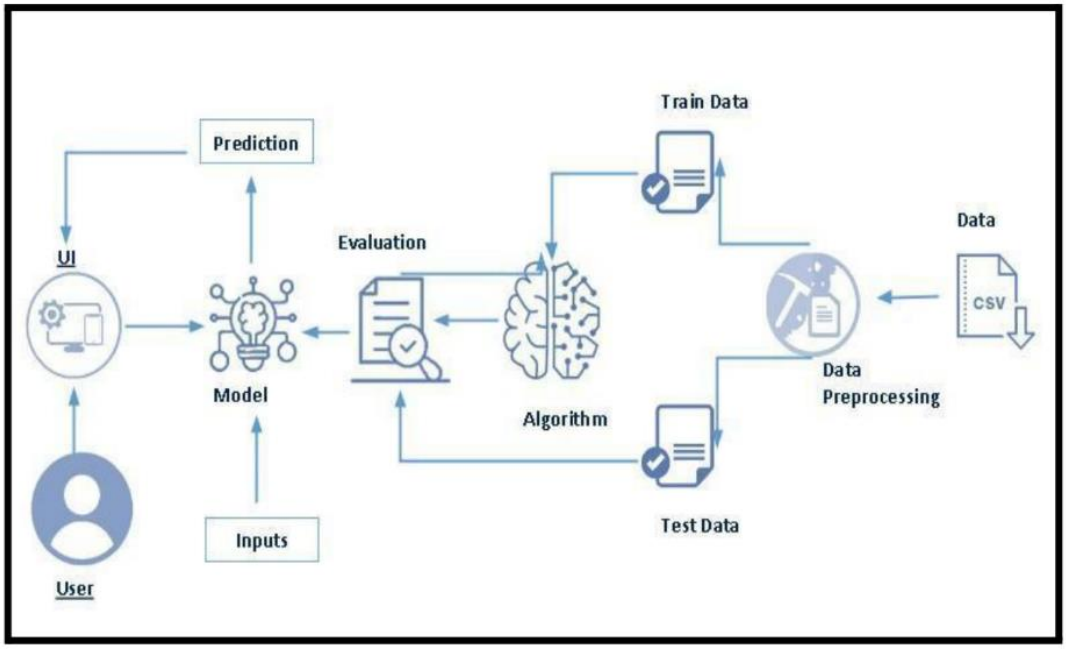
Online Payments Fraud Detection Using ML

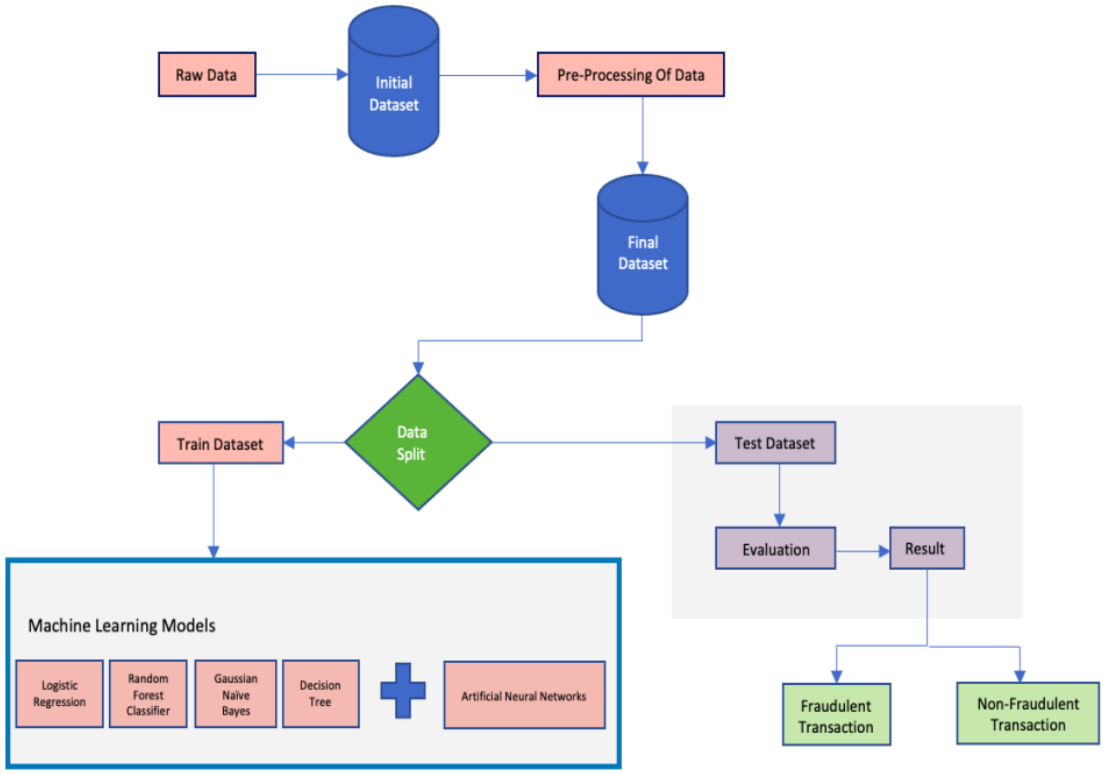
Project Description:

Online payments have become more popular during the last few decades. This is because it's so simple to send money from anywhere, but the pandemic has also contributed significantly to the rise in e-payments. Numerous studies have demonstrated that e-commerce and online payments will continue to grow in popularity in the years to come. The risk of online payment fraud has also increased as a result of this rise in online payments. Online payment fraud has been shown to have increased over the past few years, making it crucial for consumers and service providers to be aware of these frauds. It is crucial for users to be certain that the payments they make are going to the legitimate recipients; otherwise, they run the risk of having to report fraud, freeze their payment 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:

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

- **Anaconda navigator and pycharm:**
  - o Refer the link below to download anaconda navigator
  - o Link : https://youtu.be/1ra4zH2G4o0

- **Python packages:**
  - o Open anaconda prompt as administrator
  - o Type"pip install numpy"and click enter.
  - o Type"pip install pandas"andclickenter.
  - o Type"pip install scikit-learn"andclickenter.
  - o Type"pip install matplotlib"andclickenter.
  - o Type"pip install scipy"andclickenter.
  - o Type"pip install pickle-mixin"andclickenter.
  - o Type"pip install seaborn"andclickenter.
  - o Type"pipinstallFlask"and click enter.

Prior Knowledge:

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

- **ML Concepts**
  - o Supervisedlearning:
    https://www.javatpoint.com/supervised-machine-learning

o Unsupervisedlearning:
https://www.javatpoint.com/unsupervised-machine-learning

o Regression and classification
o Decisiontree:
https://www.javatpoint.com/machine-learning-decision-tree classification-algorithm

o Randomforest:
https://www.javatpoint.com/machine-learning-random-forest-algorithm

o xgboost Classifier

https://www.javatpoint.com/xgboost-classifier-algorithm-for-machine-learning

o Svm:

https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-Svm/

o Extra tree classifier:

https://www.javatpoint.com/Extratreeclassifier-algorithm-for-machine-learning

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

o 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:

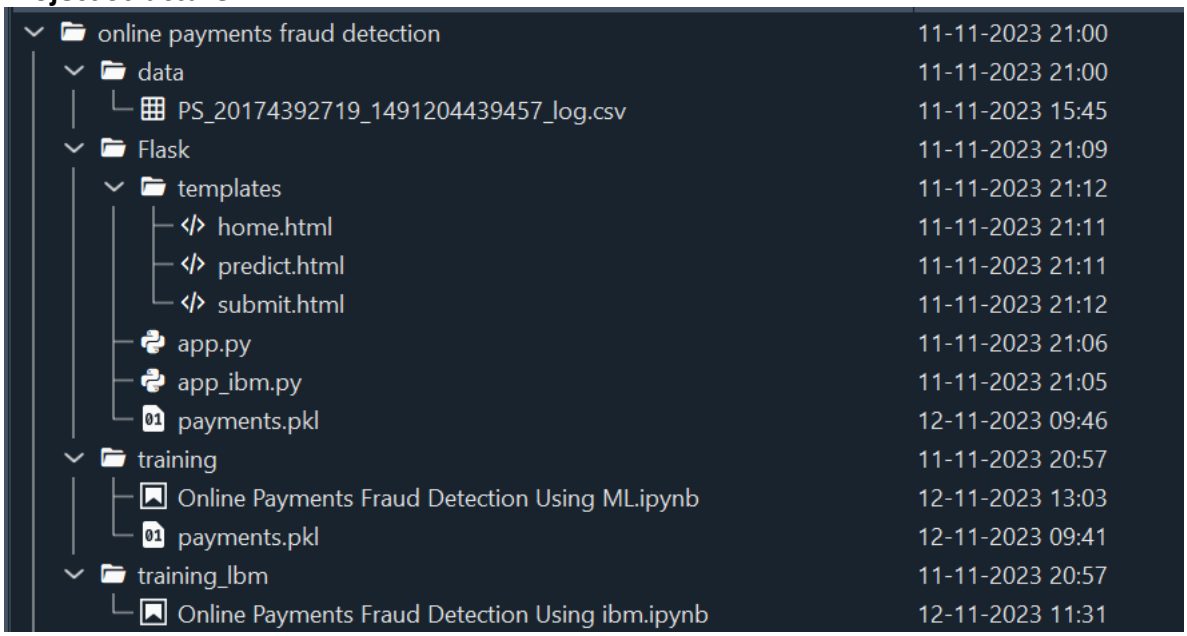
- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

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. Thisdata 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. (optional) Here we have used visualisation style as fivethirtyeight.

Importing Librariers

In [1]:
```
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
from sklearn.metrics import classification_report, confusion_matrix
import warnings
import pickle
```

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.

In [2]:
```
#Importing the dataset.
df = pd.read_csv("Online Payments Fraud Detection Dataset.csv")
```

In [3]:
```
df
```

Out[3]:

	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

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()
```

	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

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

```
df.tail(20)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6362600	742	TRANSFER	652993.91	C40604503	652993.91	0.0	C1166857907	0.00	0.00	1	0
6362601	742	CASH_OUT	652993.91	C1614818636	652993.91	0.0	C362803701	0.00	652993.91	1	0
6362602	742	TRANSFER	1819543.69	C2089752665	1819543.69	0.0	C112833674	0.00	0.00	1	0
6362603	742	CASH_OUT	1819543.69	C1039979813	1819543.69	0.0	C2078394828	0.00	1819543.69	1	0
6362604	742	TRANSFER	54652.46	C1674778854	54652.46	0.0	C1930074465	0.00	0.00	1	0
6362605	742	CASH_OUT	54652.46	C43545501	54652.46	0.0	C830041824	0.00	54652.46	1	0
6362606	742	TRANSFER	303846.74	C959102961	303846.74	0.0	C114421319	0.00	0.00	1	0
6362607	742	CASH_OUT	303846.74	C1148860488	303846.74	0.0	C846260566	343660.89	647507.63	1	0
6362608	742	TRANSFER	258355.42	C1226129332	258355.42	0.0	C1744173808	0.00	0.00	1	0
6362609	742	CASH_OUT	258355.42	C1113162093	258355.42	0.0	C797688696	25176.67	283532.09	1	0
6362610	742	TRANSFER	63416.99	C778071008	63416.99	0.0	C1812552860	0.00	0.00	1	0
6362611	742	CASH_OUT	63416.99	C994950684	63416.99	0.0	C1662241365	276433.18	339850.17	1	0
6362612	743	TRANSFER	1258818.82	C1531301470	1258818.82	0.0	C1470998563	0.00	0.00	1	0
6362613	743	CASH_OUT	1258818.82	C1436118706	1258818.82	0.0	C1240760502	503464.50	1762283.33	1	0
6362614	743	TRANSFER	339682.13	C2013999242	339682.13	0.0	C1850423904	0.00	0.00	1	0
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.0	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.0	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.0	C873221189	6510099.11	7360101.63	1	0

above, the dataset's last 20 values are loaded using the tail method.

```
df.corr()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.c
orr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_onl
y to silence this warning.
  df.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFlaggedFraud
step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.003277
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.012295
oldbalanceOrg	-0.010058	-0.002762	1.000000	0.998803	0.066243	0.042029	0.003835
newbalanceOrig	-0.010299	-0.007861	0.998803	1.000000	0.067812	0.041837	0.003776
oldbalanceDest	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.000513
newbalanceDest	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	-0.000529
isFlaggedFraud	0.003277	0.012295	0.003835	0.003776	-0.000513	-0.000529	1.000000

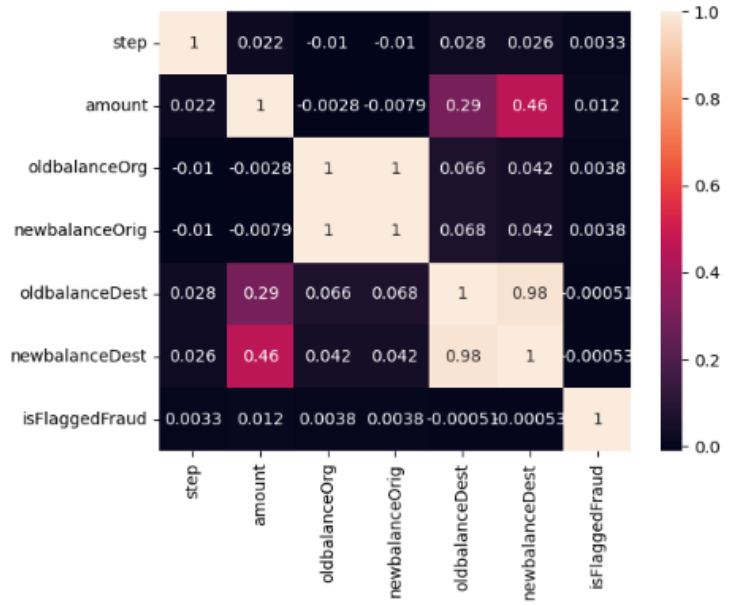
utilising the corr function to examine the dataset's correlation

Heatmap

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\4277794465.py:1: FutureWarning: The default value of numeric_only in DataFrame.c
orr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_onl
y to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

```
<Axes: >
```



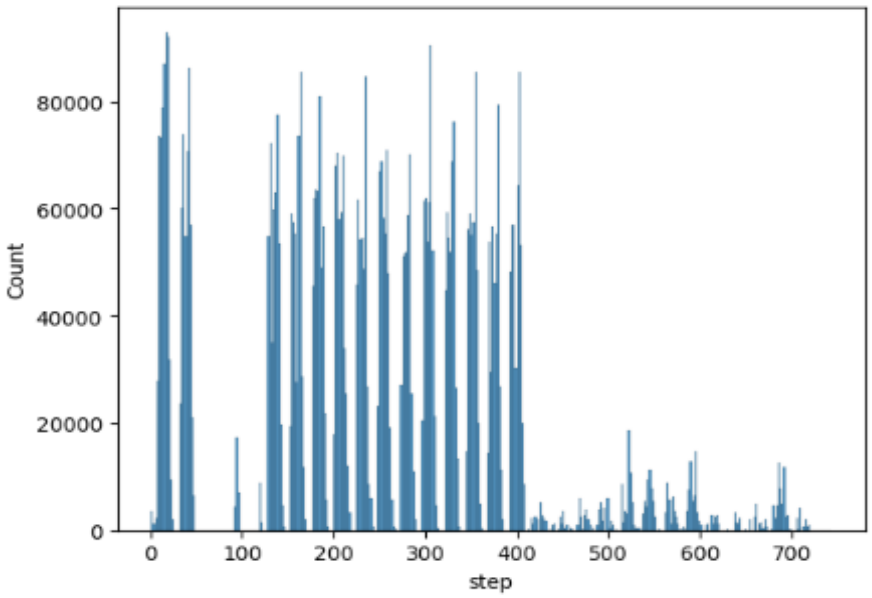
Here, a heatmap is used to understand the relationship between the input attributes and the anticipated goal value.

Activity 3: Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature.

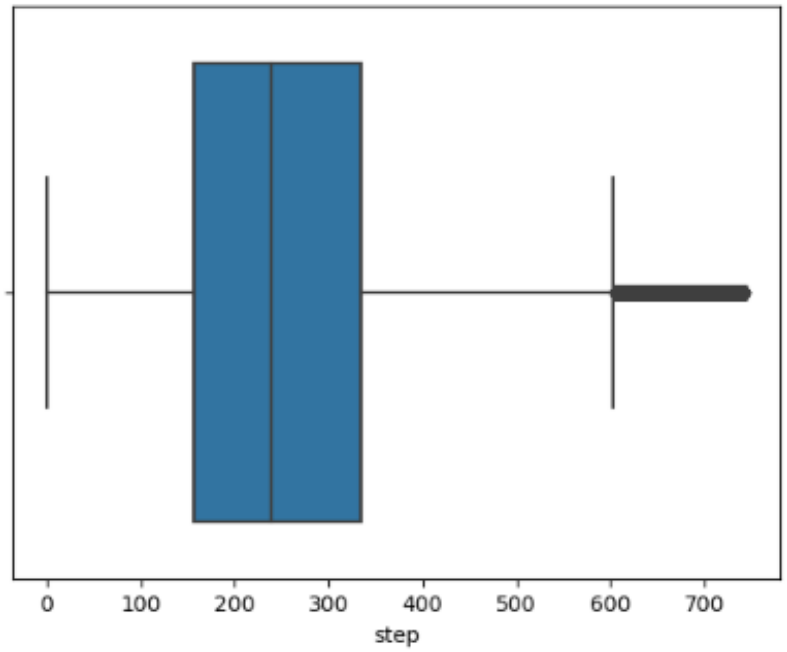
Here I have displayed the graph such as histplot .

```
: sns.histplot(data=df,x='step')
: <Axes: 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.

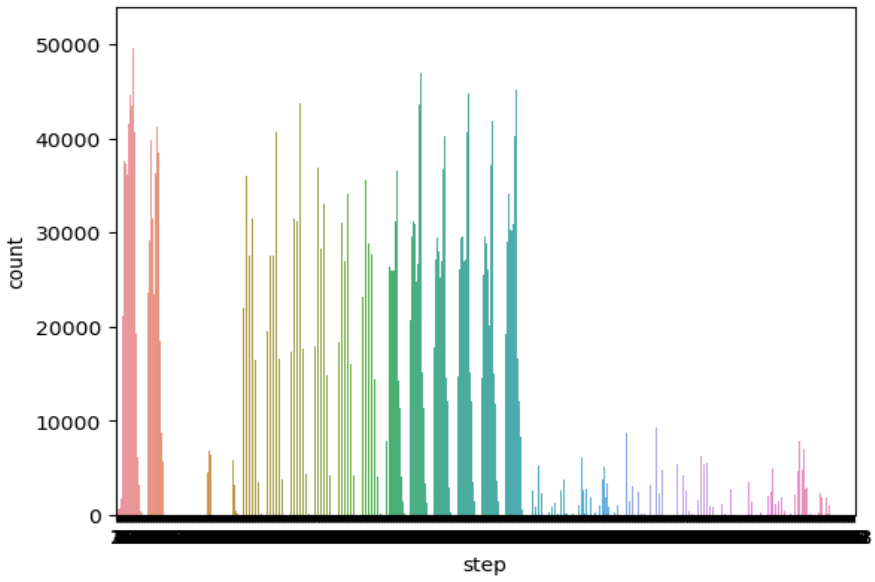
```
: sns.boxplot(x='step',data=df)
: <Axes: xlabel='step'>
```



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

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

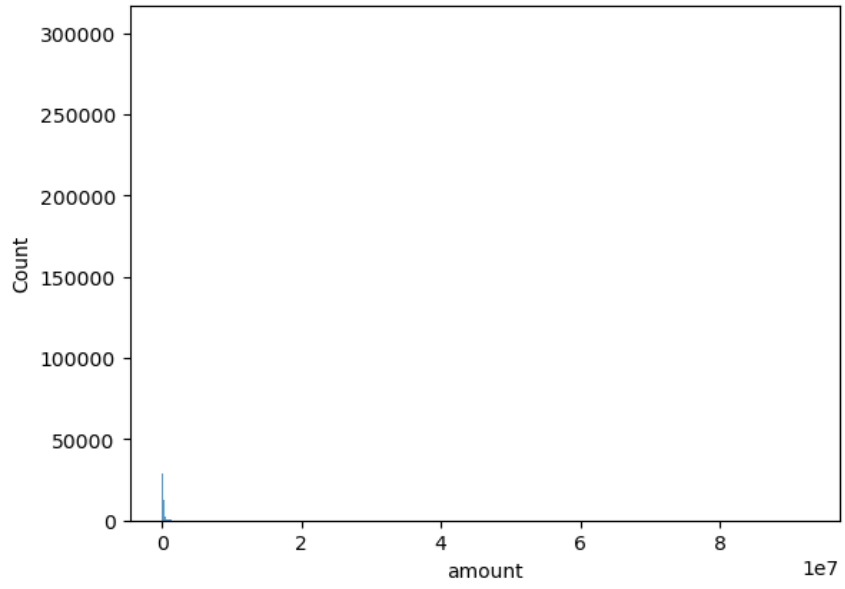
```
<Axes: xlabel='step', ylabel='count'>
```



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

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

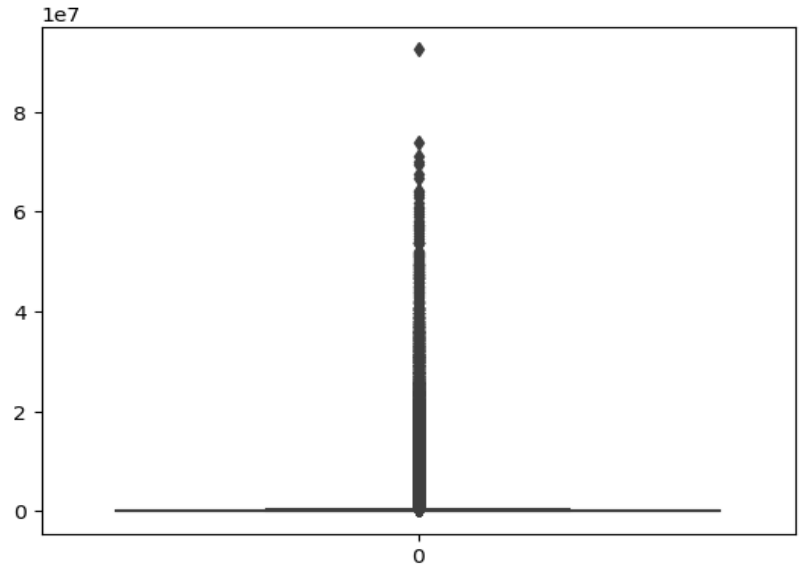
```
: <Axes: xlabel='amount', ylabel='Count'>
```



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.

```
sns.boxplot(df.amount)
#sns.boxplot(df["amount"])
```

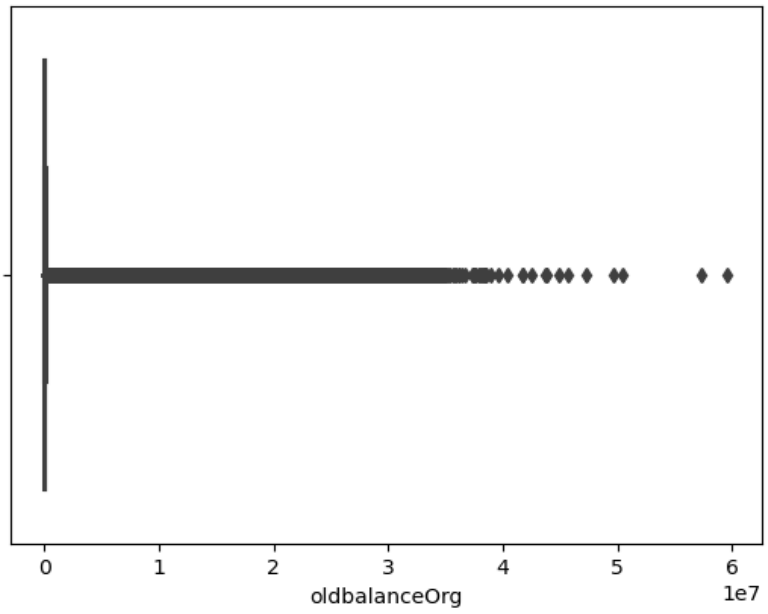
```
<Axes: >
```



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

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

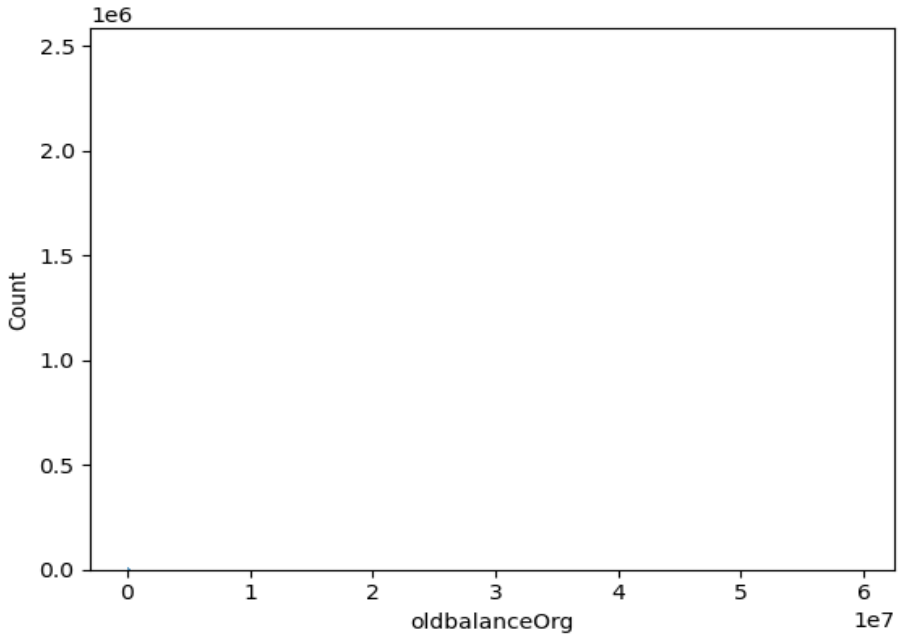
```
: <Axes: xlabel='oldbalanceOrg'>
```



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

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

```
<Axes: 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.

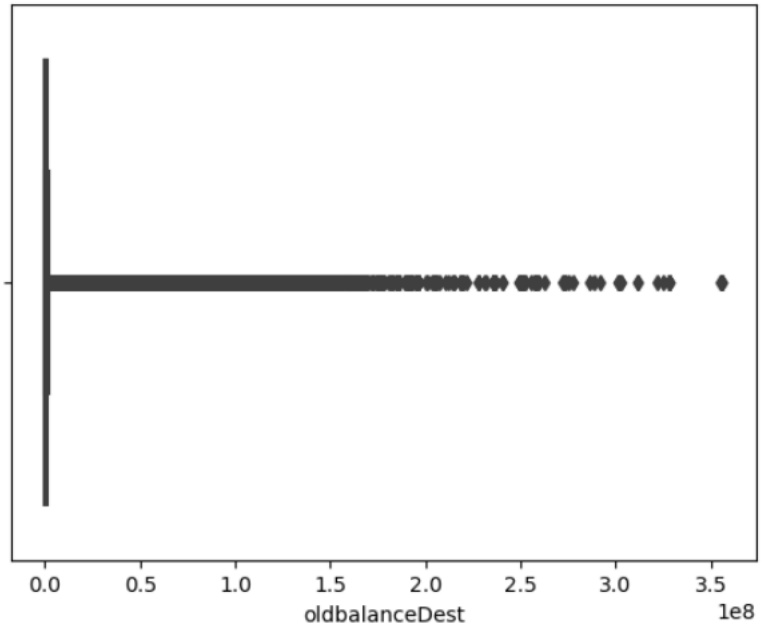
```
df['nameOrig'].value_counts()
```

```
C1902386530    3
C363736674     3
C545315117     3
C724452879     3
C1784010646    3
              ..
C98968405      1
C720209255     1
C1567523029    1
C644777639     1
C1280323807    1
Name: nameOrig, Length: 6353307, dtype: int64
```

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

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

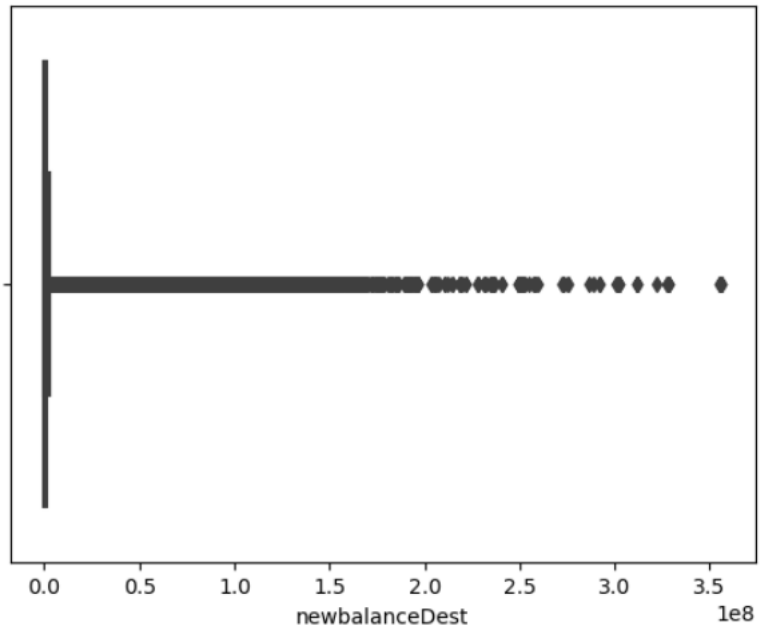
```
<Axes: xlabel='oldbalanceDest'>
```



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

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

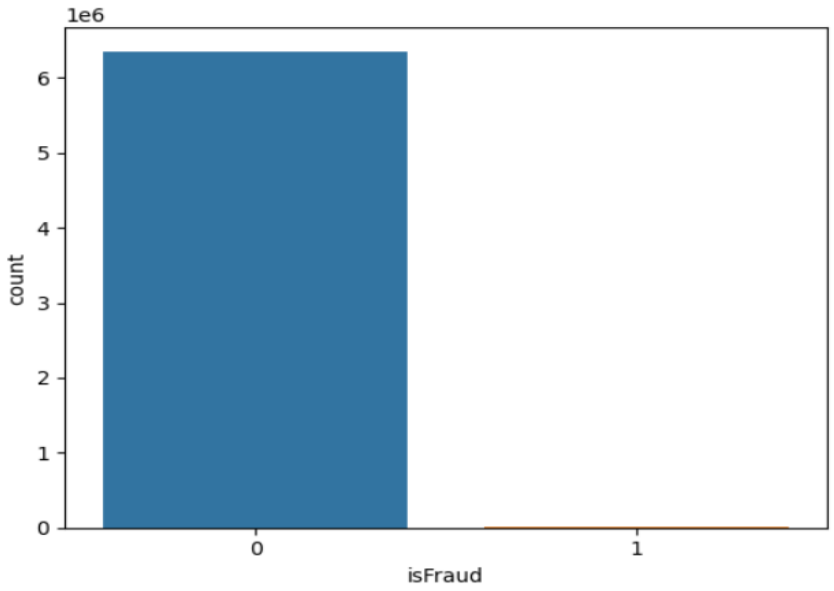
```
<Axes: xlabel='newbalanceDest'>
```



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

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

```
<Axes: 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    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.

```
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	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	is not Fraud	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	is not Fraud	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	is Fraud	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	is Fraud	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	is not Fraud	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	is Fraud	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	is Fraud	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	is Fraud	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	is Fraud	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	is Fraud	0

6362620 rows × 11 columns

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

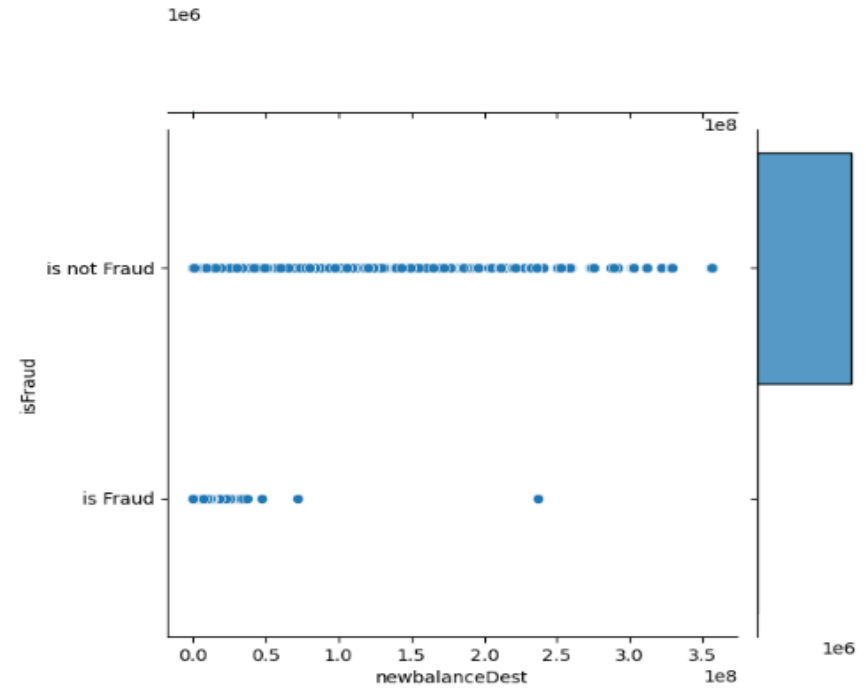
Activity 4: Bivariate analysis

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 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

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

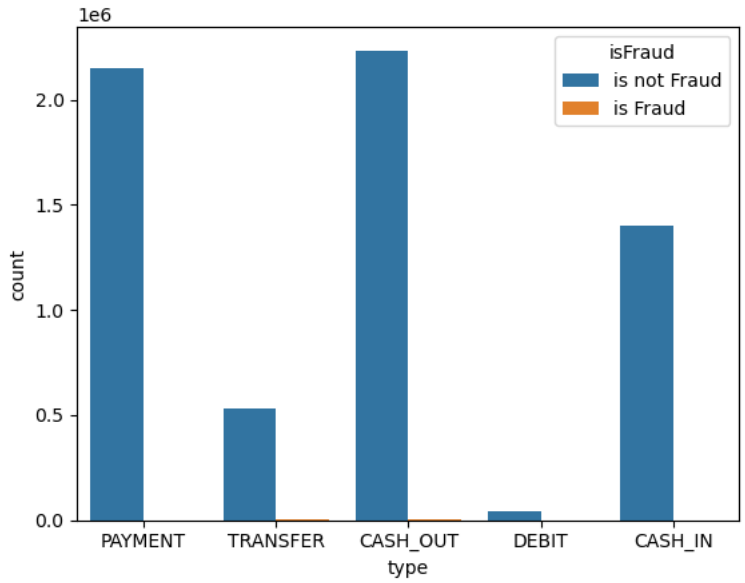
```
<seaborn.axisgrid.JointGrid at 0x20797ff8a50>
```



Here we are visualising the relationship between type and isFraud. Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

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

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



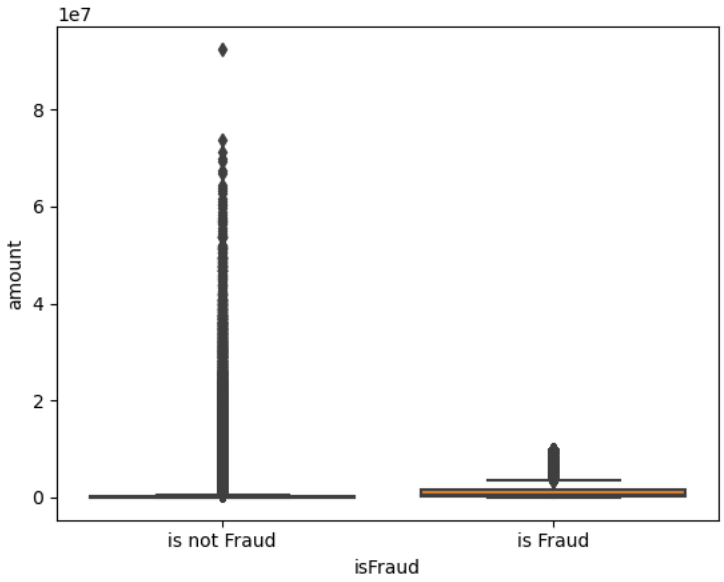
Here we are visualising the relationship between isFraud and step. Boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

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

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

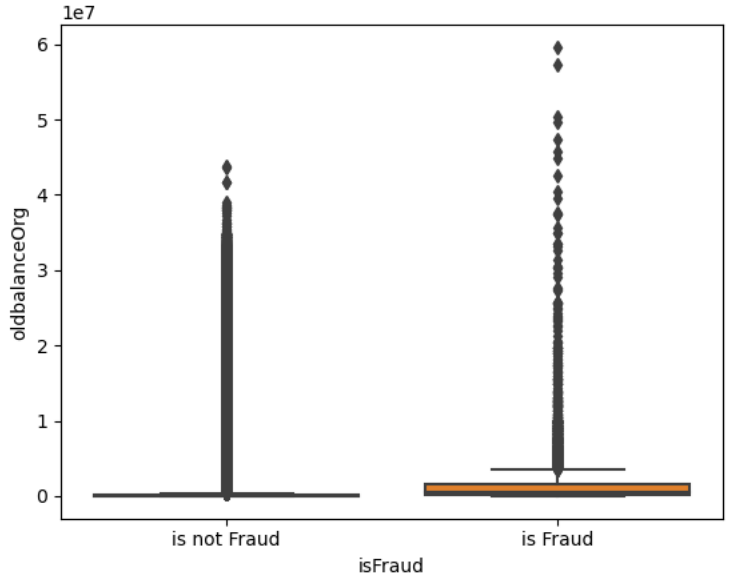
Here we are visualising the relationship between isFraud and amount. Boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```
: sns.boxplot(x='isFraud',y='amount',data=df)
: <Axes: xlabel='isFraud', ylabel='amount'>
```



Here we are visualising the relationship between isFraud and oldbalanceOrg. Boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

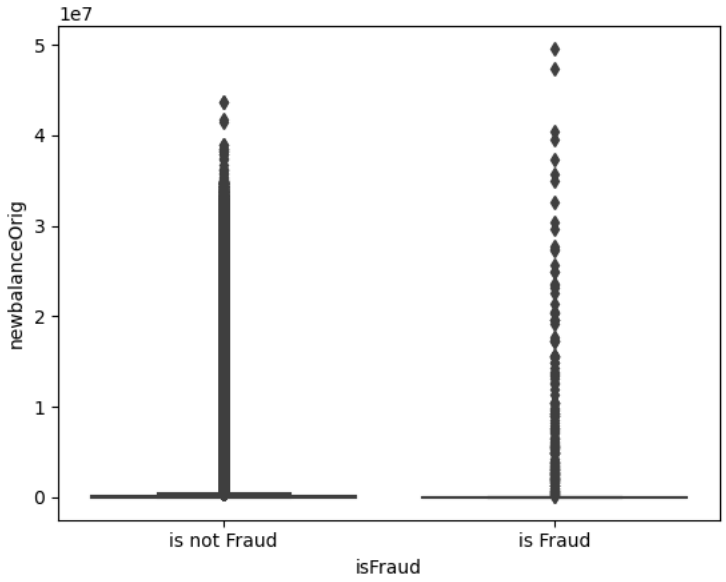
```
sns.boxplot(x='isFraud',y='oldbalanceOrg',data=df)
<Axes: xlabel='isFraud', ylabel='oldbalanceOrg'>
```



Here we are visualising the relationship between isFraud and newbalanceOrig. Boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

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

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



Here we are visualising the relationship between isFraud and oldbalanceDest. Violinplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

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

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

Here we are visualising the relationship between isFraud and newbalanceDest. Violinplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

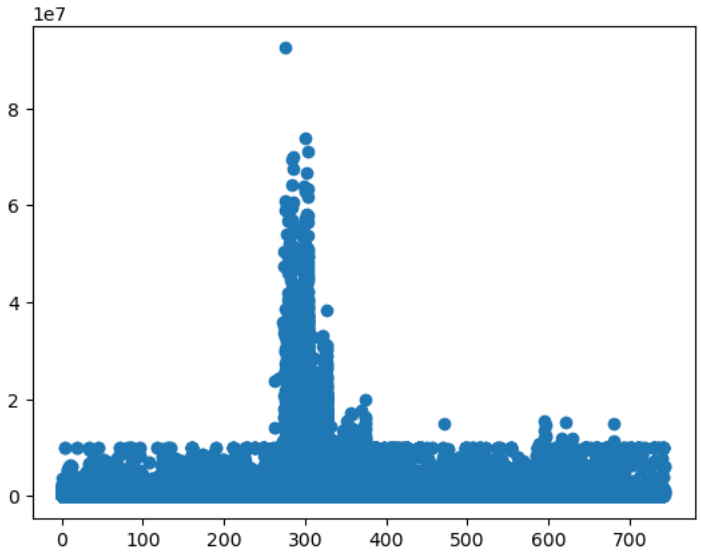
```
sns.violinplot(x='isFraud',y='newbalanceDest',data=df)
```

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

as

```
plt.scatter(df["step"],df["amount"])
```

```
<matplotlib.collections.PathCollection at 0x20827714ed0>
```



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	newbalanceDest	isFraud	isFlag
count	6.362620e+06	6362620	6.362620e+06	6362620	6.362620e+06	6.362620e+06	6362620	6.362620e+06	6.362620e+06	6362620	6.36
unique	NaN	5	NaN	6353307	NaN	NaN	2722362	NaN	NaN	2	
top	NaN	CASH_OUT	NaN	C1902386530	NaN	NaN	C1286084959	NaN	NaN	is not Fraud	
freq	NaN	2237500	NaN	3	NaN	NaN	113	NaN	NaN	6354407	
mean	2.433972e+02	NaN	1.798619e+05	NaN	8.338831e+05	8.551137e+05	NaN	1.100702e+06	1.224996e+06	NaN	2.51
std	1.423320e+02	NaN	6.038582e+05	NaN	2.888243e+06	2.924049e+06	NaN	3.399180e+06	3.674129e+06	NaN	1.58
min	1.000000e+00	NaN	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.00
25%	1.560000e+02	NaN	1.338957e+04	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.00
50%	2.390000e+02	NaN	7.487194e+04	NaN	1.420800e+04	0.000000e+00	NaN	1.327057e+05	2.146614e+05	NaN	0.00
75%	3.350000e+02	NaN	2.087215e+05	NaN	1.073152e+05	1.442584e+05	NaN	9.430367e+05	1.111909e+06	NaN	0.00
max	7.430000e+02	NaN	9.244552e+07	NaN	5.958504e+07	4.958504e+07	NaN	3.560159e+08	3.561793e+08	NaN	1.00

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
- Handling Object 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 a dataset
df.shape
```

```
(6362620, 11)
```

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', 'isFlaggedFraud'],
      dtype='object')
```

```
df.head()
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	170136.0	160296.36	0.0	0.0	is not Fraud	0
1	1	PAYMENT	1864.28	21249.0	19384.72	0.0	0.0	is not Fraud	0
2	1	TRANSFER	181.00	181.0	0.00	0.0	0.0	is Fraud	0
3	1	CASH_OUT	181.00	181.0	0.00	21182.0	0.0	is Fraud	0
4	1	PAYMENT	11668.14	41554.0	29885.86	0.0	0.0	is not Fraud	0

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

Activity 1: Checking for null values

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

```
#Checking for Null Values.
df.isnull().any()
```

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

```
df.isnull().sum()
```

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

```
df.nunique()
```

```
step               743
type                 5
amount         5316900
oldbalanceOrg  1845844
newbalanceOrig 2682586
oldbalanceDest 3614697
newbalanceDest 3555499
isFraud              2
isFlaggedFraud       2
dtype: int64
```

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()
```

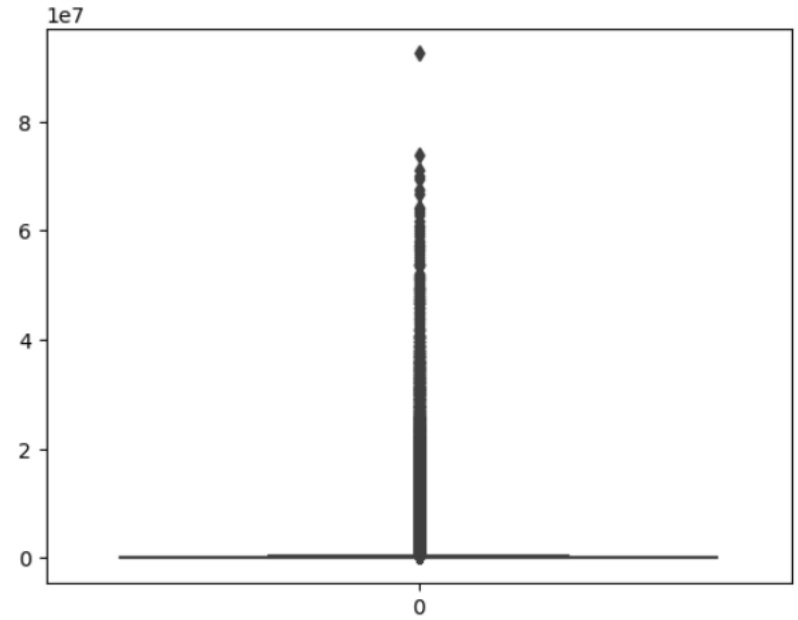
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 9 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            object 
 2   amount          float64
 3   oldbalanceOrg   float64
 4   newbalanceOrig  float64
 5   oldbalanceDest  float64
 6   newbalanceDest  float64
 7   isFraud         object 
 8   isFlaggedFraud  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 436.9+ MB
```

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

Activity 2: Handling outliers

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

```
<Axes: >
```



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

Remove Outliers:

Remove the Outliers

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\1242469671.py:2: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  print(stats.mode(df['amount']))
```

```
ModeResult(mode=array([10000000.]), count=array([3207]))
179861.90354913071
```

```
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)
print('Upper Bound :',upper_bound)
print('Lower Bound :',lower_bound)
print('Skewed data :',len (df[df['amount']>upper_bound]))
print('Skewed data :',len (df[df['amount']<lower_bound]))
```

```
q1:  13389.57
q3:  208721.4775
IQR:  195331.9075
Upper Bound : 501719.33875
Lower Bound : -279608.29125
Skewed data : 338078
Skewed data : 0
```

```
df['log_amount'] = np.log(df['amount']).replace([np.inf, -np.inf], np.nan)

df = df.dropna(subset=['log_amount'])
```

```
C:\Users\HP\anaconda3\Lib\site-packages\pandas\core\arraylike.py:402: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
#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)
    stats.probplot(feature, plot=plt)
```

```
transformationPlot(df['log_amount'])
```

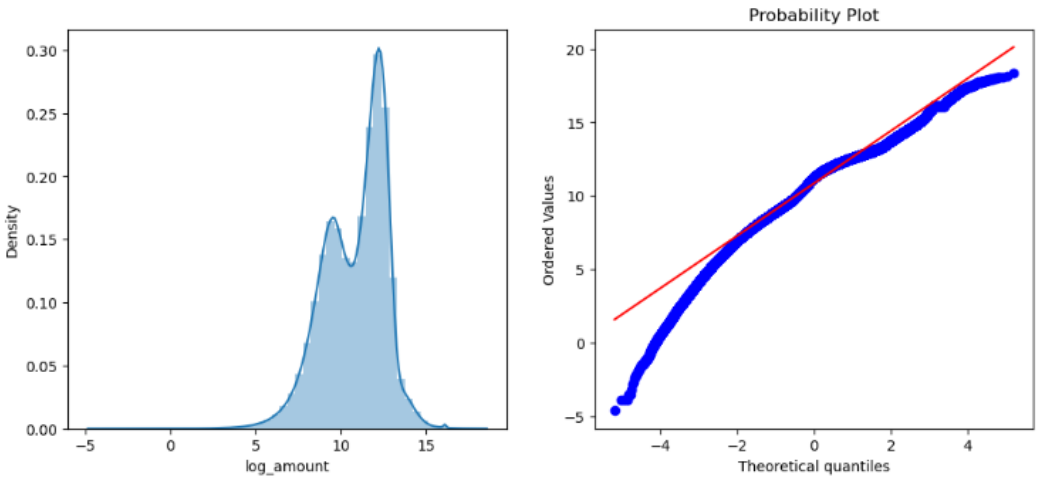
```
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\965733018.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(feature)
```



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

Activity 3: Object data labelencoding

Lablel encoding

```
from sklearn.preprocessing import LabelEncoder

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24496\3714687182.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['type']= la.fit_transform(df['type'])
```

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

```
1    2237484
3    2151495
0    1399284
4     532909
2      41432
Name: type, dtype: int64
```

using labelencoder to encode the dataset's object type

Splitting Dependent and Independent variables

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

```
x
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	log_amount
0	1	3	9.194174	170136.00	160296.36	0.00	0.00	is not Fraud	9.194174
1	1	3	7.530630	21249.00	19384.72	0.00	0.00	is not Fraud	7.530630
2	1	4	5.198497	181.00	0.00	0.00	0.00	is Fraud	5.198497
3	1	1	5.198497	181.00	0.00	21182.00	0.00	is Fraud	5.198497
4	1	3	9.364617	41554.00	29885.86	0.00	0.00	is not Fraud	9.364617
...	...	...	...	...	...	...	...	...	...
6362615	743	1	12.735766	339682.13	0.00	0.00	339682.13	is Fraud	12.735766
6362616	743	4	15.657870	6311409.28	0.00	0.00	0.00	is Fraud	15.657870
6362617	743	1	15.657870	6311409.28	0.00	68488.84	6379898.11	is Fraud	15.657870
6362618	743	4	13.652995	850002.52	0.00	0.00	0.00	is Fraud	13.652995
6362619	743	1	13.652995	850002.52	0.00	6510099.11	7360101.63	is Fraud	13.652995

6362604 rows × 9 columns

```
y
```

```
0          0
1          0
2          0
3          0
4          0
          ..
6362615    0
6362616    0
6362617    0
6362618    0
6362619    0
Name: isFlaggedFraud, Length: 6362604, dtype: int64
```

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.

Splitting data into train and test

```
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_train.shape)
print(y_test.shape)
```

```
(5090083, 9)
(1272521, 9)
(5090083,)
(1272521,)
```

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: Decision tree Classifier

A function named Decisiontree is 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.

1. Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

```
dtc.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred=dtc.predict(x_test)
```

```
pred
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
y_test
```

```
1302136    0
3150766    0
5346044    0
2346207    0
5001405    0
          ..
5165317    0
2551404    0
224722     0
495935     0
6018994    0
Name: isFlaggedFraud, Length: 1272521, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
```

```
ms.fit(x_train)
```

```
▾ MinMaxScaler
MinMaxScaler()
```

```
x_train_scaled = ms.transform(x_train)
x_train
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	log_amount
4320263	308	4	13.689327	0.00	0.00	895941.26	1777394.51	1	13.689327
816499	40	3	8.254165	0.00	0.00	0.00	0.00	1	8.254165
6052080	495	0	11.628403	29415.00	141655.94	464190.03	351949.09	1	11.628403
5737175	399	3	9.435016	290586.15	278066.98	0.00	0.00	1	9.435016
1214038	133	0	10.550267	2700201.96	2738389.61	44711.77	6524.12	1	10.550267
...	...	...	...	...	...	...	...	...	...
2249467	187	1	11.641247	4231.00	0.00	397758.64	511450.47	1	11.641247
5157702	357	1	12.021156	167.00	0.00	321544.24	487778.99	1	12.021156
2215104	186	0	11.287786	10152925.86	10232766.38	276175.87	196335.36	1	11.287786
1484405	141	0	13.250940	1081784.28	1650388.77	2631796.49	2063191.99	1	13.250940
4500018	325	3	10.878047	423.00	0.00	0.00	0.00	1	10.878047

5090083 rows × 9 columns

```
x_test_scaled = ms.transform(x_test)
x_test
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	log_amount
1302136	136	3	8.879953	0.00	0.00	0.00	0.00	1	8.879953
3150766	236	3	8.089430	5005.00	1745.17	0.00	0.00	1	8.089430
5346044	375	4	12.908264	0.00	0.00	899613.40	1303247.48	1	12.908264
2346207	189	3	8.370286	780857.00	776540.13	0.00	0.00	1	8.370286
5001405	353	0	12.459026	5530185.33	5787749.91	7099482.44	6841917.87	1	12.459026
...	...	...	...	...	...	...	...	...	...
5165317	358	0	11.319888	18621.00	101066.12	407712.72	325267.60	1	11.319888
2551404	206	1	12.217523	202509.00	205.83	2096979.23	2299282.40	1	12.217523
224722	14	4	14.644671	0.00	0.00	3729395.93	6020789.00	1	14.644671
495935	20	0	9.559148	11447430.20	11461603.97	3047711.43	3033537.66	1	9.559148
6018994	455	3	7.306766	0.00	0.00	0.00	0.00	1	7.306766

1272521 rows × 9 columns

```
# Fit the scaler on your training data
ms.fit(x_train)

# Transform your input data
input_data = [[136,3,8.879953,0.00,0.00,0.00,0.00,0,8.879953]]
transformed_data = ms.transform(input_data)

# Now you can use the transformed data for prediction
dtc.predict(transformed_data)
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but MinMaxScaler
was fitted with feature names
  warnings.warn(
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but DecisionTree
Classifier was fitted with feature names
  warnings.warn(
```

```
array([0], dtype=int64)
```

Predicting our data is Fraud or Not Fraud , 0 - Not Fraud, 1 – Fraud

Evaluation of classification model

```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
accuracy_score(y_test,pred)
```

```
1.0
```

```
confusion_matrix(y_test,pred)
```

```
array([[1272519,       0],
       [      0,       2]], dtype=int64)
```

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

col_0	0	1
isFlaggedFraud		
0	1272519	0
1	0	2

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272519
           1       1.00      1.00      1.00         2

    accuracy                           1.00   1272521
   macro avg       1.00      1.00      1.00   1272521
weighted avg       1.00      1.00      1.00   1272521
```

Roc-AUC curve

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

```
probability
```

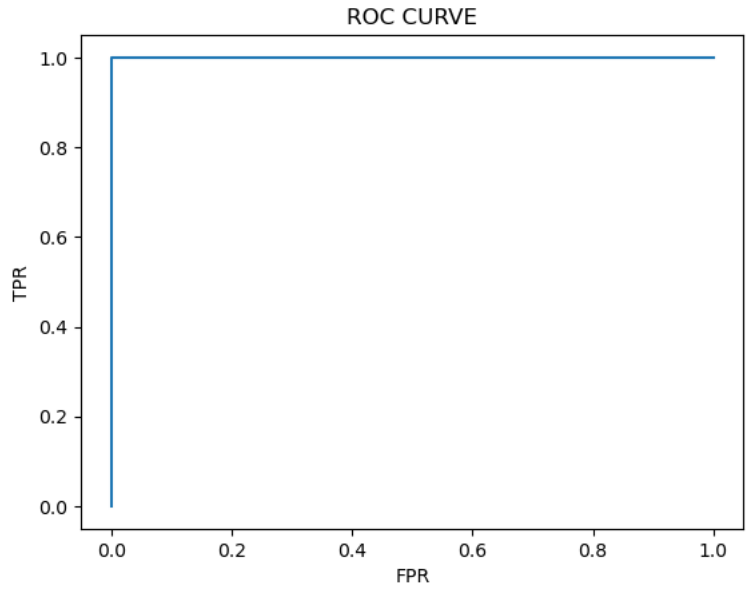
```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
print(x.columns)
```

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

```
# roc_curve
fpr,tpr,threshsholds = roc_curve(y_test,probability)
```

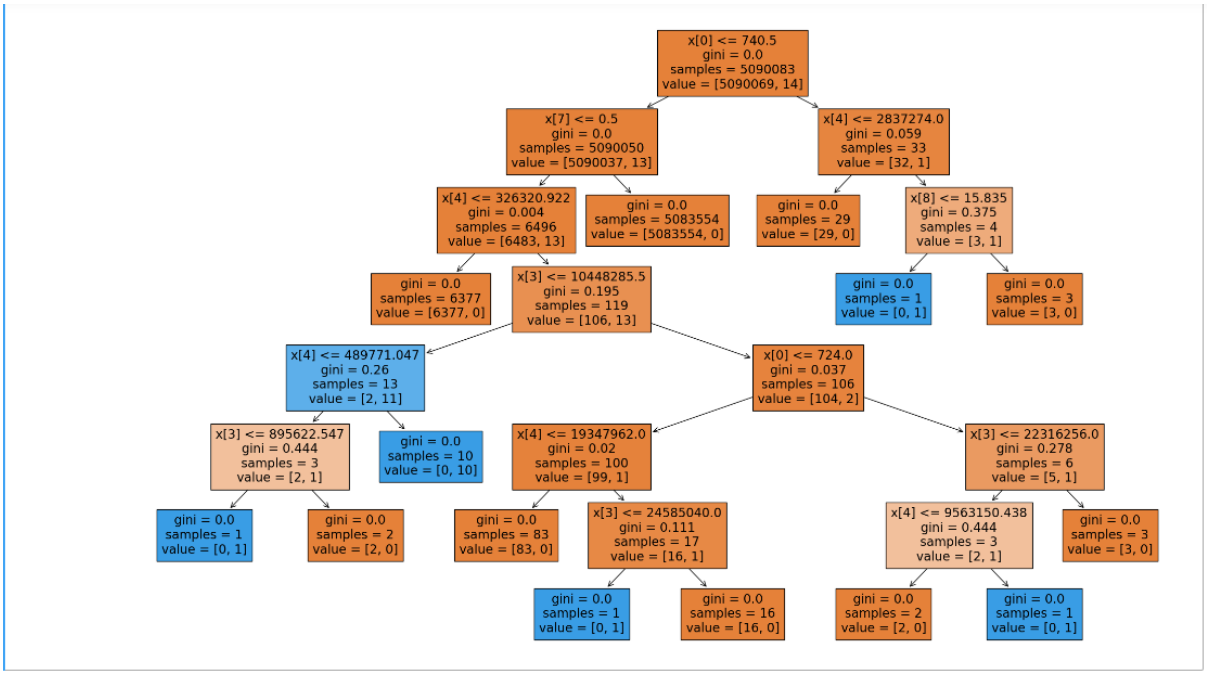
```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



The Decision Tree:

```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

```
[Text(0.5714285714285714, 0.9375, 'x[0] <= 740.5\ngini = 0.0\nsamples = 5090083\nvalue = [5090069, 14]'),
 Text(0.42857142857142855, 0.8125, 'x[7] <= 0.5\ngini = 0.0\nsamples = 5090050\nvalue = [5090037, 13]'),
 Text(0.35714285714285715, 0.6875, 'x[4] <= 326320.922\ngini = 0.004\nsamples = 6496\nvalue = [6483, 13]'),
 Text(0.2857142857142857, 0.5625, 'gini = 0.0\nsamples = 6377\nvalue = [6377, 0]'),
 Text(0.42857142857142855, 0.5625, 'x[3] <= 10448285.5\ngini = 0.195\nsamples = 119\nvalue = [106, 13]'),
 Text(0.21428571428571427, 0.4375, 'x[4] <= 489771.047\ngini = 0.26\nsamples = 13\nvalue = [2, 11]'),
 Text(0.14285714285714285, 0.3125, 'x[3] <= 895622.547\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.07142857142857142, 0.1875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.21428571428571427, 0.1875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.2857142857142857, 0.3125, 'gini = 0.0\nsamples = 10\nvalue = [0, 10]'),
 Text(0.6428571428571429, 0.4375, 'x[0] <= 724.0\ngini = 0.037\nsamples = 106\nvalue = [104, 2]'),
 Text(0.42857142857142855, 0.3125, 'x[4] <= 19347962.0\ngini = 0.02\nsamples = 100\nvalue = [99, 1]'),
 Text(0.35714285714285715, 0.1875, 'gini = 0.0\nsamples = 83\nvalue = [83, 0]'),
 Text(0.5, 0.1875, 'x[3] <= 24585040.0\ngini = 0.111\nsamples = 17\nvalue = [16, 1]'),
 Text(0.42857142857142855, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.5714285714285714, 0.0625, 'gini = 0.0\nsamples = 16\nvalue = [16, 0]'),
 Text(0.8571428571428571, 0.3125, 'x[3] <= 22316256.0\ngini = 0.278\nsamples = 6\nvalue = [5, 1]'),
 Text(0.7857142857142857, 0.1875, 'x[4] <= 9563150.438\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.7142857142857143, 0.0625, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.8571428571428571, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.9285714285714286, 0.1875, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.5, 0.6875, 'gini = 0.0\nsamples = 5083554\nvalue = [5083554, 0]'),
 Text(0.7142857142857143, 0.8125, 'x[4] <= 2837274.0\ngini = 0.059\nsamples = 33\nvalue = [32, 1]'),
 Text(0.6428571428571429, 0.6875, 'gini = 0.0\nsamples = 29\nvalue = [29, 0]'),
 Text(0.7857142857142857, 0.6875, 'x[8] <= 15.835\ngini = 0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(0.7142857142857143, 0.5625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.8571428571428571, 0.5625, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]')]
```




```
from sklearn.model_selection import GridSearchCV
parameter={
 'criterion':['gini','entropy'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']
}
```

```
grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")
```

```
grid_search.fit(x_train,y_train)
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
100 fits failed out of a total of 300.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
100 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py", line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py", line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\HP\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py", line 96, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of DecisionTreeClassifier must be an int in
the range [1, inf), a float in the range (0.0, 1.0], a str among {'sqrt', 'log2'} or None. Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:979: UserWarning: One or more of the test scores are
non-finite: [       nan        nan 0.99999725 0.99999725 0.99999725 0.99999725
        nan        nan 0.99999725 0.99999725 0.99999725 0.99999725
        nan        nan 0.99999725 0.99999725 0.99999725 0.99999725
        nan        nan 0.99999705 0.99999725 0.99999725 0.99999764
        nan        nan 0.99999823 0.99999745 0.99999705 0.99999725
        nan        nan 0.99999725 0.99999725 0.99999725 0.99999725
        nan        nan 0.99999725 0.99999725 0.99999725 0.99999725
        nan        nan 0.99999725 0.99999764 0.99999745 0.99999725
        nan        nan 0.99999764 0.99999725 0.99999764 0.99999725
        nan        nan 0.99999823 0.99999725 0.99999745 0.99999745]
  warnings.warn(
```

GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'sqrt',
 'splitter': 'best'}
```

```
dtc_cv=DecisionTreeClassifier(criterion= 'entropy',
 max_depth=3,
 max_features='sqrt',
 splitter='best')
dtc_cv.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')
```

```
pred=dtc_cv.predict(x_test)
```

```
print(classification_report(y_test,pred))
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272519
           1       0.00      0.00      0.00         2

    accuracy                           1.00   1272521
   macro avg       0.50      0.50      0.50   1272521
weighted avg       1.00      1.00      1.00   1272521
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

Decision Tree Classifiers

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_test_predict1=dtc.predict(x_test)
test_accuracy=accuracy_score (y_test,y_test_predict2)
test_accuracy
```

```
0.9999992141583518
```

```
y_train_predict1=dtc.predict(x_train)
train_accuracy=accuracy_score (y_train,y_train_predict1)
train_accuracy
```

```
1.0
```

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

col_0	0	1
isFlaggedFraud		
0	1272517	0
1	1	3

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272517
           1       1.00      0.75      0.86         4

    accuracy                           1.00   1272521
   macro avg       1.00      0.88      0.93   1272521
weighted avg       1.00      1.00      1.00   1272521
```

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.

2. Random Forest Classifier

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

```
▾ RandomForestClassifier
RandomForestClassifier()
```

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

```
1.0
```

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

```
1.0
```

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

col_0	0	1
isFlaggedFraud		
0	1272517	0
1	0	4

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272517
           1       1.00      1.00      1.00         4

    accuracy                           1.00   1272521
   macro avg       1.00      1.00      1.00   1272521
weighted avg       1.00      1.00      1.00   1272521
```

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.

3. ExtraTrees Classifier

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

```
ExtraTreesClassifier()
```

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

```
1.0
```

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

```
1.0
```

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

col_0	0	1
isFlaggedFraud		
0	1272519	0
1	0	2

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272519
           1       1.00      1.00      1.00         2

    accuracy                           1.00   1272521
   macro avg       1.00      1.00      1.00   1272521
weighted avg       1.00      1.00      1.00   1272521
```

Activity 4: SupportVectorMachine 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

4. SupportVectorMachine Classifier

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC()
svc.fit(x_train,y_train)
```

SVC()

```
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score (y_test,y_test_predict4)
test_accuracy
```

```
0.9999984283167036
```

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

```
0.999997249553691
```

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

col_0	0
isFlaggedFraud	
0	1272519
1	2

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

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272519
           1       0.00      0.00      0.00         2

    accuracy                           1.00   1272521
   macro avg       0.50      0.50      0.50   1272521
weighted avg       1.00      1.00      1.00   1272521

C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
df.columns
```

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

```
from sklearn.preprocessing import LabelEncoder

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

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

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_test
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
y_train
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

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

5. xgboost Classifier

```
from sklearn.model_selection import train_test_split
import xgboost as xgb
xgb1 = xgb.XGBClassifier()
xgb1.fit(x_train, y_train)
```

```
XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
```

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

```
0.9999992141583518
```

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

```
0.9999998035395493
```

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

col_0	0	1
isFlaggedFraud		
0	1272519	0
1	1	1

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1272519
           1       1.00      0.50      0.67         2

    accuracy                           1.00   1272521
   macro avg       1.00      0.75      0.83   1272521
weighted avg       1.00      1.00      1.00   1272521
```

Activity 6: Compare the model

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.

6. Compare the model

```
from sklearn.metrics import accuracy_score

def compareModel():
    print("train accuracy for dtc", accuracy_score(y_train_predict1, y_train))
    print("test accuracy for dtc", accuracy_score(y_test_predict1, y_test))
    print("train accuracy for rfc", accuracy_score(y_train_predict2, y_train))
    print("test accuracy for rfc", 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 svc", accuracy_score(y_test_predict4, y_test))
    print("train accuracy for xgb1", accuracy_score(y_train_predict5, y_train))
    print("test accuracy for xgb1", accuracy_score(y_test_predict5, y_test))
```

```
print(len(y_train), len(pred))
```

```
5090083 1272521
```

```
compareModel()
```

```
train accuracy for dtc 1.0
test accuracy for dtc 0.9999992141583518
train accuracy for rfc 1.0
test accuracy for rfc 1.0
train accuracy for etc 0.9999948920282832
test accuracy for etc 0.9999952849501108
train accuracy for svc 0.9999976424745922
test accuracy for svc 0.9999968566334072
train accuracy for xgb1 0.9999950884887339
test accuracy for xgb1 0.999996070791759
```

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().

7. Evaluating performance of the model and saving the model

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC ()
svc.fit(x_train,y_train)
```

```
SVC()
```

```
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score (y_test,y_test_predict4)
test_accuracy
```

```
0.9999984283167036
```

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

```
0.999997249553691
```

```
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

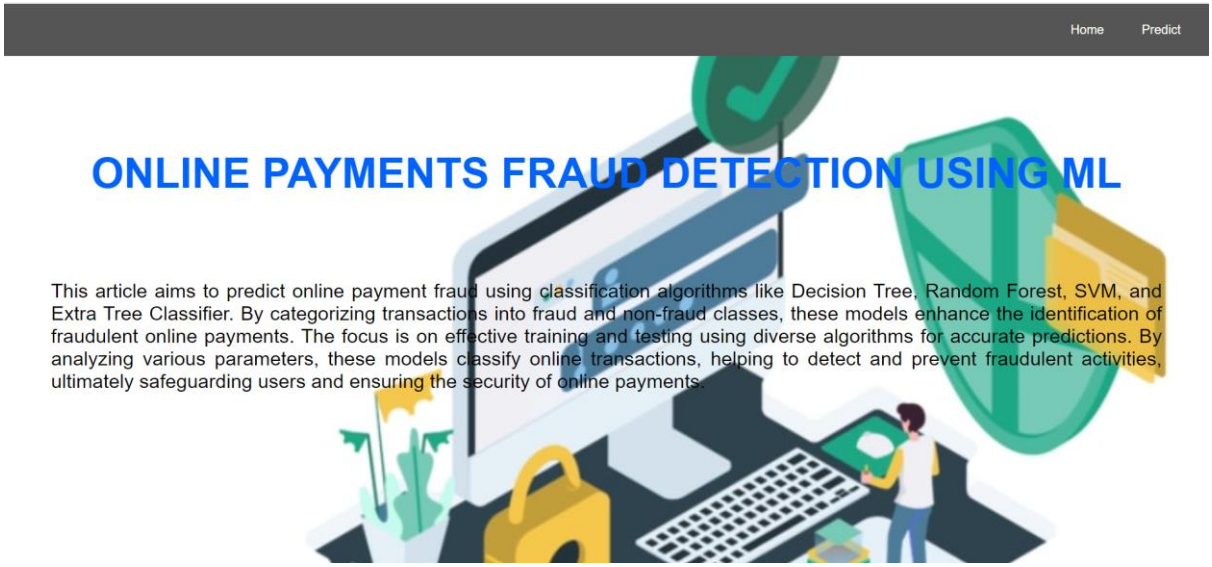
Activity1: Building Html Pages:

For this project create three HTML files namely

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

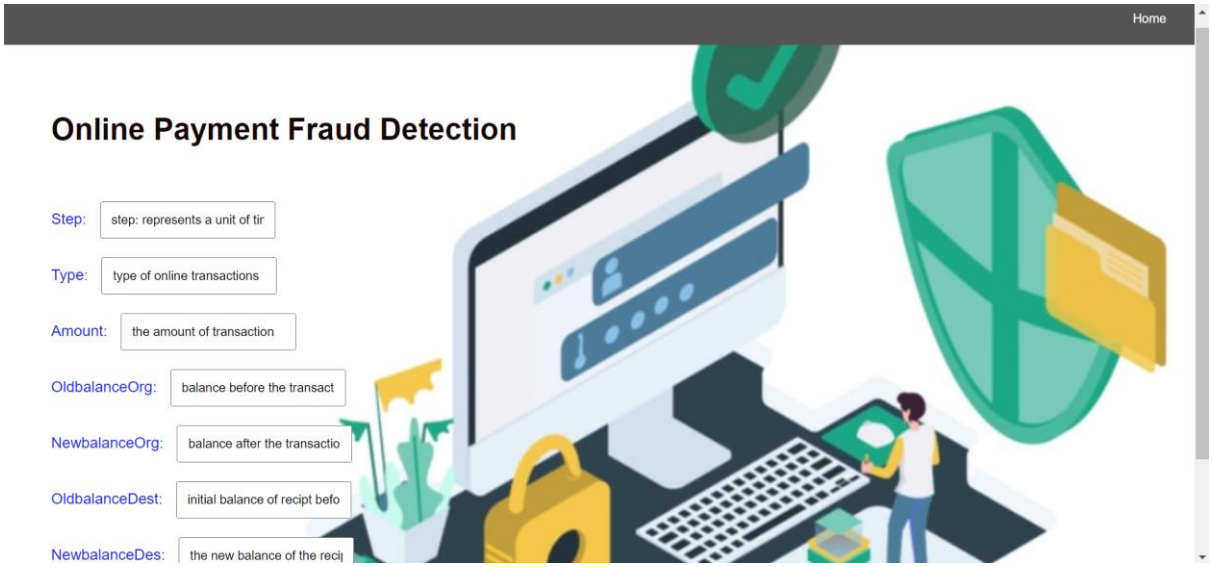
and save them in the templates folder.

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



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:



Now when you click on submit button at bottom in middle you will get redirected to submit.html

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

Activity 2: Build Python code:

Import the libraries

```
app_ibm.py | app.py | home.html | predict.html | submit.html

1  from flask import Flask, render_template, request
2  import numpy as np
3  import pickle
4  import pandas as pd
5
6  model = pickle.load(open('D:/Project/online payments fraud detection/Flask/p
7
```

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('D:/Project/online payments fraud detection/Flask/payments.pkl', 'rb'))

app = Flask (___name___)
```

Render HTML page:

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

@app.route("/home")
def about1():
    return render_template('home.html')
```

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)
    pred = model.predict(x)
    print(pred[0])
    return render_template('submit.html', prediction_text=str(pred))
```

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)
```

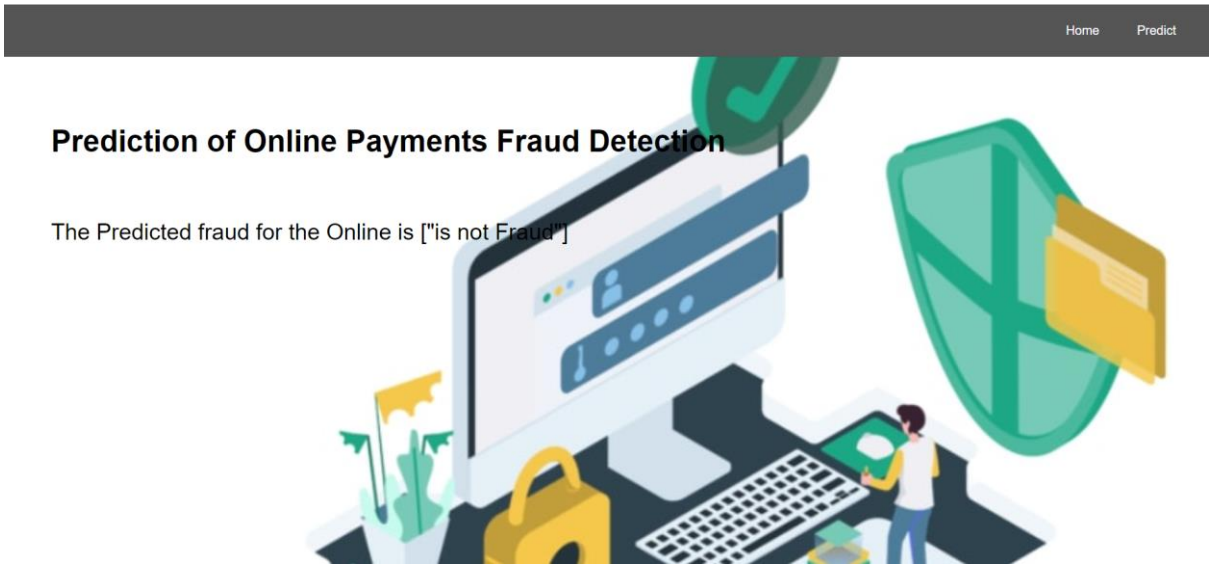
Activity 3: Run the application

- 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, lick on the submit button, and see the result/prediction on the web.

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  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 Payment Fraud Detection

Step: 76

Type: 378

Amount: 1373.43

OldbalanceOrg: 13854.00

NewbalanceOrg: 4

OldbalanceDest: 45498.0

NewbalanceDes: 0.00

Submit

Prediction of Online Payments Fraud Detection

The Predicted fraud for the Online is ["isFraud"]

Online Payment Fraud Detection

Step: 1

Type: 2

Amount: 5337.77

OldbalanceOrg: 41720.00

NewbalanceOrg: 36382.23

OldbalanceDest: 41898.79

NewbalanceDes: 40348.79

Submit

Home Predict

Prediction of Online Payments Fraud Detection

The Predicted fraud for the Online is ["is not Fraud"]

Online Payment Fraud Detection

Step: 18579

Type: 3

Amount: 87657

OldbalanceOrg: 41720.00

NewbalanceOrg: 790.007

OldbalanceDest: 41898.79

NewbalanceDes: 54387.079

Submit

Prediction of Online Payments Fraud Detection

The Predicted fraud for the Online is ["isFraud"]