ONLINE PAYMENTS FRAUD DETECTION USING MACHINE LEARNING

A UG PROJECT PHASE-2 REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

UPPULA DIVYA	**19UK1A05F5**
VEMUNOORI RAMANA	**19UK1A05F4**
DEVA NAGESH	**19UK1A05K0**
VEMURU JAGADEESHWARI	**19UK1A05G3**

Under the esteemed guidance of

Mr. G. RAMESH

(Associative Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
(Affiliated to JNTUH, Hyderabad)
Bollikunta, Warangal – 506005

2019–2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
BOLLIKUNTA, WARANGAL – 506005
2019 – 2023

CERTIFICATE OF COMPLETION

UG PROJECT PHASE-2

This is to certify that the UG Project Phase-2 entitled **" ONLINE PAYMENTS FRAUD DETECTION USING MACHINE LEARNING"** is being submitted by ***UPPULA DIVYA(19UK1A05F5), VEMUNOORI RAMANA(19UK1A05F4)*** **, DEVA NAGESH(19UK1A05K0), VEMURU JAGADEESHWARI(19UK1A05G3)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** during the academic year **2022-23**, is a record of work carried out by them under the guidance and supervision.

Project Guide
Mr. G. RAMESH
(Associate Professor)

Head of the Department
Dr. R. Naveen Kumar
(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-2 in the institute.

We extend our heartfelt thanks to **Dr. R. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-2.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-2 and for their support in completing the UG Project Phase-2.

We express heartfelt thanks to the guide, **Mr. G. RAMESH,** Associate Professor, Department of CSE for her constant support and giving necessary guidance for completion of this UG Project Phase-2.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experiencing through this.

U. DIVYA	**19UK1A05F5**
V.RAMANA	**19UK1A05F4**
D.NAGESH	**19UK1A05K0**
V.JAGADEESHWARI	**19UK1A05G3**

TABLE OF CONTENTS:

LIST OF FIGURES PAGE NO

1. INTRODUCTION

In today's world, we are on the verge to become a cashless world. According to various surveys and researches, people performing online transactions has increased a lot, it's expected that in future years this will go on increasing. Now, while this might be exciting news, on the other-side fraudulent transactions are on the rise as well. Even due to various security systems being implemented, we still have a very high amount of money being lost due to fraudulent transactions. Online Fraud Transaction can be defined as a case where a person uses someone else's credit card for personal reasons while the owner and the card-issuing authorities are unaware of the fact that the card is being used. Fraud detection involves monitoring the activities of populations of users to estimate, perceive or avoid objectionable behavior, which consists of fraud, intrusion, and defaulting. Most of the time, a person who has become a victim of such fraud doesn't have any idea about it until the very end.

UG Project Phase-2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase-1. All the implementation is done and conclusions are retrieved in this phase. We will also work on the applications, advantages, and disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase-2.

2. CODE SNIPPETS
2.1 LOADING THE DATASET:

Activity 1: Importing Libraries

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

Figure 1 : importing required libraries

Activity 2 : Reading the csv data

```
# Reading the csv data
df = pd.read_csv(r'/content/PS_20174392719_1491204439457_logs.csv')
```

Python

```
df
```

Python

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlag
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	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.00	0	
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.00	0	
...	...	...	...	...	...	...	...	...	...	...	...
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.02	1	
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.00	1	
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.59	1	
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.00	C757947873	0.00	0.00	1	
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.00	C1827219533	0.00	87999.25	1	

2430 rows × 11 columns

```
df.columns
```

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

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

```
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	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.00	0
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.00	0
...	...	...	...	...	...	...	...	...	...	...
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.02	1
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.00	1
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.59	1
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.00	C757947873	0.00	0.00	1
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.00	C1827219533	0.00	87999.25	1

2430 rows × 10 columns

Figure 2 : Reading the csv data

About the data set

The below column reference:

1.step: represents a unit of time where 1 step equals 1 hour

+ Code + Markdown

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

Figure 3 : About the dataset

```
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	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0
3	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0	0
4	1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M408069119	0.0	0.0	0

```
df.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.0	C79051264	51433.88	108179.02	1
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.0	C1140210295	0.00	0.00	1
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.0	C1759363094	0.00	33676.59	1
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.0	C757947873	0.00	0.00	1
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.0	C1827219533	0.00	87999.25	1

+ Code + Markdown

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

```
df.columns
```

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

Checking for Co relation

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

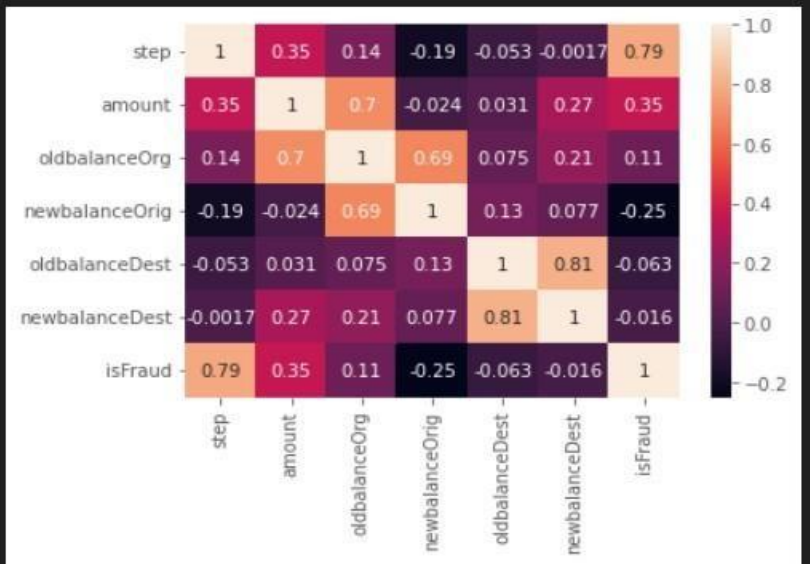
	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	0.352348	0.139868	-0.194391	-0.053366	-0.001745	0.788370
amount	0.352348	1.000000	0.703566	-0.023694	0.030711	0.274788	0.354960
oldbalanceOrg	0.139868	0.703566	1.000000	0.685439	0.075271	0.212087	0.105713
newbalanceOrig	-0.194391	-0.023694	0.685439	1.000000	0.127352	0.077034	-0.250987
oldbalanceDest	-0.053366	0.030711	0.075271	0.127352	1.000000	0.811400	-0.063175
newbalanceDest	-0.001745	0.274788	0.212087	0.077034	0.811400	1.000000	-0.015916
isFraud	0.788370	0.354960	0.105713	-0.250987	-0.063175	-0.015916	1.000000

Figure 4 :checking correlation of dataset using d

Heat map

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

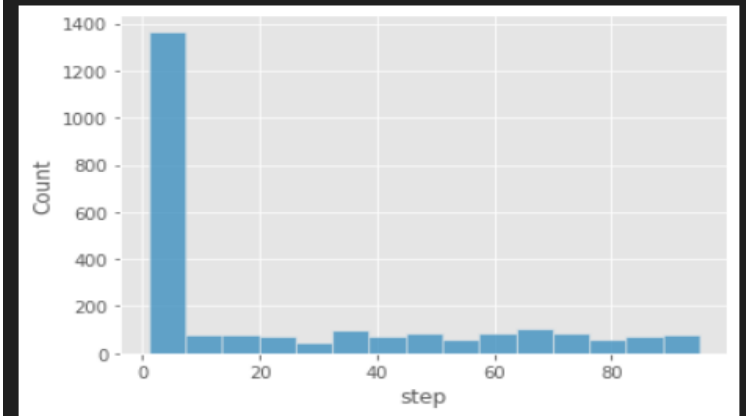
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc7914067f0>
```



Activity 3 : Univariate Analysis

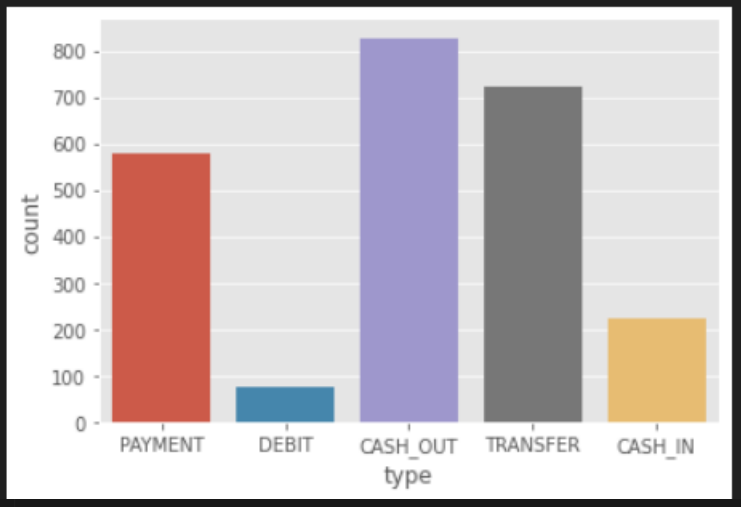
```
#step
sns.histplot(data=df,x='step')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc791400dc0>
```



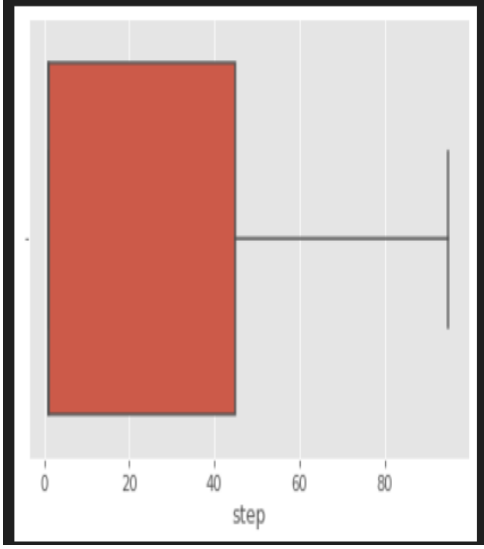
```
#type
sns.countplot(data=df,x='type')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e427b20>
```

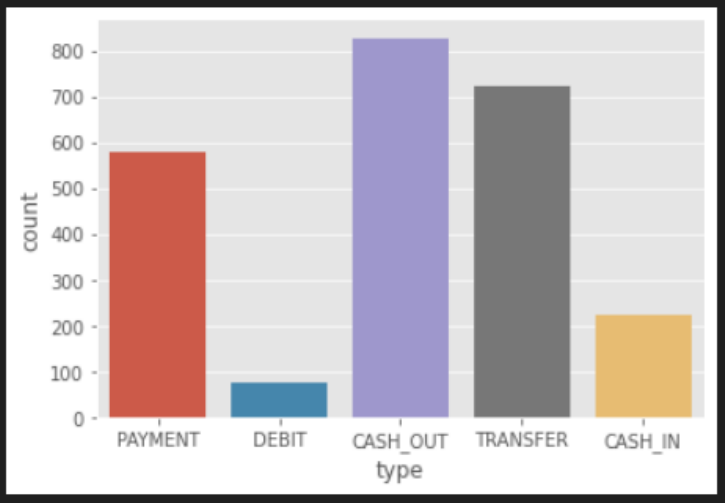



```
sns.boxplot(data=df,x='step')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc7914063a0>
```

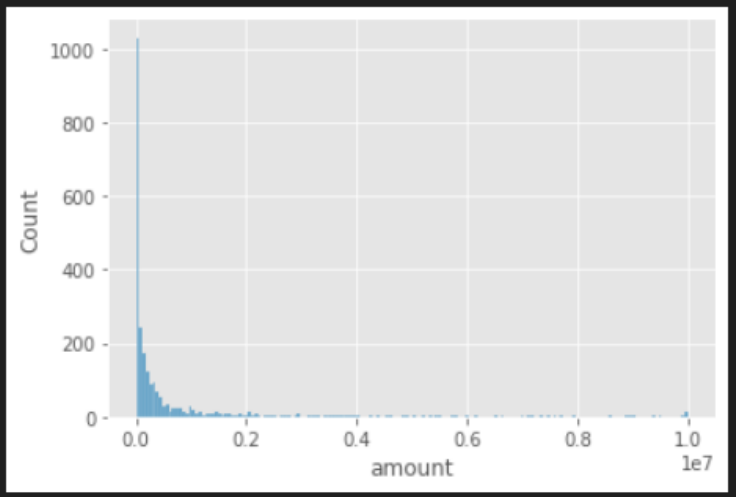


```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e427b20>
```

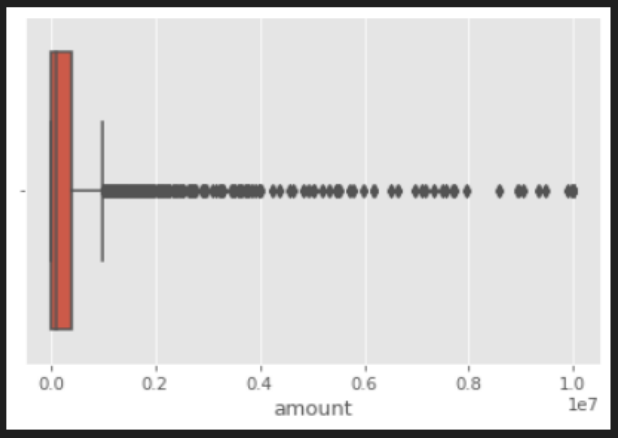



```
#amount
sns.histplot(data=df,x='amount')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e4df910>
```

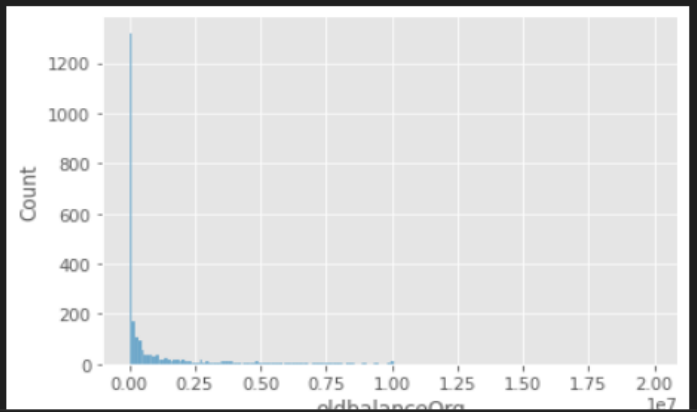


```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e3d6310>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e196a30>
```



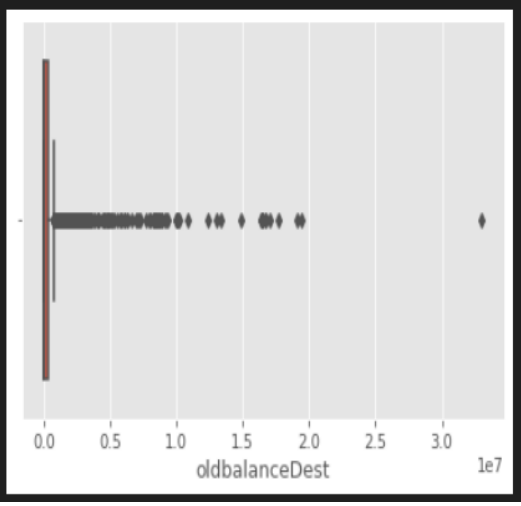

```
#nameDest
df['nameDest'].value_counts()
```

```
C1590550415    25
C985934102     22
C564160838     19
C451111351     17
C1023714065    15
               ..
M1113829504     1
M936219350      1
M178401052      1
M1888639813     1
C757947873      1
Name: nameDest, Length: 1870, dtype: int64
```

FIGURE

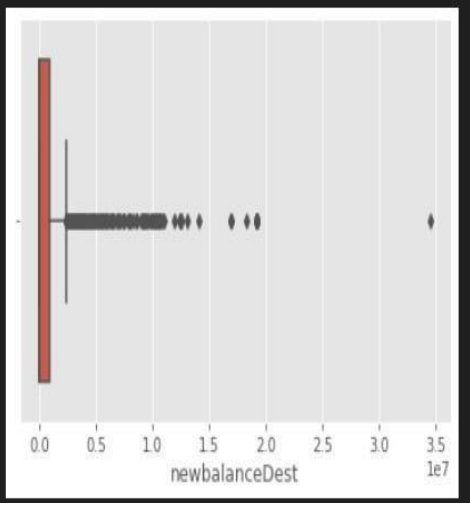
```
#oldbalanceDest
sns.boxplot(data=df,x='oldbalanceDest')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e4305b0>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78dec2c40>
```



FIGURE

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78e430a90>
```


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

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

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

Python

```
df
```

Python

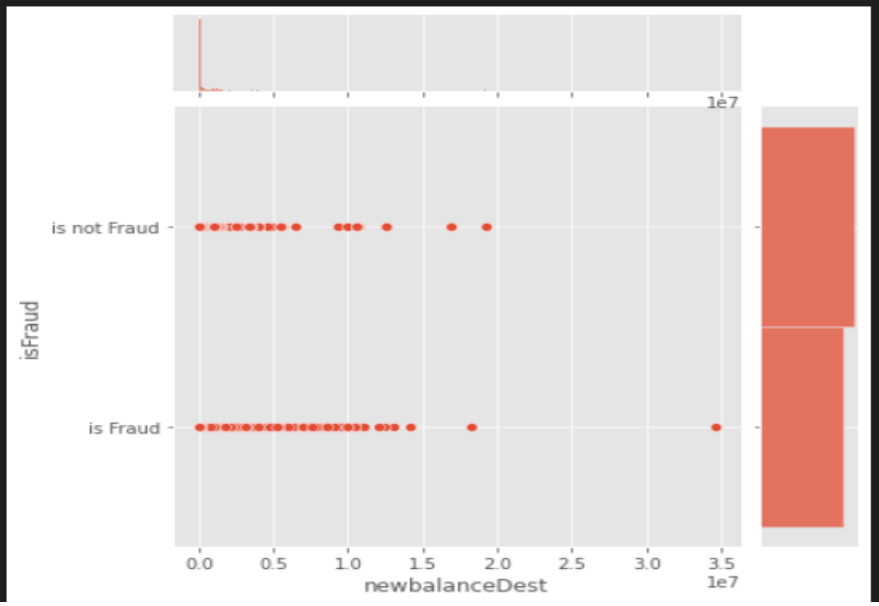
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	is not Fraud
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	is not Fraud
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	is not Fraud
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.00	is not Fraud
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.00	is not Fraud
...	...	...	...	...	...	...	...	...	...	...
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.02	is Fraud
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.00	is Fraud
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.59	is Fraud
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.00	C757947873	0.00	0.00	is Fraud
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.00	C1827219533	0.00	87999.25	is Fraud

2430 rows × 10 columns

ACTIVITY 4 :

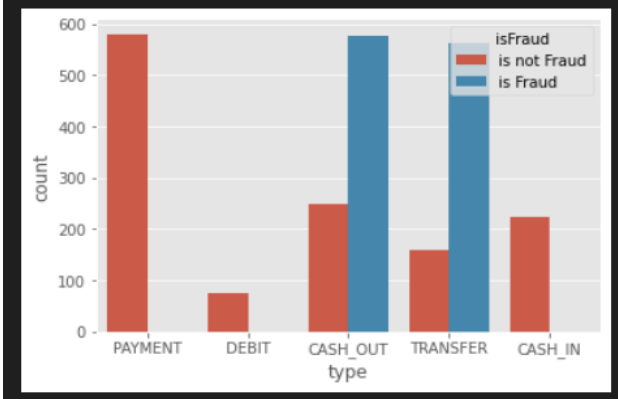
```
sns.jointplot(data=df,x='newbalanceDest',y='isFraud')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc7912685e0>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78dad0be0>
```



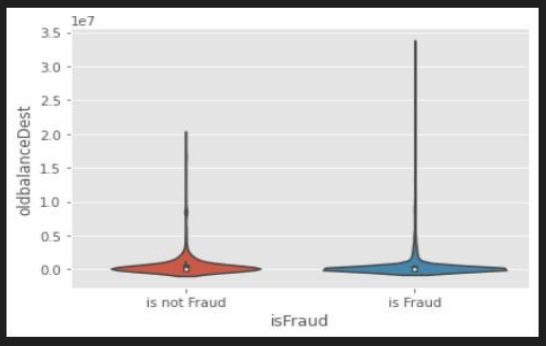

```
sns.boxplot(data=df,x='isFraud',y='step')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78da5d5e0>
```

FIGURE

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

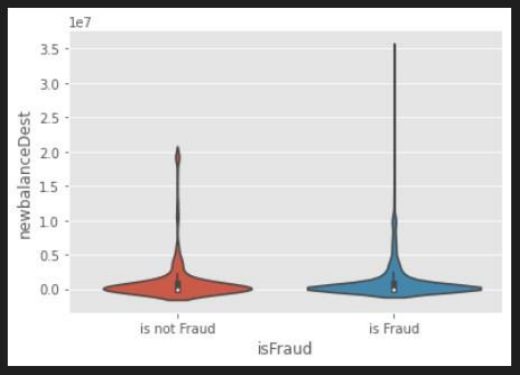
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78d896d60>
```



FIGURE

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78b803910>
```



+ Code + Markdown

Activity 5: Descriptive analysis

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

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
count	2430.000000	2430	2.430000e+03	2430	2.430000e+03	2.430000e+03	2430	2.430000e+03	2.430000e+03	2430
unique	NaN	5	NaN	2430	NaN	NaN	1870	NaN	NaN	2
top	NaN	CASH_OUT	NaN	C1231006815	NaN	NaN	C1590550415	NaN	NaN	is not Fraud
freq	NaN	827	NaN	1	NaN	NaN	25	NaN	NaN	1288
mean	23.216049	NaN	6.258361e+05	NaN	9.849040e+05	4.392755e+05	NaN	5.797246e+05	1.127075e+06	NaN
std	29.933036	NaN	1.503866e+06	NaN	2.082361e+06	1.520978e+06	NaN	1.891192e+06	2.907401e+06	NaN
min	1.000000	NaN	8.730000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
25%	1.000000	NaN	9.018493e+03	NaN	8.679630e+03	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
50%	1.000000	NaN	1.058692e+05	NaN	8.096250e+04	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
75%	45.000000	NaN	4.096098e+05	NaN	7.606258e+05	1.247804e+04	NaN	3.096195e+05	9.658701e+05	NaN
max	95.000000	NaN	1.000000e+07	NaN	1.990000e+07	9.987287e+06	NaN	3.300000e+07	3.460000e+07	NaN

2.3 DATA PREPROCESSING

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

```
(2430, 10)
```

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

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

```
df.head()
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	170136.0	160296.36	0.0	0.0	is not Fraud
1	1	PAYMENT	1864.28	21249.0	19384.72	0.0	0.0	is not Fraud
2	1	PAYMENT	11668.14	41554.0	29885.86	0.0	0.0	is not Fraud
3	1	PAYMENT	7817.71	53860.0	46042.29	0.0	0.0	is not Fraud
4	1	PAYMENT	7107.77	183195.0	176087.23	0.0	0.0	is not Fraud

FIGURE

Activity1: Checking null values

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

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

```
df.info()
```

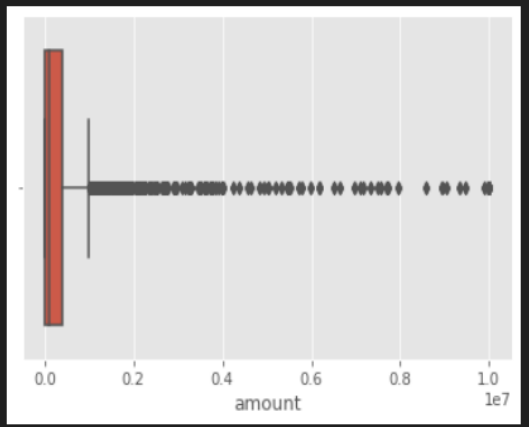
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2430 entries, 0 to 2429
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   step            2430 non-null   int64
 1   type            2430 non-null   object
 2   amount          2430 non-null   float64
 3   oldbalanceOrg   2430 non-null   float64
 4   newbalanceOrig  2430 non-null   float64
 5   oldbalanceDest  2430 non-null   float64
 6   newbalanceDest  2430 non-null   float64
 7   isFraud         2430 non-null   object
dtypes: float64(5), int64(1), object(2)
memory usage: 152.0+ KB
```

FIGURE

Activity 2: Handling Outliers

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc78daedf70>
```



FIGURE

Remove the outliers:

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

```
ModeResult(mode=array([10000000.]), count=array([14]))
625836.0974156379
```

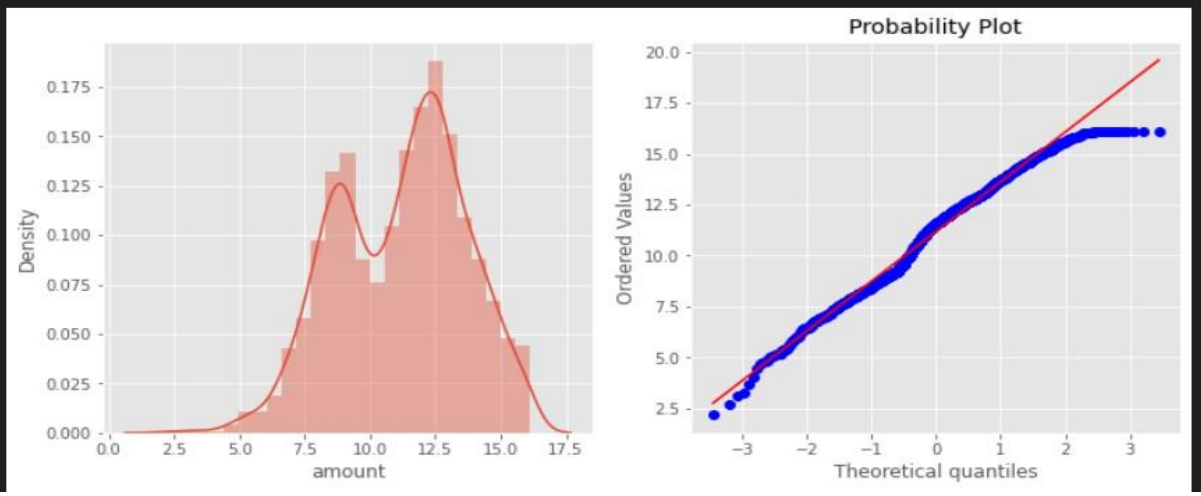
```
q1 = np.quantile(df['amount'],0.25)
q3 = np.quantile(df['amount'],0.75)

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 : 9018.4925
q3 : 409609.8225
IQR : 400591.33
Upper Bound : 1010496.8175
Lower Bound : -591868.5025
Skewed data : 354
Skewed data : 0
```

Figure

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



Fig

Activity 3:OBJECT DATA ENCODING

```
from sklearn.preprocessing import LabelEncoder

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

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

```
1    827
4    724
3    580
0    224
2     75
Name: type, dtype: int64
```

FIGURE

Divding data into dependent and independent

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

```
x
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	1	3	9.194174	170136.00	160296.36	0.00	0.00
1	1	3	7.530630	21249.00	19384.72	0.00	0.00
2	1	3	9.364617	41554.00	29885.86	0.00	0.00
3	1	3	8.964147	53860.00	46042.29	0.00	0.00
4	1	3	8.868944	183195.00	176087.23	0.00	0.00
...	...	...	...	...	...	...	...
2425	95	1	10.946325	56745.14	0.00	51433.88	108179.02
2426	95	4	10.424558	33676.59	0.00	0.00	0.00
2427	95	1	10.424558	33676.59	0.00	0.00	33676.59
2428	95	4	11.385084	87999.25	0.00	0.00	0.00
2429	95	1	11.385084	87999.25	0.00	0.00	87999.25

2430 rows × 7 columns

Figure

```
y
```

```
0       is not Fraud
1       is not Fraud
2       is not Fraud
3       is not Fraud
4       is not Fraud
           ...
2425       is Fraud
2426       is Fraud
2427       is Fraud
2428       is Fraud
2429       is Fraud
Name: isFraud, Length: 2430, dtype: object
```

Activity 4: 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_test.shape)
print(y_train.shape)
```

```
(1944, 7)
(486, 7)
(486,)
(1944,)
```

MODEL BULDING

Activity 1:Random forest classifier1

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

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

```
0.9958847736625515
```

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

```
              precision    recall  f1-score   support

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

    accuracy                           1.00       486
   macro avg       1.00      1.00      1.00       486
weighted avg       1.00      1.00      1.00       486
```

Activity 2: Decision tree classifier

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

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

```
0.9917695473251029
```

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

```
1.0
```

figure

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

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

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

```
              precision    recall  f1-score   support

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

    accuracy                           0.99       486
   macro avg       0.99      0.99      0.99       486
weighted avg       0.99      0.99      0.99       486
```

figure

Activity 3

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

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

```
0.9938271604938271
```

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

```
              precision    recall  f1-score   support

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

    accuracy                           0.99       486
   macro avg       0.99      0.99      0.99       486
weighted avg       0.99      0.99      0.99       486
```

Figure

Activity 4:

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

```
0.7901234567901234
```

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

```
0.8009259259259259
```

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

```
              precision    recall  f1-score   support

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

    accuracy                           0.79       486
   macro avg       0.86      0.78      0.78       486
weighted avg       0.85      0.79      0.78       486
```

```
from sklearn.preprocessing import LabelEncoder

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

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

```
y_test1
```

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

Figure

```
y_train1
```

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

figu

Activity 5:

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

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

```
0.9958847736625515
```

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

```
1.0
```

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

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       234
           1       0.99      1.00      1.00       252

    accuracy                           1.00       486
   macro avg       1.00      1.00      1.00       486
weighted avg       1.00      1.00      1.00       486
```

Comparing The Models:

figure

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

```
(function) def accuracy_score() -> Any
```

```
compareModel()
```

```
train accuracy for rfc 1.0
test accuracy for rfc 0.9958847736625515
train accuracy for dtc 1.0
test accuracy for dtc 0.9917695473251029
train accuracy for etc 1.0
test accuracy for etc 0.9938271604938271
train accuracy for svc 0.8009259259259259
test accuracy for svcc 0.7901234567901234
train accuracy for xgb1 1.0
test accuracy for xgb1 0.9958847736625515
```

figure

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

figure

```
df.head(20)
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	3	9.194174	170136.00	160296.36	0.0	0.00	is not Fraud
1	1	3	7.530630	21249.00	19384.72	0.0	0.00	is not Fraud
2	1	3	9.364617	41554.00	29885.86	0.0	0.00	is not Fraud
3	1	3	8.964147	53860.00	46042.29	0.0	0.00	is not Fraud
4	1	3	8.868944	183195.00	176087.23	0.0	0.00	is not Fraud
5	1	3	8.969751	176087.23	168225.59	0.0	0.00	is not Fraud
6	1	3	8.300121	2671.00	0.00	0.0	0.00	is not Fraud
7	1	2	8.582563	41720.00	36382.23	41898.0	40348.79	is not Fraud
8	1	2	9.174189	4465.00	0.00	10845.0	157982.12	is not Fraud
9	1	3	8.039148	20771.00	17671.03	0.0	0.00	is not Fraud
10	1	3	7.848052	5070.00	2509.26	0.0	0.00	is not Fraud
11	1	3	9.361666	10127.00	0.00	0.0	0.00	is not Fraud
12	1	3	8.318445	503264.00	499165.22	0.0	0.00	is not Fraud
13	1	1	12.342062	15325.00	0.00	5083.0	51513.44	is not Fraud
14	1	3	7.354887	450.00	0.00	0.0	0.00	is not Fraud
15	1	3	7.054329	21156.00	19998.14	0.0	0.00	is not Fraud
16	1	3	6.509722	15123.00	14451.36	0.0	0.00	is not Fraud
17	1	4	12.279836	705.00	0.00	22425.0	0.00	is not Fraud
18	1	3	7.225067	13854.00	12480.57	0.0	0.00	is not Fraud
19	1	2	9.138070	11299.00	1996.21	29832.0	16896.70	is not Fraud

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

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
2410	94	4	14.590090	2169679.91	0.0	0.00	0.00	is Fraud
2411	94	1	14.590090	2169679.91	0.0	0.00	2169679.91	is Fraud
2412	94	4	14.190236	1454592.61	0.0	0.00	0.00	is Fraud
2413	94	1	14.190236	1454592.61	0.0	264042.92	1718635.53	is Fraud
2414	94	4	13.040363	460635.82	0.0	0.00	0.00	is Fraud
2415	94	1	13.040363	460635.82	0.0	544728.69	1005364.51	is Fraud
2416	94	4	14.688284	2393539.65	0.0	0.00	0.00	is Fraud
2417	94	1	14.688284	2393539.65	0.0	5157128.07	7550667.73	is Fraud
2418	94	4	13.006408	445257.43	0.0	0.00	0.00	is Fraud
2419	94	1	13.006408	445257.43	0.0	0.00	445257.43	is Fraud
2420	94	4	12.803201	363378.75	0.0	0.00	0.00	is Fraud
2421	94	1	12.803201	363378.75	0.0	3609871.44	3973250.18	is Fraud
2422	95	4	13.393424	655676.97	0.0	0.00	0.00	is Fraud
2423	95	1	13.393424	655676.97	0.0	53614.28	709291.25	is Fraud
2424	95	4	10.946325	56745.14	0.0	0.00	0.00	is Fraud
2425	95	1	10.946325	56745.14	0.0	51433.88	108179.02	is Fraud
2426	95	4	10.424558	33676.59	0.0	0.00	0.00	is Fraud
2427	95	1	10.424558	33676.59	0.0	0.00	33676.59	is Fraud
2428	95	4	11.385084	87999.25	0.0	0.00	0.00	is Fraud
2429	95	1	11.385084	87999.25	0.0	0.00	87999.25	is Fraud

```
# prediction
#features = [step,type,amount,oldbalanceOrg,newbalanceOrig,oldbalanceDest,newbalanceDest]
features = np.array([[1,3,9.194174,170136.00,160296.36,0.0,0.00]])
print(svc.predict(features))
```

```
['is not Fraud']
```

```
# prediction
#features = [step,type,amount,oldbalanceOrg,newbalanceOrig,oldbalanceDest,newbalanceDest]
features = np.array([[94,4,14.590090,2169679.91,0.0,0.00,0.00]])
print(svc.predict(features))
```

```
['is Fraud']
```

```
# prediction
#features = [step,type,amount,oldbalanceOrg,newbalanceOrig,oldbalanceDest,newbalanceDest]
features = np.array([[1,2,9.138070,11299.00,1996.21,29832.0,16896.70]])
print(svc.predict(features))
```

```
['is not Fraud']
```

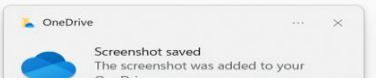
```
# prediction
#features = [step,type,amount,oldbalanceOrg,newbalanceOrig,oldbalanceDest,newbalanceDest]
features = np.array([[94,1,14.190236,1454592.61,0.0,264042.92,1718635.53]])
print(svc.predict(features))
```

```
['is Fraud']
```

2.1. APPLICATION BUILDING:

HTML Code:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Home</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <style>
        body
        {
        background-image: url("
sCR8rX82yGt0BMzosjTMhOixjlcg6eWRYIgHq1Sx5ML/o1zpEkWHJy8bzsUxF4wV7FRWWcsqHpaoM3iMCfeWiX+wQRYQl9k3SVdOseu9PpSIayiPCEvmh1K+AdEX1sKzgSMKMCEsfs1QEyitXTMXr59GMPhFh
i3jdFza1mxv+3U/xwZOlSBFjKvV4wZDzKsPBTVdws1b49aCgzjNe0/OAv/7MsVaXqi5LH8G5b2GLpct0zlnGgbJuFfxFCEWCJ+hYMAky/IdNU7F27NssKOW9vwxxZRRQNlv1zuj4wTfcUhSgp2TkR0UvJkvYC
L2PqD6Z6GEqoqmhbCKBqmj9NqXSEIIrJ43bizBwZppUaGGyY1KtUhx2zog7sC7IUHarVCncIHG61WS9NaBkEIP7777NmzNUC5vy7bf0LPnK6MFCWM7lGoPs7qahNQWnYhakjh0PC1We2RqMZ6rckBebbnKz3x
4RMj1gOK7xy4R4nO6rDNTC/9g9pBY+ra2uizjSTxtTT9KVOMsKcrkgoHp4+M673HWZrI49P0UIbH0bWo5djlNhQgg10xR/JFkMMnSIEt4kmy7kGEHYJfjypL4beh27FLVLHG7bPGlQpr2eTh2San5wcOMVjhg
iYqmR3W40iovl4ttjyZILYWynIeBd39ULRJJDZZnOn1zNvZg/aeTz+dLyvdN0Ll98663vt4uLXI23v3/rrcVxZelInAaW4c8SFssEH2pzxZMEHyjSp6enIg6lOUvQtRs3rl0TN8afJRffchIqy9WNjeJJLsFb
RRPRGuzEz1/gcvPTngFZTJ37JX49z9d/9+cc3rFZGye4aWaHoBELPuLIk7y3dutmrgClpaxlm1n7O5U7SwawAhkQ9ef06fFwVf5wutdtlF0vnZ3PFLBtnljdvvuPaICveLyL5y6+nAbIUeyulTObQMlBhpP4s
I4g5XWHMUWqvVqiOkteAfpfVmq9mSkN7SfnuRTgSk3MZ8vzoPd++3CAKEzNztmIzI6DKwlOxj/4i11oD2vmSk6cCUmXbtX4PbfNeE9fSUjerNUVwa9ZmNIVbGI7O/YRo5y0usCcvgNGEZnCYsg9OEZXCasAxC
        background-size: cover;
        }
        h3.big
        {
        line-height: 1.8;
        }
    </style>
</head>
<body>
    <br>
    <div class="container">

        <div class="row">
            <div class="col-md-12 bg-light text-right">
                <a href="/home" class="btn btn-info btn-lg">Home</a>
                <a href="/predict" class="btn btn-primary btn-lg">Predict</a>

            </div>
        </div>
         <center>
             <h1><strong>Online Payments Fraud Detection</strong></h1>
         </center>

         <h3 class="big"><em>The objective of this article is to predict online payments fraud given the various parameters. This will be a classification pr
We will be using classification algorithms such as Decision tree, Random forest,  svm, and Extra tree classifier.  We will train and test the data with thes

        </em></h3><br>

    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</body>
</html>
```

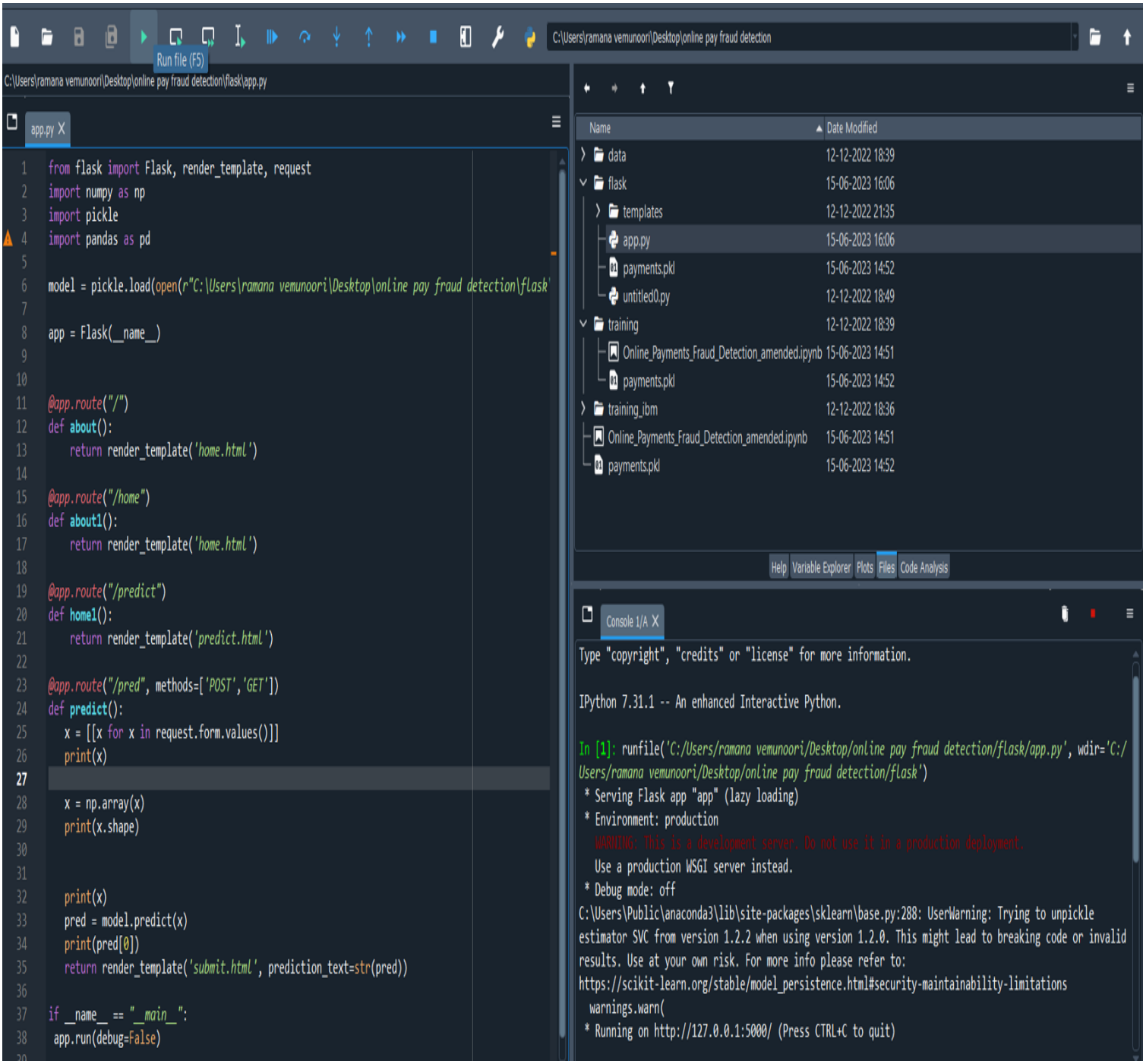



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predict</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <style>
        body
        {
        background-image: url("
sCR8rX82yGt0BMzosjTMhOixjlcg6eWRYIgHq1Sx5ML/o1zpEkWHJy8bzsUxF4wV7FRWWcsqHpaoM3iMCfeWiX+wQRYQl9k3SVdOseu9PpSIayiPCEvmh1K+AdEX1sKzgSMKMCEsfs1QEyitXTMXr59GMPhFh6RPCtSu2dMXD0pjY5VD55ZZXHA06eSx
i3jdFza1mxv+3U/xwZOlSBFjKvV4wZDzKsPBTVdws1b49aCgzjNe0/OAv/7MsVaXqi5LH8G5b2GLpct0zlnGgbJuFfxFCEWCJ+hYMAky/IdNU7F27NssKOW9vwxxZRRQNlv1zuj4wTfcUhSgp2TkR0UvJkvYCqju0IkT/OjA/7pmG5AWP8zOiQDfOuWc
L2PqD6Z6GEqoqmhbCKBqmj9NqXSEIIrJ43bizBwZppUaGGyY1KtUhx2zog7sC7IUHarVCncIHG61WS9NaBkEIP7777NmzNUC5vy7bf0LPnK6MFCWM7lGoPs7qahNQWnYhakjh0PC1We2RqMZ6rckBebbnKz3xDib/zVSrzTXnnjbnDvZkeEpRQ+7/He+
4RMj1gOK7xy4R4nO6rDNTC/9g9pBY+ra2uizjSTxtTT9KVOMsKcrkgoHp4+M673HWZrI49P0UIbH0bWo5djlNhQgg10xR/JFkMMnSIEt4kmy7kGEHYJfjypL4beh27FLVLHG7bPGlQpr2eTh2San5wcOMVjhgWvjvChICS5LJZOrNDqOD06YOSz67DWp
iYqmR3W40iovl4ttjyZILYWynIeBd39ULRJJDZZnOn1zNvZg/aeTz+dLyvdN0Ll98663vt4uLXI23v3/rrcVxZelInAaW4c8SFssEH2pzxZMEHyjSp6enIg6lOUvQtRs3rl0TN8afJRffchIqy9WNjeJJLsFb82Lc5aHMZuno2mIj9G2Ro5nvsQqPEFj
RRPRGuzEz1/gcvPTngFZTJ37JX49z9d/9+cc3rFZGye4aWaHoBELPuLIk7y3dutmrgClpaxlm1n7O5U7SwawAhkQ9ef06fFwVf5wutdtlF0vnZ3PFLBtnljdvvuPaICveLyL5y6+nAbIUeyulTObQMlBhpP4sN/JzZGxZsn6WfMUQQSSZSJ+U0mLQjL+
I4g5XWHMUWqvVqiOkteAfpfVmq9mSkN7SfnuRTgSk3MZ8vzoPd++3CAKEzNztmIzI6DKwlOxj/4i11oD2vmSk6cCUmXbtX4PbfNeE9fSUjerNUVwa9ZmNIVbGI7O/YRo5y0usCcvgNGEZnCYsg9OEZXCasAxOE5bBacIyMP0/QtHs7+6cnKcAAAAASUV
        background-size: cover;
        }
        h3.big
        {
        line-height: 1.8;
        }
    </style>
</head>
<body>
    <br>
    <div class="container">

        <div class="row">
            <div class="col-md-12 bg-light text-right">
                <a href="/home" class="btn btn-info btn-lg">Home</a>
                <a href="/predict" class="btn btn-primary disabled btn-lg">Predict</a>
            </div>
        </div> <br>
         <h1><strong>Online Payments Fraud Detection</strong></h1><br>
          <h4>
          <form action="/pred", method="POST">
```

```
        <div class="form-group row">
            <div class="col-md-3">
                <label for="step">Step</label>
                <input type="number" class="form-control" name="step" id="step"  placeholder="step: represents a unit of time where 1 step equals 1 hour" required="required"/>
            </div>
        </div>
         <div class="form-group row">
            <div class="col-md-3">
                <label for="type">Type</label>
                <input type="number" class="form-control" name="type" id="type"  placeholder="type of online transaction" required="required"/>
            </div>
         </div>
          <div class="form-group row">
              <div class="col-md-3">
                  <label for="amount">Amount</label>
                  <input type="number" class="form-control" name="amount" min=5 max=15 step=0.000001 id="amount"  placeholder="the amount of the transaction" required="required"/>
              </div>
          </div>
          <div class="form-group row">
              <div class="col-md-3">
                  <label for="oldbalanceOrg">OldbalanceOrg</label>
                  <input type="number" class="form-control" name="oldbalanceOrg" min=1000 max=7500000 step=0.01 id="oldbalanceOrg"  placeholder="balance before the transaction" requ
              </div>
          </div>
           <div class="form-group row">
               <div class="col-md-3">
                   <label for="newbalanceOrig">NewbalanceOrig</label>
                   <input type="number" class="form-control" name="newbalanceOrig" min=0 max=500000 step=0.01 id="newbalanceOrig"  placeholder="balance after the transaction" requir
               </div>
           </div>
            <div class="form-group row">
                <div class="col-md-3">
                    <label for="oldbalanceDest">OldbalanceDest</label>
                    <input type="number" class="form-control" name="oldbalanceDest" min=0 max=6500000 step=0.01 id="oldbalanceDest"  placeholder="initial balance of recipient before
                </div>
            </div>
             <div class="form-group row">
                 <div class="col-md-3">
                     <label for="newbalanceDest">NewbalanceDest</label>
                    <div class="form-group row">
                        <div class="col-md-3">
                            <label for="newbalanceOrig">NewbalanceOrig</label>
                            <input type="number" class="form-control" name="newbalanceOrig" min=0 max=500000 step=0.01 id="newbalanceOrig"  placeholder="balance after the transaction" requir
                        </div>
                    </div>
                    <div class="form-group row">
                        <div class="col-md-3">
                            <label for="oldbalanceDest">OldbalanceDest</label>
                            <input type="number" class="form-control" name="oldbalanceDest" min=0 max=6500000 step=0.01 id="oldbalanceDest"  placeholder="initial balance of recipient before
                        </div>
                    </div>
                     <div class="form-group row">
                         <div class="col-md-3">
                             <label for="newbalanceDest">NewbalanceDest</label>
                             <input type="number" class="form-control" name="newbalanceDest" min=0 max=7500000 step=0.01 id="newbalanceDest"  placeholder="the new balance of recipient after
                         </div>
                     </div>

                <button type="submit" class="btn btn-success btn-lg">Submit</button>
    </form>
            <br>
            </h4>
        </div>


        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    </body>
    </html>
```

Figure : html code

Python code:


Run file (F5)
C:\Users\ramana vemunoori\Desktop\online pay fraud detection
C:\Users\ramana vemunoori\Desktop\online pay fraud detection\flask\app.py
app.py
from flask import Flask, render_template, request
import numpy as np
import pickle
import pandas as pd
model = pickle.load(open(r"C:\Users\ramana vemunoori\Desktop\online pay fraud detection\flask'
app = Flask(__name__)
@app.route("/")
def about():
return render_template('home.html')
@app.route("/home")
def about1():
return render_template('home.html')
@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))
if __name__ == "__main__":
app.run(debug=False)
Name
Date Modified
data 12-12-2022 18:39
flask 15-06-2023 16:06
templates 12-12-2022 21:35
app.py 15-06-2023 16:06
payments.pkl 15-06-2023 14:52
untitled0.py 12-12-2022 18:49
training 12-12-2022 18:39
Online_Payments_Fraud_Detection_amended.ipynb 15-06-2023 14:51
payments.pkl 15-06-2023 14:52
training_ibm 12-12-2022 18:36
Online_Payments_Fraud_Detection_amended.ipynb 15-06-2023 14:51
payments.pkl 15-06-2023 14:52
Help
Variable Explorer
Plots
Files
Code Analysis
Console 1/A
Type "copyright", "credits" or "license" for more information.
IPython 7.31.1 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/ramana vemunoori/Desktop/online pay fraud detection/flask/app.py', wdir='C:/Users/ramana vemunoori/Desktop/online pay fraud detection/flask')
* 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
C:\Users\Public\anaconda3\lib\site-packages\sklearn\base.py:288: UserWarning: Trying to unpickle estimator SVC from version 1.2.2 when using version 1.2.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Web UI Page:

Figure

CONCLUSION

Inputs given to web page


127.0.0.1:5000/predict
Gmail
YouTube
Maps
TASK – Telangana A...
Home Predict
Online Payments Fraud Detection
Step 1
Type 1
Amount 5
OldbalanceOrg 1000
NewbalanceOrig 0.01
OldbalanceDest 0.01
NewbalanceDest 0.01
Submit

Figu


127.0.0.1:5000/pred
Gmail
YouTube
Maps
TASK – Telangana A...
Home Predict
Online Payments Fraud Detection
The predicted fraud for the online payment is ['is not Fraud']

figure

3. APPLICATIONS

The areas where this solution can be applied:

- Can be applied in each and every individual's Daily Life.
- Bank transfers
- Digital wallets like google pay
- QR codes/UPI
- BNPL

4. ADVANTAGES

Some advantages of online payments:

- Speed of transactions
- Convenience
- Reaching global audience
- Availability of more distribution channels
- Better customer experience
- Easy management
- Recurring payment capabilities
- Low transaction costs
- Quick and easy setup

5.DISADVANTAGES

Some disadvantages of online payments:

- Technical problems
- Password threats
- Cost of fraud
- Security Concerns
- False identity
- Loss of smart card
- Limitations on amount and time
- Service fees and other additional costs

7.FUTURE SCOPE

On our Dataset, we have applied Random Forest, Decision Tree, Xgboost Classifier, SVM, and Extra tree classifier , Xgboost has got the highest accuracy.

Enhancements that can be made in the future:

Online Fraud Transaction Detection System is basically an extension of the existing system .Using This system, the algorithms will be built to through the dataset and provide the appropriate output. In the long run, this system will be quite beneficial as it provides an efficient system to create a secure transaction system to analyse and detect fraudulent transactions. The Xgboost algorithm is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. This accuracy can be increased further by providing a huge dataset for model training. The scope of this application is very far reaching. This system can be used to detect the features of fraud transactions in a dataset which is very well applicable in various sectors like banking, insurance, e-commerce, money transfer, bill payments, etc. This will indeed help to increase security.

5. BIBILOGRAPHY

1. K.Chaudhary, J.Yadav, “A review of fraud: A comparative study.”decis. Support syst, vol 50, no3, pp.602-613,2011

2. Katherine J. Barker , Jackie D’Amato ,Paul Sheridon,2008 “Credit card fraud :awareness and prevention”, Journal+- of financial Crime ,Vol. 15issue:4,pp.398-410

3.“CreditCard Fraud Detection Based on Transaction Be haviour -by John Richard D. Kho, Larry A. Vea” published by Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017.

4. Customer Transaction Fraud Detection Using Xgboost Model -by Yixuan Zhang, Ziyi Wang, Jialiang Tong, Fengqiang Gao June, 2020

5. Wang, M., Yu, J., & Ji, Z. (2018). Credit Fraud Risk Detection Based on XGBoost-LR Hybrid Model

6. Mishra, C. Ghorpade, “Credit Card Fraud Detection on the Skewed Data Using Various Classification and Ensemble Techniques” 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) pp. 1- 5. IEEE

HELP FILE

PROJECT EXECUTION:

STEP-1: Go to **Start,** search and launch **ANACONDA NAVIGATOR.**

STEP-2: After launching of **ANACONDA NAVIGATOR,** launch **JUPYTER NOTEBOOK.**

STEP-3: Open **"Major project code" IPYNB file.**

STEP-4: Then run all the cells.

STEP-5: All the **data preprocessing**, **training and testing**, **model building**, **accuracy** of the model can be showcased.

STEP-6: And a pickle file will be generated.

STEP-7: Create a Folder named **FLASK** on the **DESKTOP.** Extract the pickle file into this Flask Folder.

STEP-8: Extract all the html files (home.html, index.html, chance.html, nochance.html) and python file(app.py) into the **FLASK Folder.**

STEP-9: Then go back to **ANACONDA NAVIGATOR** and the launch the **SPYDER.**

STEP-10: After launching Spyder, give the path of **FLASK FOLDER** which you have created on the DESKTOP.

STEP-11: Open all the app.py and html files present in the Flask Folder.

STEP-12: After running of the app.py, open **ANACONDA PROMPT** and follow the below steps:

cd File Path→click enter

python app.py→click enter (We could see running of files).

STEP-13: Then open **BROWSER,** at the URL area type **–localhost:5000".**

STEP-14: Home page of the project will be displayed.

STEP-15: Click on **–Go to Predict".** Directly it will be navigated to index page.

STEP-16:A index page will be displayed where the user needs to give the inputs and then click on **–Predict"**. Output will be generated whether a person is having liver disease or not.