ONLINE PAYMENTS FRAUD DETECTION USING MACHINE LEARNING

AN INDUSTRIAL ORIENTED UG PHASE-2 REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment 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 guidance of

Mr.G.RAMESH

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
VAAGDEVI ENGINEEERING COLLEGE

Affiliated to JNTU, HYDERBAD

BOLLIKUNTA, WARANGAL, (T.S)-506005

2019-2023

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

VAAGDEVI ENGINEEERING COLLEGE

BOLLIKUNTA, WARANGAL, (T.S)-506005

CERTIFICATE

This is to certify that the UG 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 fulfilment 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 **2019-2023.**

Project Guide	**Head of Department**
Mr. G.RAMESH	**Dr. R. NAVEEN KUMAR**
(Associate Professor)	(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 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 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 Phase-2 and for their support in completing the UG Phase-2.

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

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

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

ABSTRACT

In today's world, people depend on online payments for almost everything. Online transactions have their own merits like easy to use, feasibility, faster payments etc., but these kinds of transactions also have some demerits like fraud transactions, phishing, data loss, etc. With increase in online transactions, there is a constant threat for frauds and misleading transactions which can breach an individual's privacy. Hence, many commercial banks and insurance companies devoted millions of rupees to build a transaction detection system to prevent high risk transactions. We presented a machine learning - based transaction fraud detection model with some feature engineering. The algorithm can get experience; improve its stability and performance by processing as much as data possible. These algorithms can be used in the project that is online fraud transaction detection. In these, the dataset of certain transactions which is done online is taken. Then with the help of machine learning algorithms, we can find the unique data pattern or uncommon data patterns which will be useful to detect any fraud transactions. For the best results, the XGBoost algorithm will be used which is a cluster of decision trees. This algorithm is recently dominating this ML world. This algorithm has features like more accuracy and speed when compared to other ML algorithms.

Keywords – Fraud detection, Machine learning, Xgboost algorithm, classification, Data preprocessing, Prediction.

TABLE OF CONTENTS

1. INTRODUCTION

In today's world, we are on the way to become a cashless world. According to various surveys and researches, people performing the online transactions is 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 or for knowing a persons personal info, 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 users to estimate, perceive or avoid objectionable behavior, which consists of fraud, intrusion, and defaulting.

The online payment systems has helped a lot in the ease of payments. But, at the same time, it increased in payment frauds. Online payment frauds can happen with anyone using any payment system, especially while making payments using a credit card / debit card. That is why detecting online payment fraud is very important for credit card companies to ensure that the customers are not getting charged for the products and services they never paid.

Most of the E-commerce sites runs on online payments the fraudsters are ready to get the information / personal data once if the fraudster is known the card CVV number or payment UPI-ID then the fraudsters are entering and knowing the personal data of an individual, Even if they know the card number they can predict

CVV number. Because there are many ways now-a-days to predict and various algorithms to predict this may leads to the losing the personal data of a individual without is concern

2. CODE SNIPPETS

2.1 MODEL CODE

```
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.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import xgboost as xgb
```

```
[5] data = pd.read_csv(r'/content/drive/MyDrive/Major proj Dataset/PS_20174392719_1491204439457_logs.csv')
```

```
[6] from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Figure 1: .ipynb code importing libraries & mounting dataset from Drive.

```
data
```

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

2430 rows × 11 columns

```
[8] data.columns
```

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

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

```
[10] data.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.0	C79051264	51433.88	108179.02	1	0
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.0	C1140210295	0.00	0.00	1	0
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.0	C1759363094	0.00	33676.59	1	0
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.0	C757947873	0.00	0.00	1	0
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.0	C1827219533	0.00	87999.25	1	0

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

Figure 2: .ipynb code displaying few rows, columns & column names from the dataset.

```
data.info() #shows the descriptive statistics

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2430 entries, 0 to 2429
Data columns (total 10 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   nameOrig        2430 non-null   object
 4   oldbalanceOrg   2430 non-null   float64
 5   newbalanceOrig  2430 non-null   float64
 6   nameDest        2430 non-null   object
 7   oldbalanceDest  2430 non-null   float64
 8   newbalanceDest  2430 non-null   float64
 9   isFraud         2430 non-null   int64
dtypes: float64(5), int64(2), object(3)
memory usage: 190.0+ KB
```

Figure 3: .ipynb code describe in detail info using info() method.

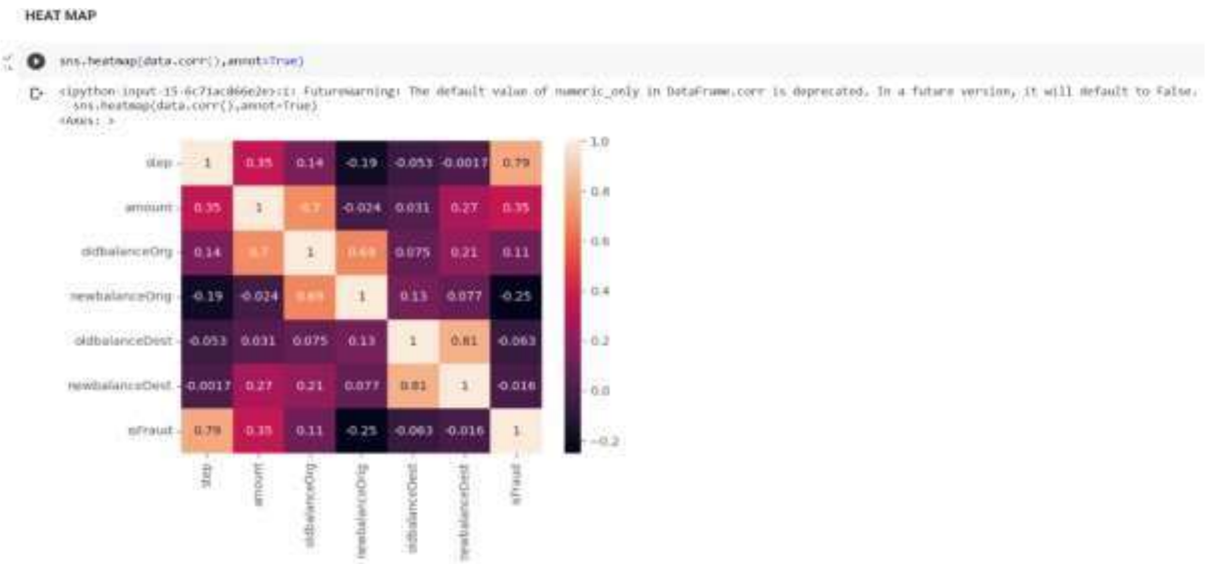


Figure 4: .ipynb code for heatmap shows 2 dimensional representation of dataset.

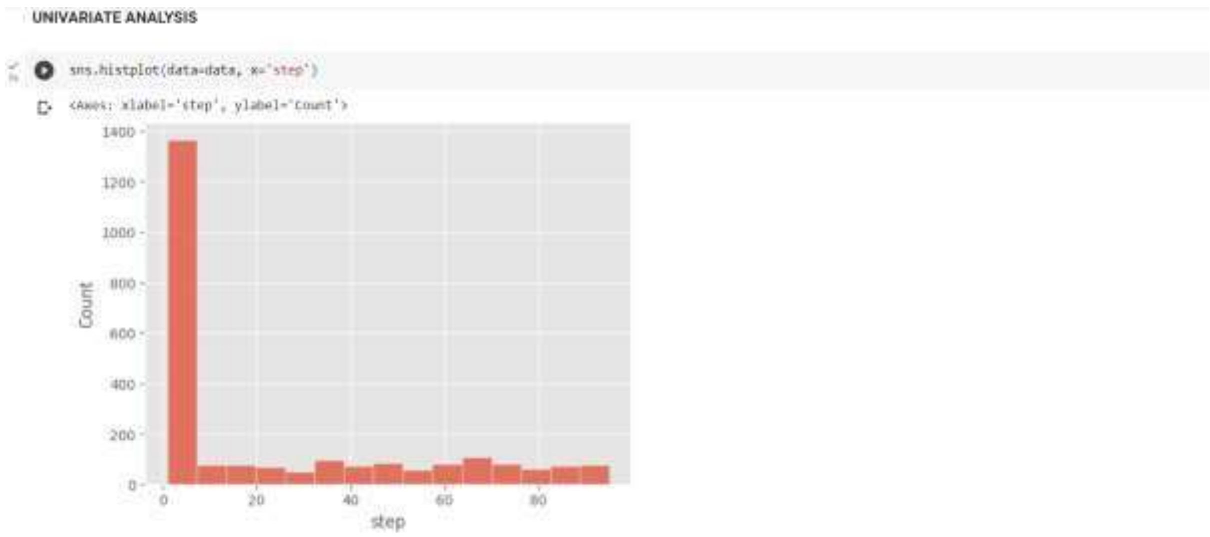
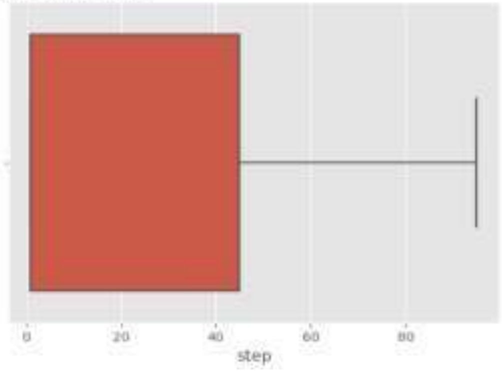


Figure 5: .ipynb code for univariate analysis of step column.

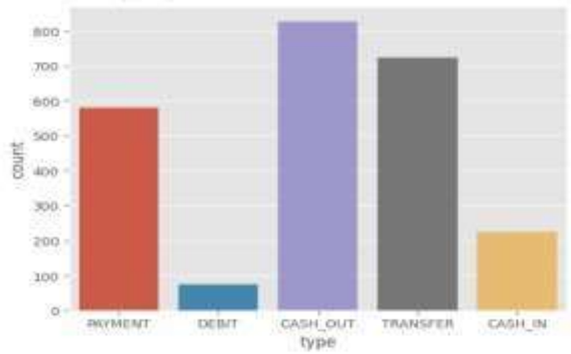
```python
sns.boxplot(data=data, x='step')
```

<Axes: xlabel='step'>



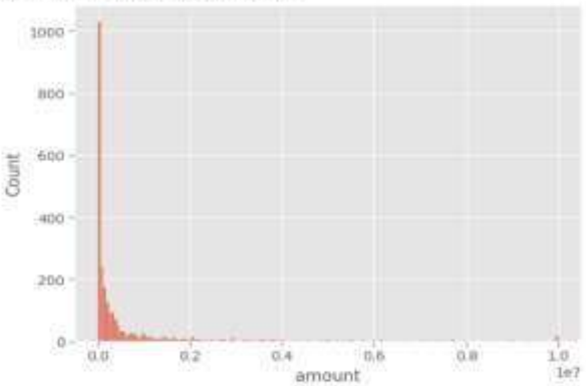

```python
sns.countplot(data=data,x='type')
```

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




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

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




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

<Axes: xlabel='amount'>

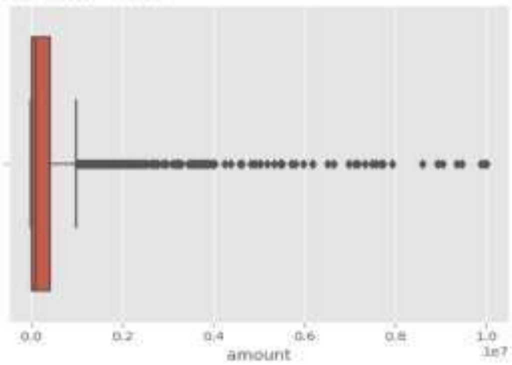
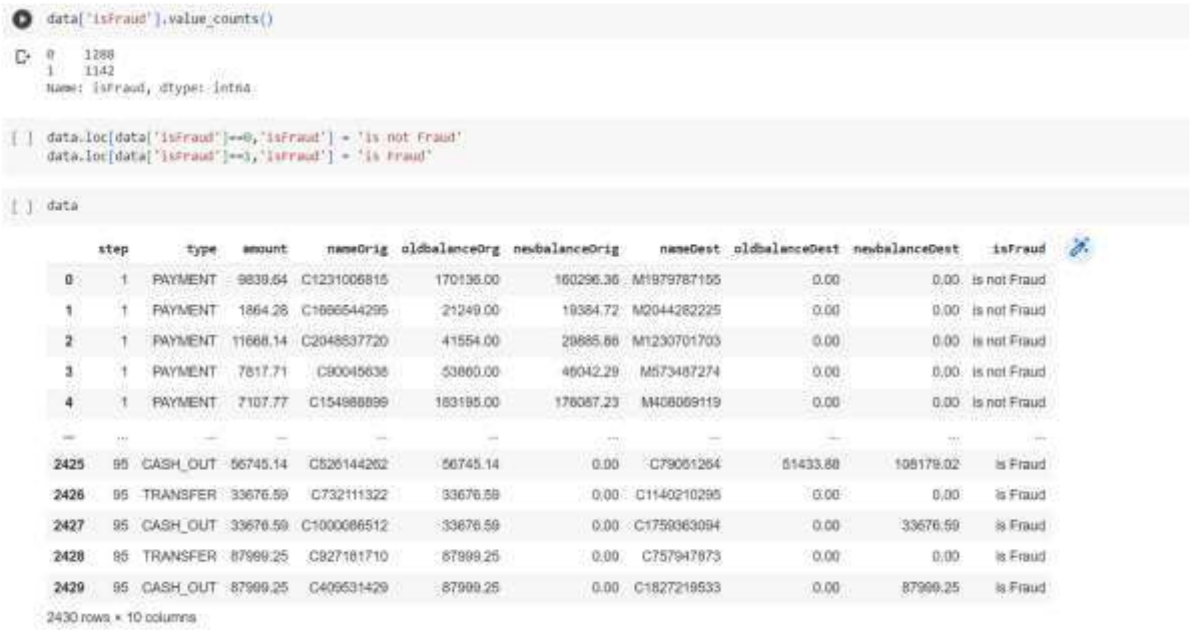


Figure 6: .ipynb code for different columns present in dataset.



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

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

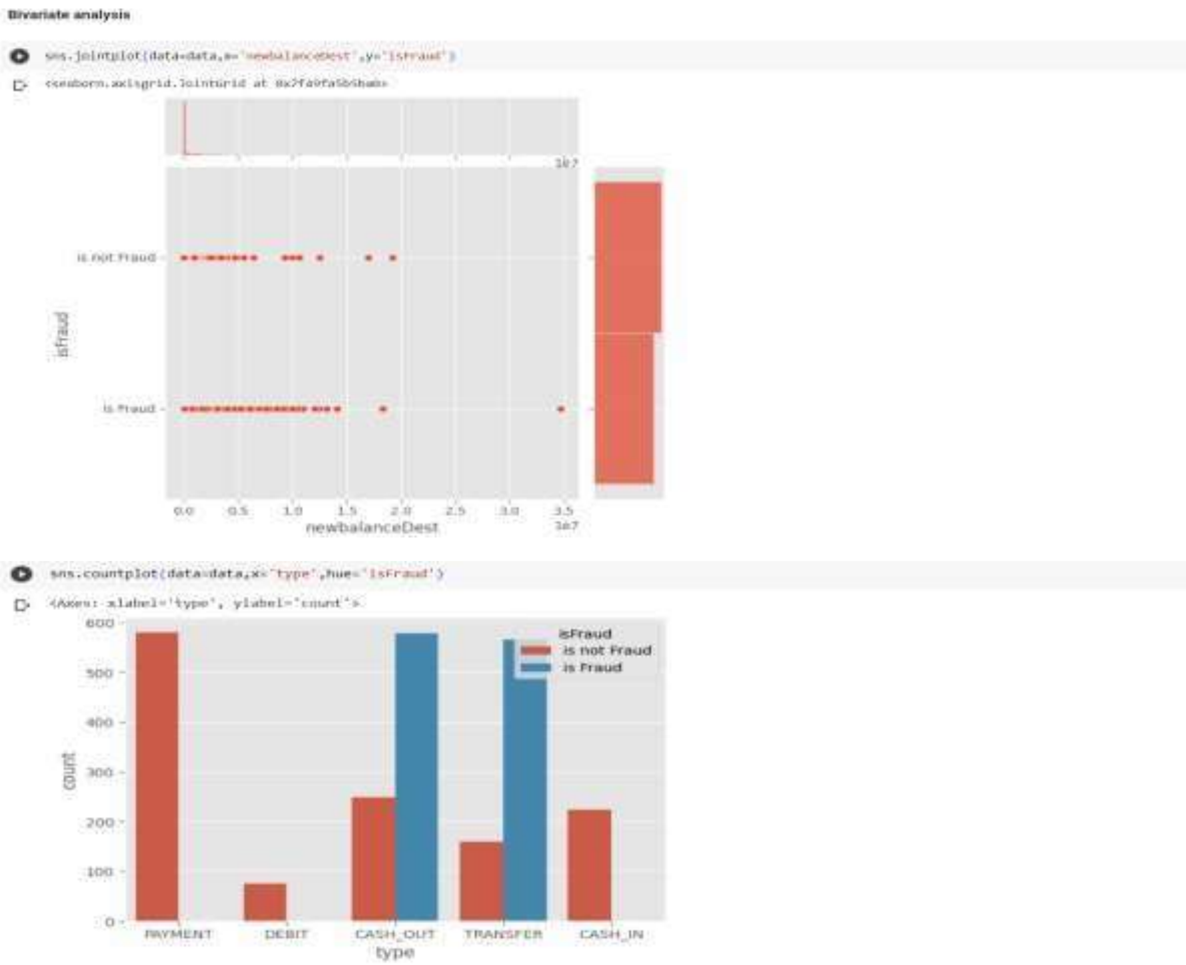
```
data.loc[data['isFraud']==0,'isFraud'] = 'is not Fraud'
data.loc[data['isFraud']==1,'isFraud'] = 'is Fraud'
```

```
data
```

	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

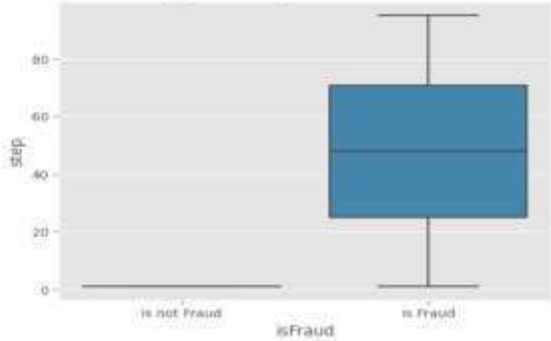
2430 rows × 10 columns

Figure 7: .ipynb code for count of fraud and non fraud transactions & Assigining is fraud=1 & is not fraud=0, displaying dataset.



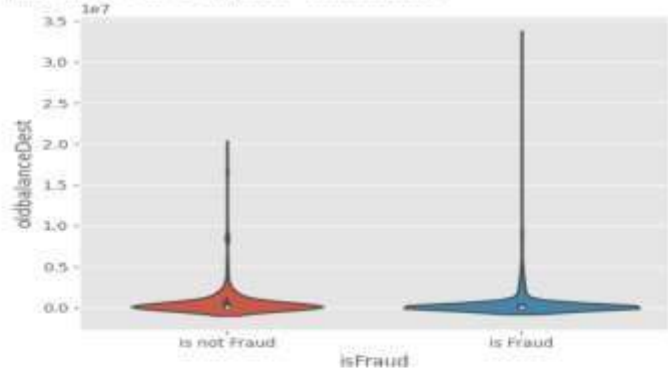
```
sns.boxplot(data=data,x='isFraud',y='step')
```

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




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

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




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

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

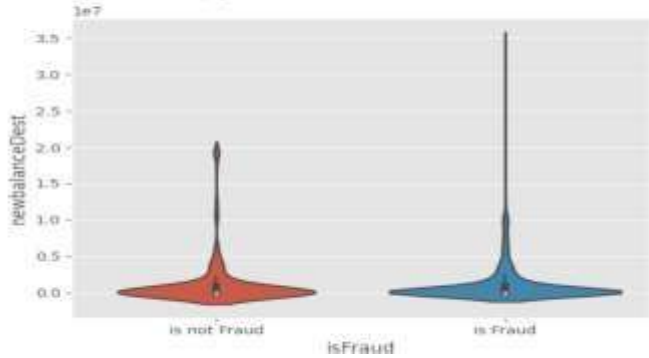


Figure 8: .ipynb code displaying Bi-variate analaysis gives relationship between each variable in dataset.

Descriptive analysis

```
data.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	6.258381e+05	NaN	9.849040e+05	4.392750e+05	NaN	8.797246e+05	1.127075e+06	NaN
std	29.933036	NaN	1.503888e+06	NaN	2.082361e+06	1.520978e+06	NaN	1.891192e+06	2.007401e+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	6.018493e+03	NaN	5.879630e+03	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
50%	1.000000	NaN	1.058882e+05	NaN	6.096250e+04	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
75%	45.000000	NaN	4.006066e+05	NaN	7.606258e+05	1.247804e+04	NaN	3.088195e+05	8.668701e+05	NaN
max	95.000000	NaN	1.000000e+07	NaN	1.990000e+07	9.887287e+06	NaN	3.300000e+07	3.460000e+07	NaN

Figure 9: .ipynb code for descriptive analysis it describes the data.

Data Preprocessing

```
[63] data.shape

(2430, 10)
```

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

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

```
[65] data.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

```
[66] data.isnull().sum()

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

```
[67] data.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 10: .ipynb code for Data preprocessing, Raw data to processing procedure.

Remove the Outliers

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

```
ModeResult(mode=array([10000000.]), count=array([14]))
825836.0074156379
<ipython-input-69-d0edd6d81bac>:2: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it
  print(stats.mode(data['amount']))
```

```
[70] q1 = np.quantile(data['amount'],0.25)
     q3 = np.quantile(data['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(data[data['amount']>upper_bound]))
     print('Skewed data :',len(data[data['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
```

```
[71] def transformationPlot(feature):          # To handle outliers transformation techniques are used.
         plt.figure(figsize=(12,5))
         plt.subplot(1,2,1)
         sns.distplot(feature)
         plt.subplot(1,2,2)
         stats.probplot(feature,plot=plt)
```

Figure 11: .ipynb code for removing outliers & transformation plot values.

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

```
<ipython-input-71-cea5d542282f>:6: 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)
```

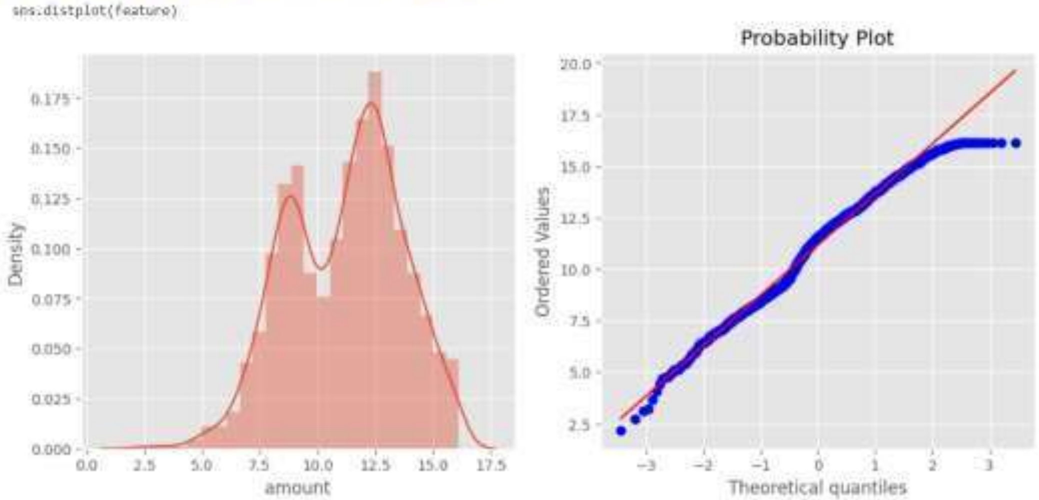


Figure 12: .ipynb code for transformation plot & graphs.

Object data labelencoding

```
[74]
la = LabelEncoder()
data['type'] = la.fit_transform(data['type'])
```

```
[75] data['type'].value_counts()
```

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

```
[76] #dividing the dataset into dependent and independent x and Y respectively
x = data.drop('isFraud',axis=1)
y = data['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

Figure 13: .ipynb code for object label encoding converts categorical values to numerical.

```
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
```

```
[79] #Splitting data into train and test
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

```
[80] print(x_train.shape)
print(x_test.shape)
print(y_test.shape)
print(y_train.shape)
```

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

Figure 14: .ipynb code splitting data into train and test.

Model Building

Random Forest classifier

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

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

```
0.9938271604938271
```

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

```
1.0
```

```
[83] pd.crosstab(y_test,y_test_predict1)
```

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

Figure 15: .ipynb code for Random Forest model.

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
```

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

```
1.0
```

```
[87] pd.crosstab(y_test,y_test_predict2)
```

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

```
[88] 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 16: .ipynb code for Decesion tree classifier.

ExtraTrees Classifier

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

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

     0.9938271604938271
```

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

1.0
```

```
[91] pd.crosstab(y_test,y_test_predict3)
```

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

```
[92] 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
```

Figure 17: .ipynb code for extra trees classifier.

SupportVectorMachine Classifier

```
[93] 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
```

```
[95] pd.crosstab(y_test,y_test_predict4)
```

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

```
[96] 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
```

Figure 18: .ipynb code for support vector machine classifier.

```python
from sklearn.preprocessing import LabelEncoder
la = LabelEncoder()
y_train1 = la.fit_transform(y_train)
```

```python
[99] y_test1=la.transform(y_test)
```

```python
[100] y_test1
```

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

```python
[101] y_train1
```

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

Figure 19: .ipynb code for Label encoding converts categorical columns to numerical columns.

xgboost Classifier

```python
import xgboost as xgb
xgb1 = xgb.XGBClassifier()
xgb1.fit(x_train, y_train1)
y_test_predict5=xgb1.predict(x_test)
test_accuracy=accuracy_score(y_test1,y_test_predict5)
test_accuracy
```

```
0.9979423868312757
```

```python
[103] y_train_predict5=xgb1.predict(x_train)
      train_accuracy=accuracy_score(y_train1,y_train_predict5)
      train_accuracy
```

```
1.0
```

```python
[104] pd.crosstab(y_test1,y_test_predict5)
```

col_0	0	1
row_0		
0	233	1
1	0	252

```python
[105] from sklearn.metrics import classification_report,confusion_matrix
      print(classification_report(y_test1,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       234
           1       1.00      1.00      1.00       252

    accuracy                           1.00       486
```

Figure 20: .ipynb code for xgboost classifier.

Compare Models

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

```
[107] compareModel()

train accuracy for rfc 1.0
test accuracy for rfc 0.9938271604938271
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 0.9979423868312757
```

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

```
[109] pwd

'/content'
```

Figure 21: .ipynb code for comparing the models & accuracy of each model, importing pickle file(.py code).

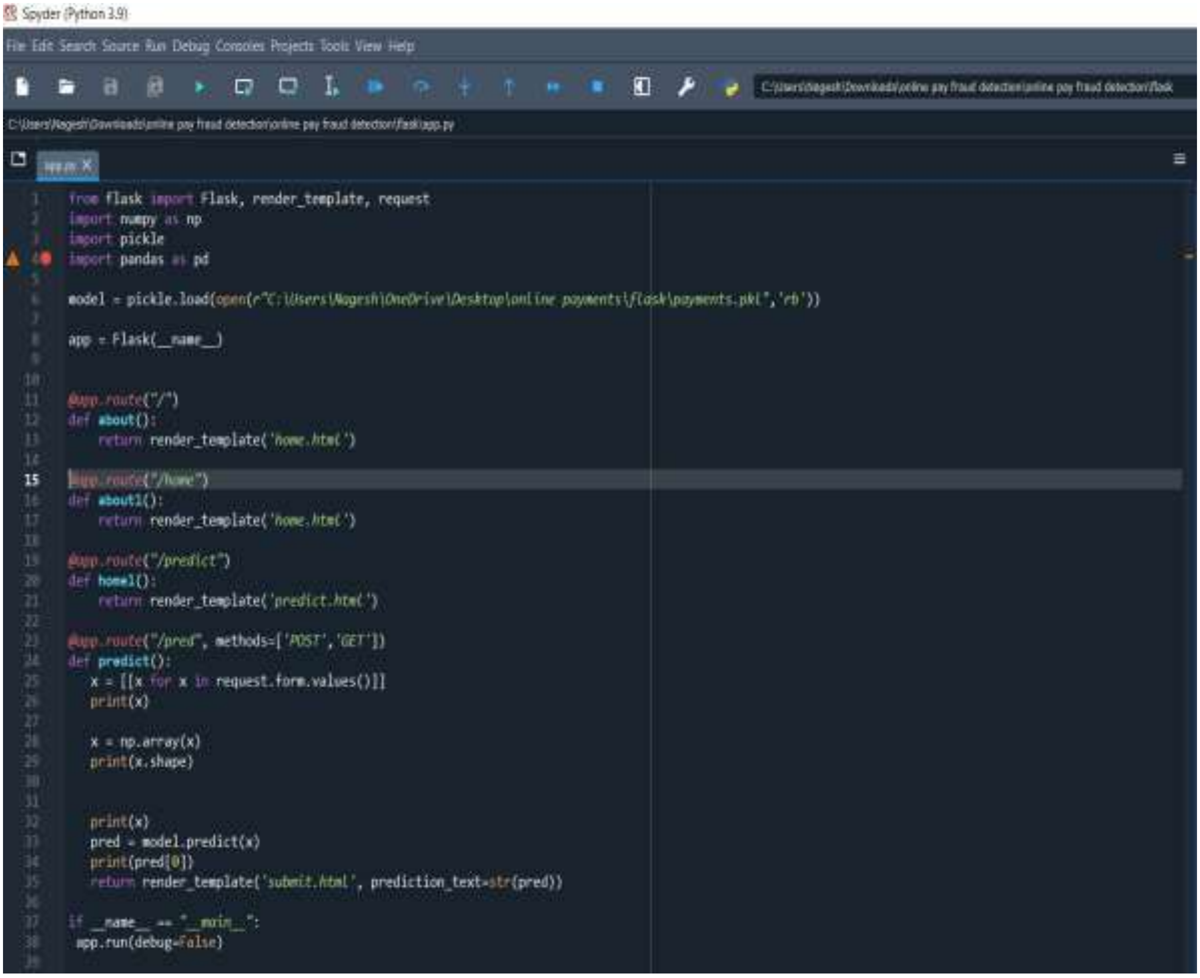
```
[112] # prediction
      #features = [step,type,amount,oldbalanceOrg,newbalanceOrig,oldbalanceDest,newbalanceDest]
      features = np.array([[1,3,9.194174,170136.00,160296.36,0.0,0.00]])
      print(svc.predict(features))

['is not Fraud']
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

Figure 22: .ipynb code for prediction & predicting by giving values.

2.2 HTML CODE AND PYTHON CODE

1. app.py code:



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

model = pickle.load(open(r"C:\Users\Nagesh\OneDrive\Desktop\online payments\flask\payments.pkl",'rb'))

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)
```

Figure 23: .python code used for rendering all the HTML pages.

home.html

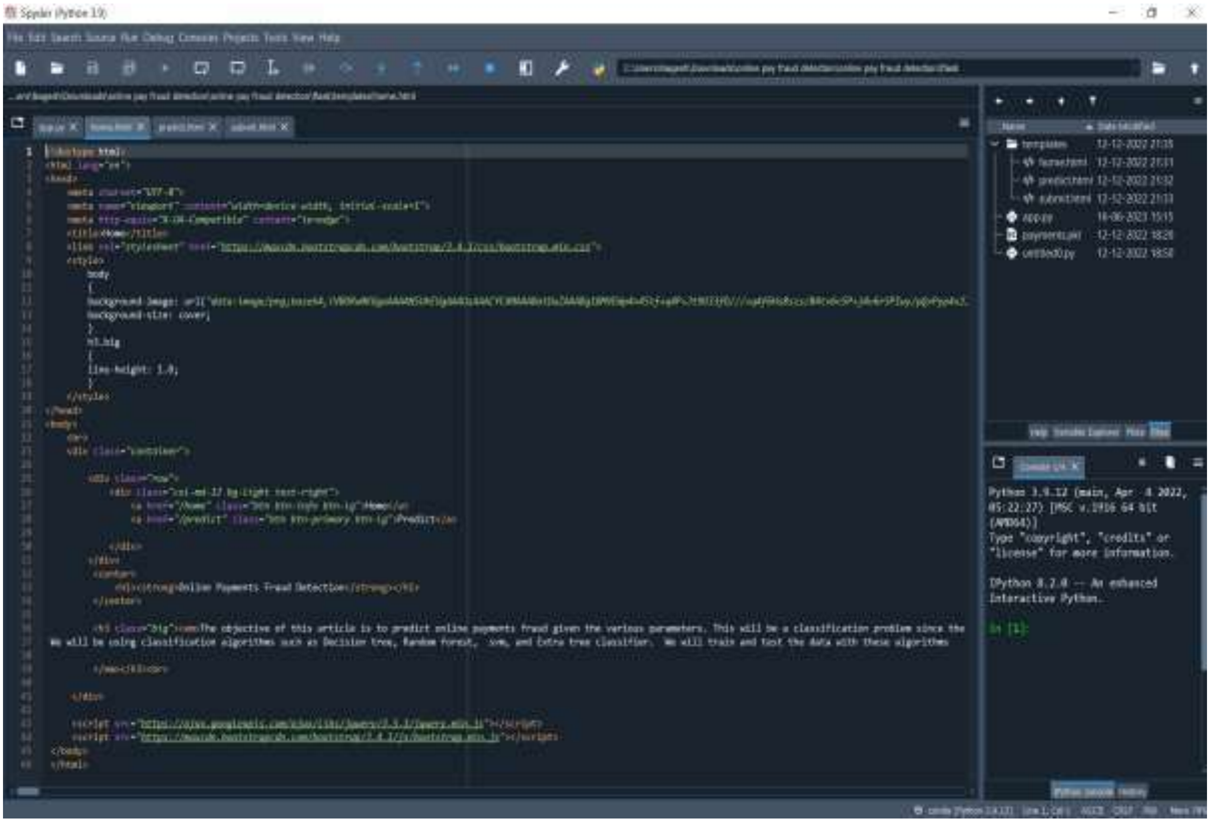
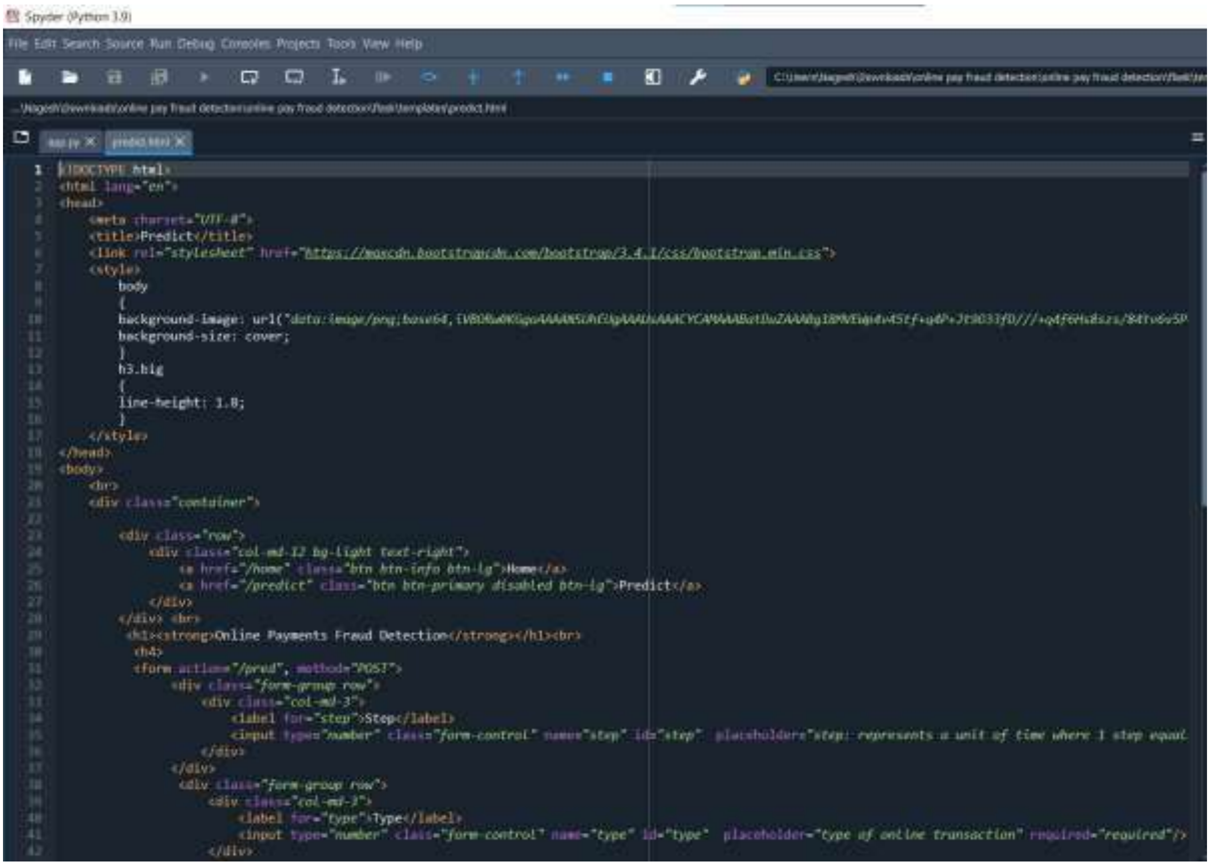


Figure 23: home.html page is the code for homepage of our web application.

predict.html:



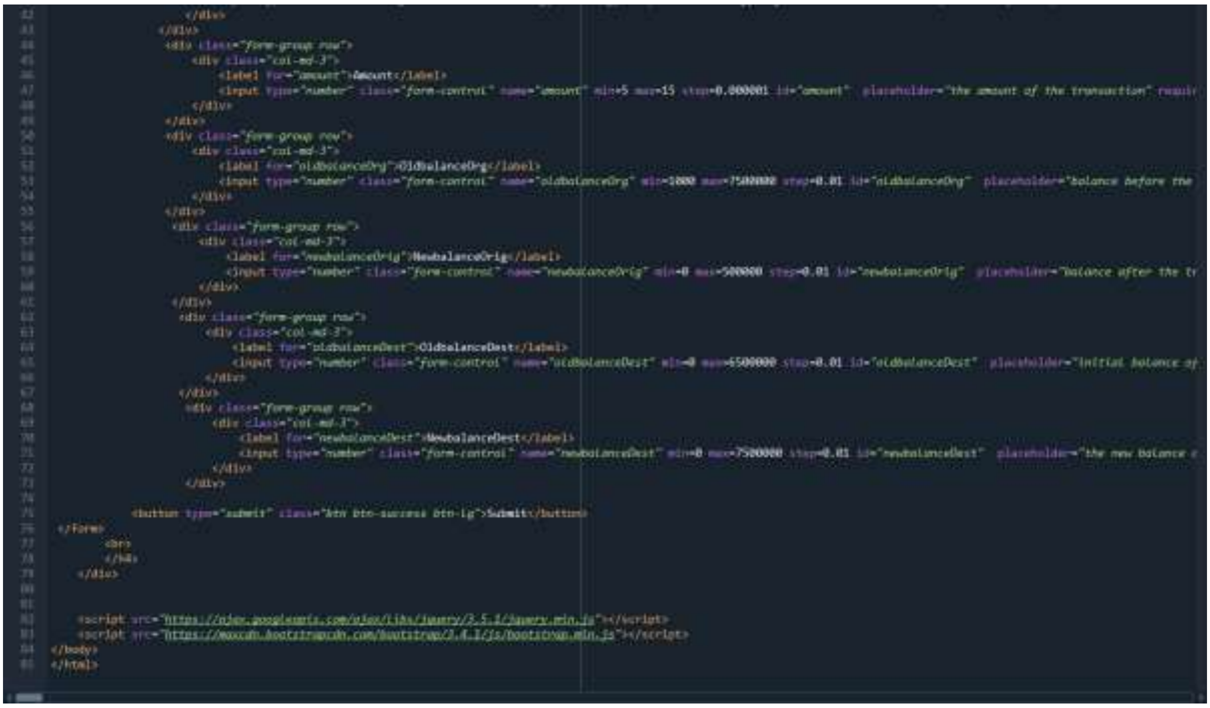
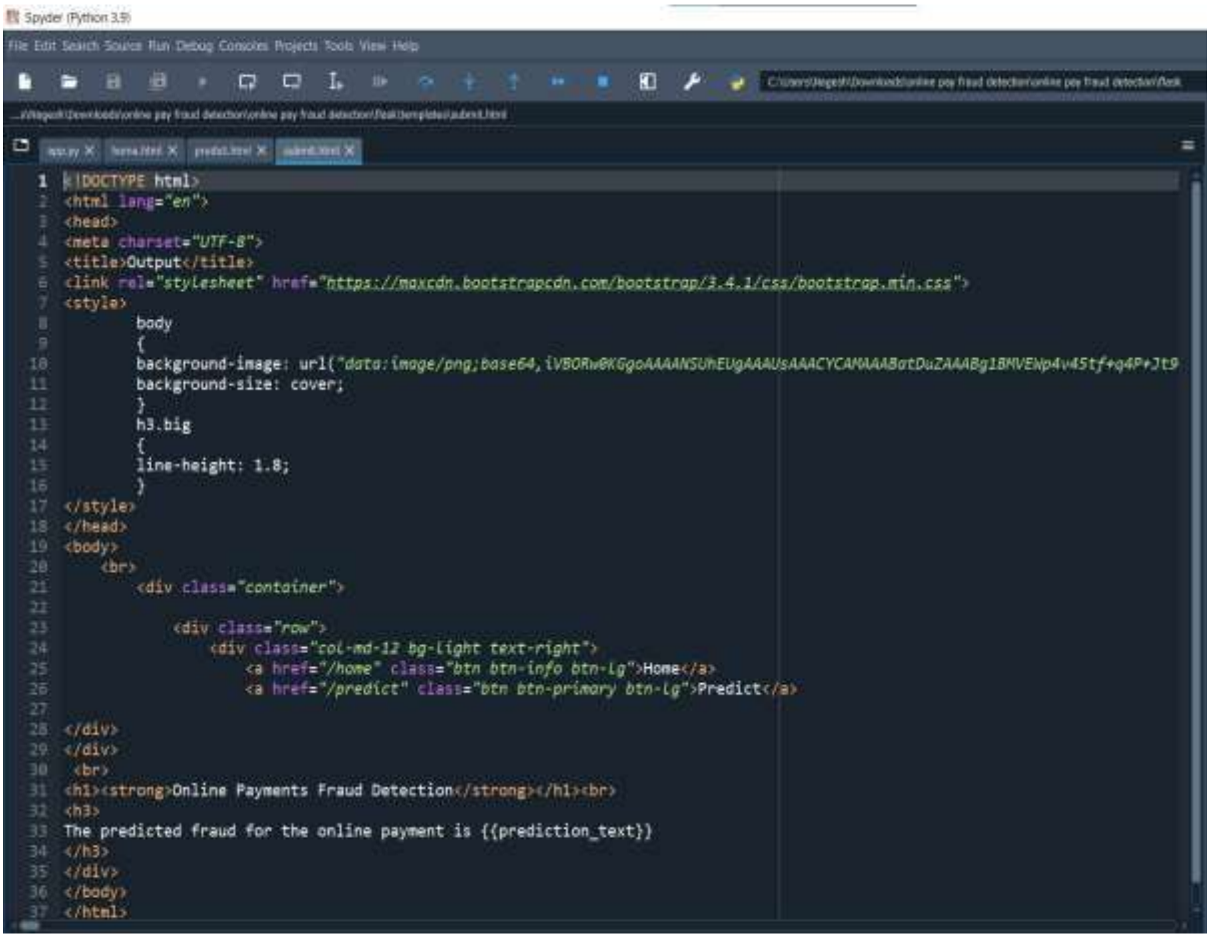


Figure 24: predict.html page which predicts the output. By taking the inputs from user.

Submit.html




```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Output</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<style>
        body
        {
        background-image: url("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAUsAAACYCAMAAABatDuZAAABg1BMVEWp4v45tf+q4P+Jt9
        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>
 <br>
<h1><strong>Online Payments Fraud Detection</strong></h1><br>
<h3>
The predicted fraud for the online payment is {{prediction_text}}
</h3>
</div>
</body>
</html>
```

Figure 25: submit.html is a button when we enter values & click on submit button it displays a message associated with code.

3. CONCLUSION

Figure 26: Home page (which gives introduction to Online payments Fraud Detection)

Figure 27: Input page (which takes input from user)

Figure 28: Output page (Displays that the payment is fraud)

Figure 29: Input page (which takes input from user)

Figure 30: Output page (Displays that the payment is not fraud)

4. APPLICATIONS

The areas where this solution can be applied:

- Bank Transfers & Banking Applications.
- QR codes/UPI payments.
- Digital wallets like phone pe, paytm etc..,
- Swipping machines (card cvv).

5. ADVANTAGES

1. **Improved Security:** Online payment fraud detection projects employ advanced algorithms and techniques to identify and prevent fraudulent activities. This helps in enhancing the overall security of online transactions and protects both businesses and customers.

2. **Real-Time Detection:** Online payment fraud detection systems can analyze transactions in real time, enabling the identification of suspicious patterns or behaviors instantly. This allows for immediate action to be taken, such as blocking a transaction or flagging it for manual review.

3. **Cost Savings:** By implementing an effective fraud detection system, businesses can minimize financial losses due to fraudulent activities. Identifying and preventing fraudulent transactions early on can save significant amounts of money that would otherwise be lost.

4. **Enhanced Customer Trust:** A robust fraud detection system reassures customers that their financial information is secure when making online payments. This helps to build trust and confidence in the business, leading to increased customer satisfaction and loyalty.

5. **Scalability:** Online payment fraud detection systems can handle large volumes of transactions, making them scalable for businesses of different sizes. As the volume of online transactions increases, the system can adapt and accommodate the growing demands.

6. DISADVANTAGES

1. **False Positives:** One of the challenges in online payment fraud detection is the occurrence of false positives, where legitimate transactions are incorrectly flagged as fraudulent. This can inconvenience customers and lead to a loss of business if genuine transactions are blocked or delayed.

2. **Evolving Fraud Techniques:** Fraudsters are continually adapting their techniques to bypass detection systems. Keeping up with new and emerging fraud patterns and updating the fraud detection algorithms accordingly can be challenging.

3. **Privacy Concerns:** Online payment fraud detection projects involve the analysis of large amounts of personal and financial data. Ensuring the privacy and security of this sensitive information is crucial to prevent unauthorized access or data breaches.

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 payment Fraud Transaction Detection System is basically an extension of the existing system. Using This system, the algorithms which we used to train 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.

8. 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.
7. https://thecleverprogrammer.com/2022/02/22/online-payments-fraud-detection-withmachine-learning/
8. https://www.geeksforgeeks.org/online-payment-fraud-detection-using-machinelearning-in-python/

9.HELP LINE

PROJECT EXCEUTION:

STEP-1: Go to Google, search google colaboratory & launch.

STEP-2: After launching of collab. STEP-3:

Open “Major project .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, predict.html, submit.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 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 — Predict. Give the inputs then it will be predict fraud payment or not.