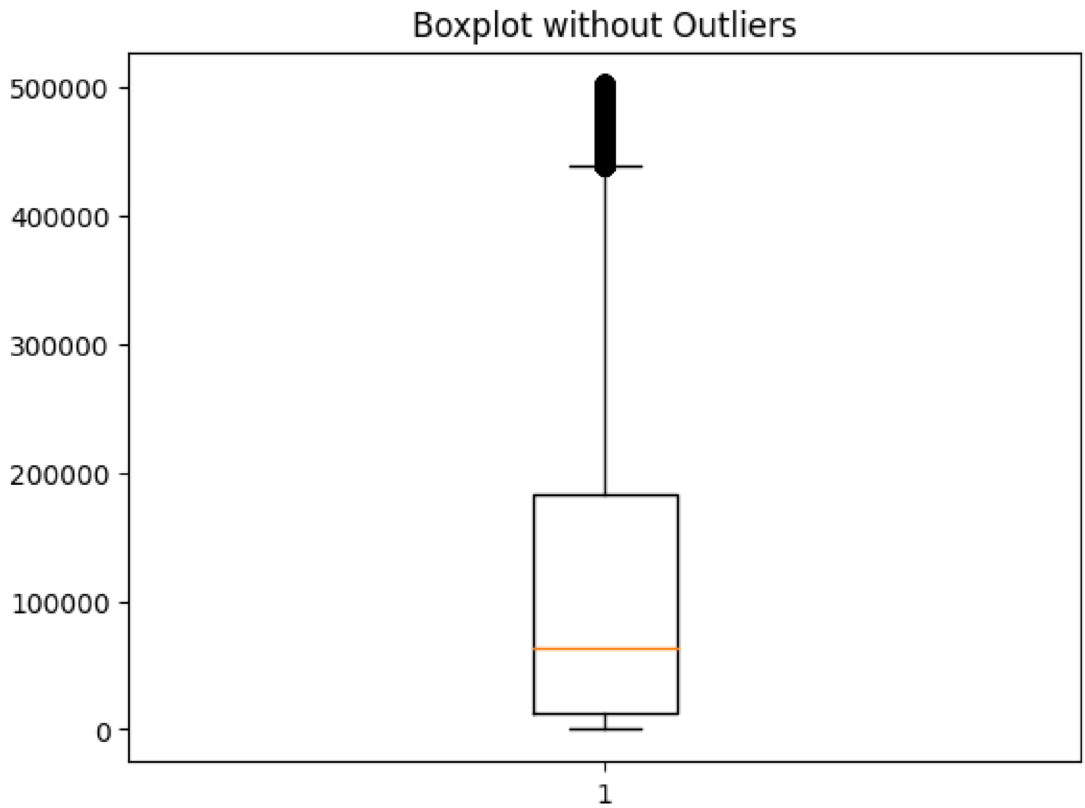
Here, a boxplot is used to identify outliers in the dataset's amount attribute.

Remove Outlier

[89]

```
import numpy as np
import matplotlib.pyplot as plt

# Generate some random data with outliers
data = df['amount']

# Create a boxplot to visualize the data and identify outliers
plt.boxplot(data)
plt.title('Boxplot with Outliers')
plt.show()

# Calculate the IQR (Interquartile Range)
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1

# Define the lower and upper bounds to identify outliers
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

# Filter out the outliers
filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

# Create a boxplot of the filtered data
plt.boxplot(filtered_data)
plt.title('Boxplot without Outliers')
plt.show()
```


Boxplot with Outliers
1e7
8
6
4
2
0
1

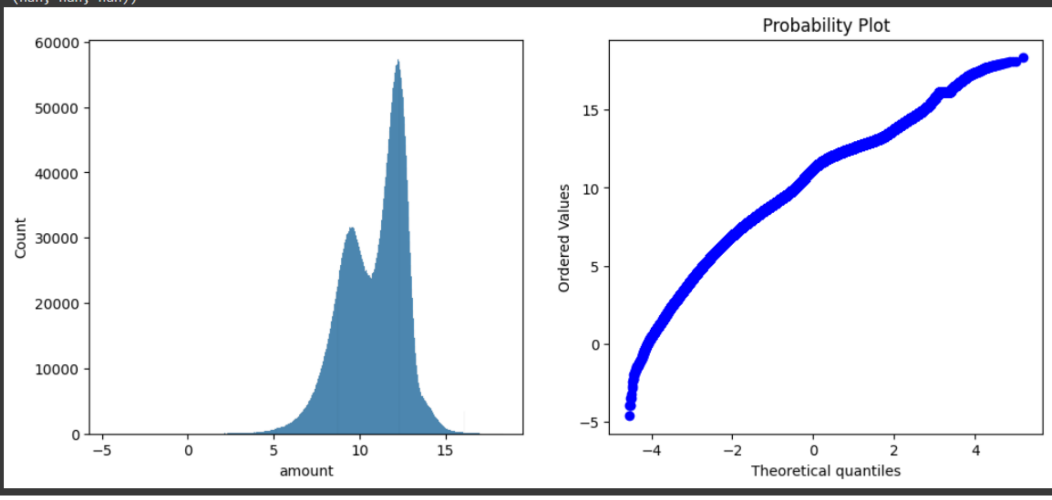

Boxplot without Outliers
500000
400000
300000
200000
100000
0
1

```python
[93] from scipy import stats
     import matplotlib.pyplot as plt
     feature=np.log(df['amount'])
     plt.figure(figsize=(12,5))
     plt.subplot(1,2,1)
     sns.histplot(feature)
     plt.subplot(1,2,2)
     stats.probplot(feature,plot=plt)
```

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

Output for it is as follows

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:402: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2698: RuntimeWarning: invalid value encountered in subtract
  X -= avg[:, None]
((array([-5.1833966 , -5.01553166, -4.92509216, ...,  4.92509216,
          5.01553166,  5.1833966 ]),
  array([       -inf,        -inf,        -inf, ..., 18.08061679,
         18.11718754, 18.34213002])),
 (nan, nan, nan))
```



3: Splitting data into dependent and independent

Now let’s split the Dataset into x and y: first split the dataset into x and y.

Before splitting into independent attribute

Splitting Dependent and Independent variables

```
[46] df.head()
```

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

X and y After splitting into df into dependent dataframe named as X and independent as y

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

```
X.head()
```

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

```
[48] y.head()
```

```
0    0
1    0
2    1
3    1
4    0
Name: isFraud, dtype: int64
```

4: Object data label encoding

.info() to see the object attribute

Encoding

```
X.info() # only need to encode type as there is no other string value
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   type            object
 1   amount          float64
 2   oldbalanceOrg   float64
 3   newbalanceOrig  float64
 4   oldbalanceDest  float64
 5   newbalanceDest  float64
dtypes: float64(5), object(1)
memory usage: 291.3+ MB
```

Displaying the X before label encoding

```
[62] X.head()
```

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

Use LabelEncoder package for label encoding.Store the original classes as a dictionary

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
X.type=le.fit_transform(X.type)
mappingType=dict(zip(le.classes_,range(len(le.classes_))))
X.head()
```

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

```
[64] mappingType
{'CASH_IN': 0, 'CASH_OUT': 1, 'DEBIT': 2, 'PAYMENT': 3, 'TRANSFER': 4}
```

5: Feature Scaling

Feature scaling is very important as it decreases the spread of data by scaling dawn on the basis of it something such as in MinMaxScaler the values are generally scaled down such that minimum value is 0 and maximum value is 1

Before feature scaling

Feature Scaling

```
X.head()
```

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

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
X_scaled=pd.DataFrame(ms.fit_transform(X),columns=X.columns)
```

Above we used minMax scaler for the feature scaling purpose

```
X_scaled.head()
```

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	0.75	0.000106	0.002855	0.003233	0.000000	0.0
1	0.75	0.000020	0.000357	0.000391	0.000000	0.0
2	1.00	0.000002	0.000003	0.000000	0.000000	0.0
3	0.25	0.000002	0.000003	0.000000	0.000059	0.0
4	0.75	0.000126	0.000697	0.000603	0.000000	0.0

6: Splitting Dataset into train and test

Splitting Data into Train and Test

```
[68] from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size =0.2,random_state =0)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
((5090096, 6), (1272524, 6), (5090096,), (1272524,))
```

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.

1: Logistic Regression

Importing the model building libraries

Importing the model building Libraries

```
from sklearn.linear_model import LogisticRegression
```

Initializing the model

Initialize the model

```
[72] model=LogisticRegression()
```

```
model
```

```
▾ LogisticRegression
LogisticRegression()
```

Training and Testing the model

In the following code we trained and also made a dataframe for orginal and predicted value

Training and Testing the model

```
[74] model.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
[75] pred=model.predict(x_test)
```

```
res = pd.DataFrame({'Original Value':y_test,'Predicted Value':pred})
res.head(5)
```

	Original Value	Predicted Value
4644207	0	0
3800666	0	0
4426240	0	0
5788765	0	0
2010701	0	0

Evaluation of the model

Evaluation of model

```
[77] # all the libraries of evaluating model
     from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

Importing all the necessary library for the model evaluation

Printing the accuracy of the model

```
print(accuracy_score(y_test,pred))
0.9988078810301416
```

Hence the accuracy of the model is 99.88 percentage

Printing confusion matrix

```
[79] confusion_matrix(y_test,pred)

array([[1270880,       3],
       [   1514,     127]])
```

Printing cross-tabulation matrix

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

col_0	0	1
isFraud		
0	1270880	3
1	1514	127

Printing the classification report

```
[82] print(classification_report(y_test,pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270883
           1       0.98      0.08      0.14      1641

    accuracy                           1.00   1272524
   macro avg       0.99      0.54      0.57   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Finding probability estimate of classes for the ROC curve

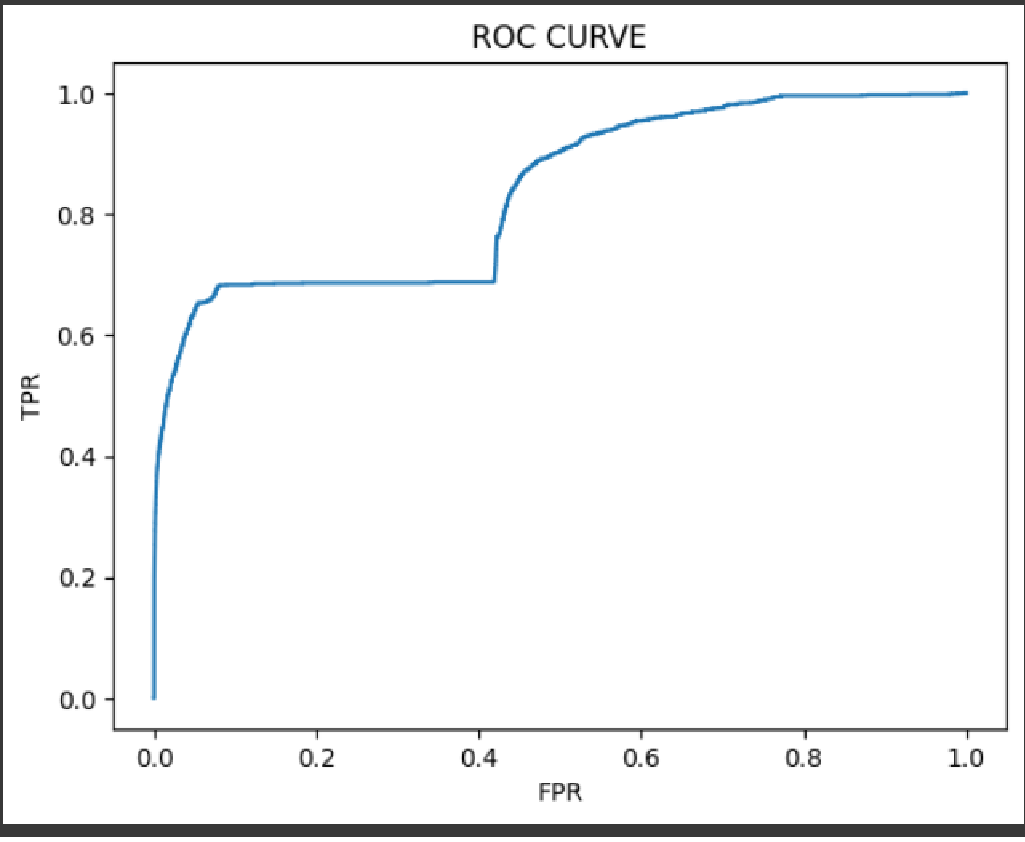
```
[85] probability=model.predict_proba(x_test)[:,1]
     probability

array([0.00034006, 0.00169209, 0.00082203, ..., 0.00079488, 0.00073166,
       0.00235239])
```

Finding fpr,tpr and thresholds of roc curve from using roc curve function

```
[86] fpr,tpr,threshsholds = roc_curve(y_test,probability)
```

Ploting the ROC curve

```
[87] plt.plot(fpr,tpr)
     plt.xlabel('FPR')
     plt.ylabel('TPR')
     plt.title('ROC CURVE')
     plt.show()
```



Saving the model

Use pickle package to save the logistic regression model and min max scaler model.

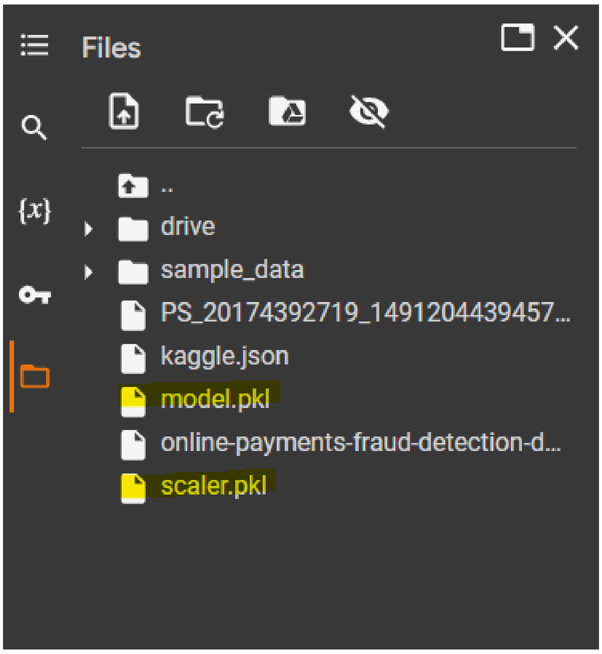
Use with to open a new file named as model.pkl and dump the model to file using pickle dump function

Save the model

```
[95] import pickle
     with open('scaler.pkl', 'wb') as file:
         pickle.dump(ms, file)
     with open('model.pkl', 'wb') as file:
         pickle.dump(model, file)
```

After running you will get two file named as model.pkl and scaler.pkl in Files section .

Download those We will be using those in building application



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

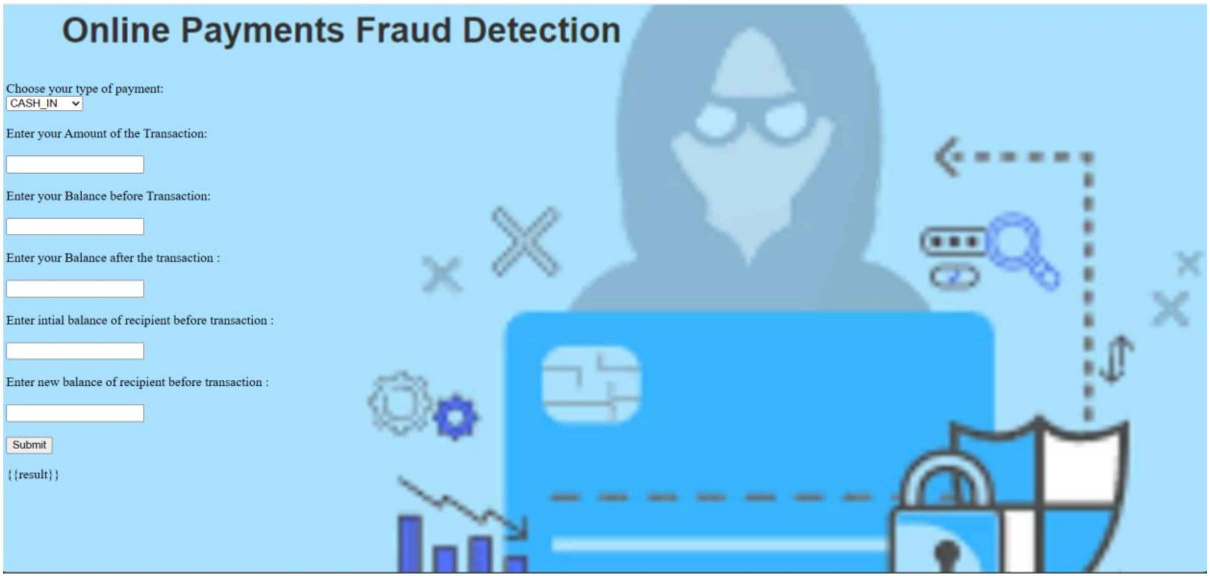
Building server side script

Activity1: Building Html Pages:

For this project create one HTML files namely

- home.html
- result.html

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



Now when you click on Submit button after filling all the detail you will be redirected to result.html

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

{{ result }}

result.html is blank html page which will change colour on the basis of result value If there no fradulant transaction , green colour with written no fradulant transaction written otherwise in case of fradulant transaction red colour with written fradulant transaction

Activity 2: Build Python code:

Import the libraries

```
from flask import Flask,render_template,request, redirect, url_for
import pickle
import numpy as np
# loading my mlr model
model=pickle.load(open('model.pkl','rb'))
#loading Scaler
scalar=pickle.load(open('scaler.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.

```
# Flask is used for creating your application
# render template is use for rendering the html page


app= Flask(__name__)  # your application
```

Render HTML page:

```
@app.route('/')  # default route
def home():
    return render_template('home.html') # rendering if your home page.
```

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( rule: '/pred',methods=['POST']) # prediction route
def predict():
    td= request.form["type"]
    ad = request.form["amount"]
    obo = request.form["oldbalanceOrg"]
    nbo = request.form["newbalanceOrg"]
    obd = request.form["oldbalanceDest"]
    nbd = request.form["newbalanceDest"]

    t = [[float(td),float(ad),float(obo),float(nbo),float(obd),float(nbd)]]

    x=scalar.transform(t)
    output =model.predict(x)
    print(output)
    return redirect(url_for( endpoint: 'result', result=np.round(output[0])))
```

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 result.html page earlier.

Routing to result.html

```
@app.route('/result/<int:result>')
def result(result):
    if result == 0:
        result_class = "no-fraud"
        message = "No Fraudulent Transaction"
    else:
        result_class = "fraud"
        message = "Fraudulent Transaction!!!!!!"

    return render_template( template_name_or_list: 'result.html', result_class=result_class,result=message)
```

According to value of result we give message . The point of notice here is result_class

In HTML file of result that is result.html we have used dynamic CSS styling

Result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result Page</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style_result.css') }}">
</head>
<body class="{{ result_class }}">
<h1>{{ result }}</h1>
</body>
</html>
```

Here style_result.css is styling file for result.html

```css
body {
    text-align: center;
    padding: 50px;
}

.no-fraud {
    background-color: green;
    color: white;
}

.fraud {
    background-color: red;
    color: white;
}

```

NOTE: Above according to the value of result_class that is fraud and no_fraud the different colour will be use. if there is fraud i.e result_class=fraud thus .fraud styling will be used which will change background colour to red

Main Function:

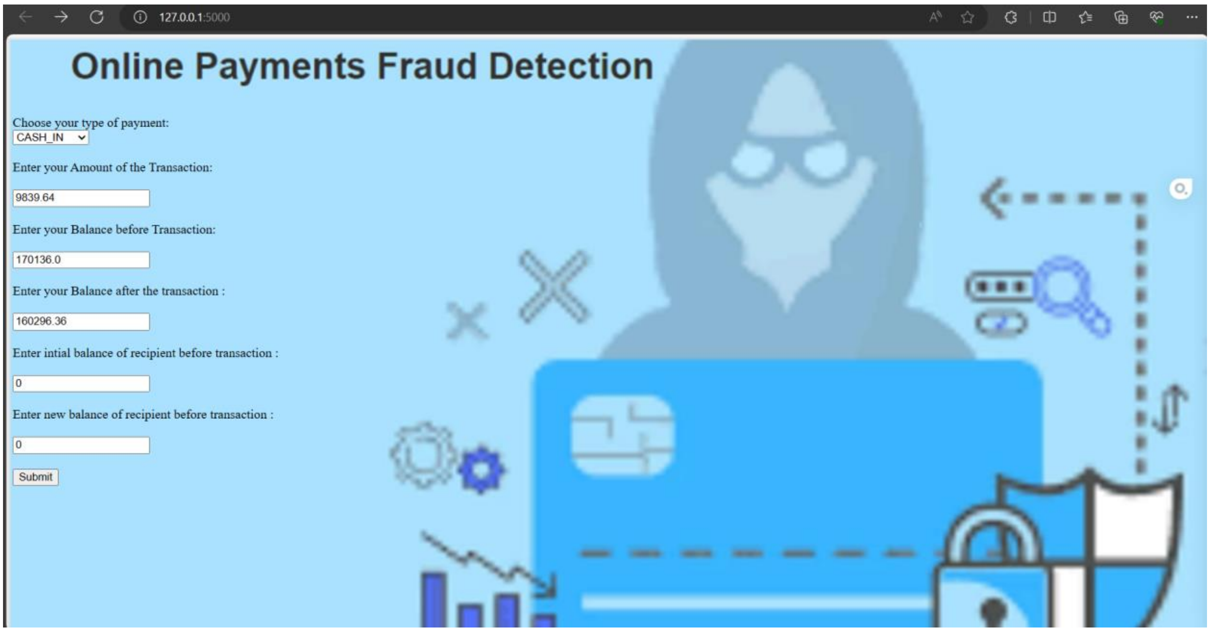
```python
# running your application,
if __name__ == "__main__":
    app.run()

#http://localhost:5000/ or localhost:5000
```

Activity 3: Run the application

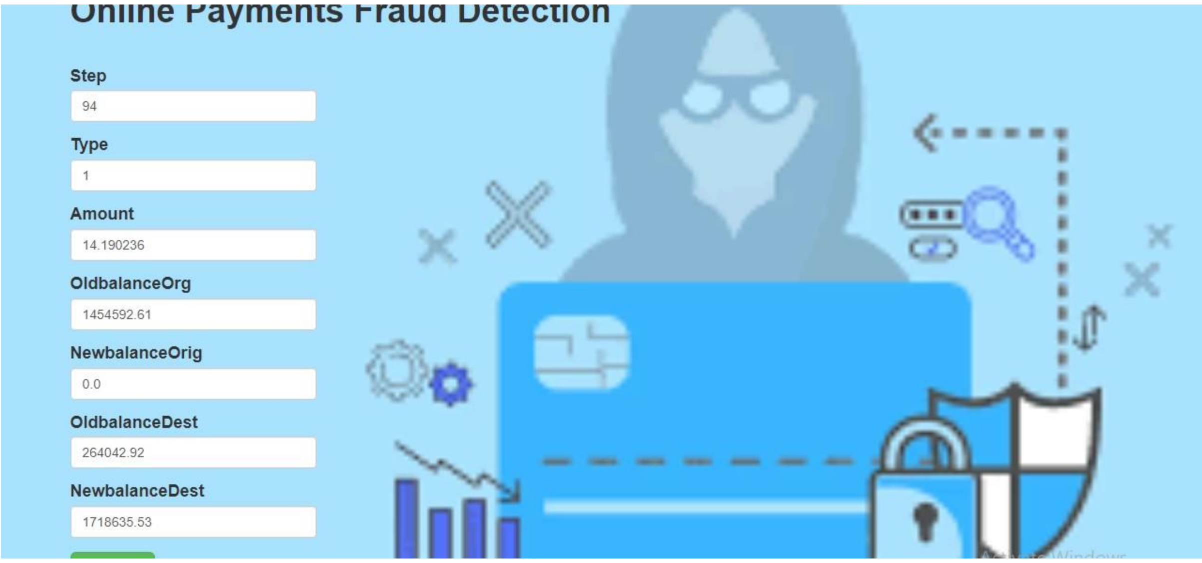
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.

- Fill all the value according to detail and click on submit

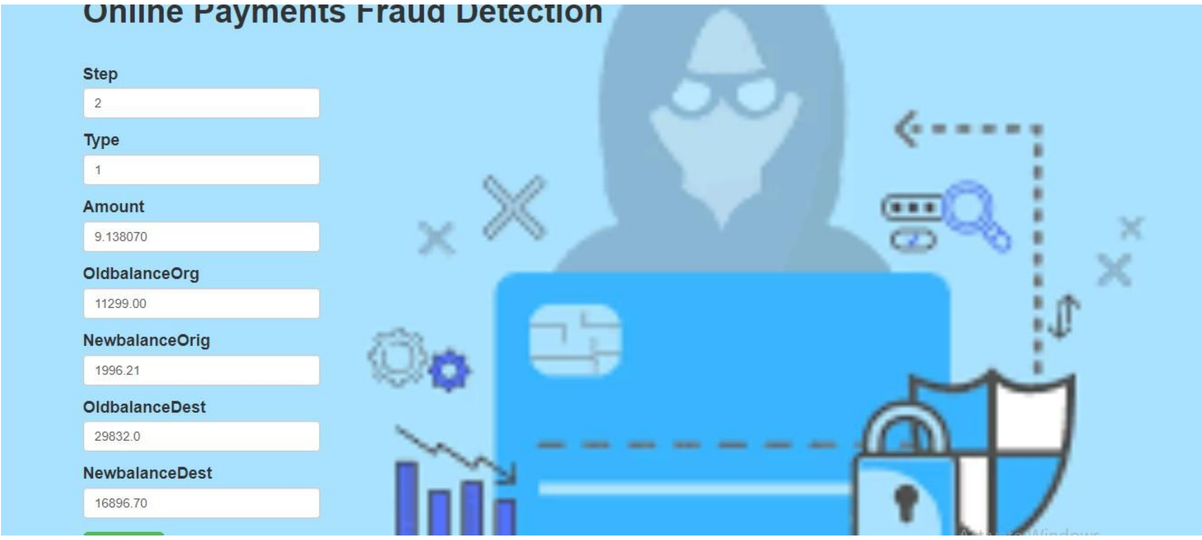
Output screenshots:




127.0.0.1:5000/result/0
No Fraudulent Transaction


Online Payments Fraud Detection
Step
94
Type
1
Amount
14.190236
OldbalanceOrg
1454592.61
NewbalanceOrig
0.0
OldbalanceDest
264042.92
NewbalanceDest
1718635.53


Home
Predict
Online Payments Fraud Detection
The predicted fraud for the online payment is ['is Fraud']


Online Payments Fraud Detection
Step
2
Type
1
Amount
9.138070
OldbalanceOrg
11299.00
NewbalanceOrig
1996.21
OldbalanceDest
29832.0
NewbalanceDest
16896.70


Home
Predict
Online Payments Fraud Detection
The predicted fraud for the online payment is ['is not Fraud']