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…………………………………………...1**
   1.1 Motivation……………………………………………...1
   1.2 Defination……………………………………………...1
   1.3 Objective of project……………………………………….2
   1.4 Purpose………………………………………………….2
2. **PROBLEM STATEMENT…………………………………….3**
3. **LITERATURE SURVEY……………………………………...4-5**
   3.1 Exisisting problem…………………………………………..4
   3.2 Problem solution……………………………………………..4-5
4. **EXPERIMENTAL INVESTIGATIONS………………………6-9**
   4.1 Block diagram………………………………………………..8
   4.2 Architecture…………………………………………………..8
   4.3 Software requirements………………………………………..8-9
   4.4 Project flow……………………………………………………9
5. **DESIGN………………………………………………………10**
   5.1 Dataset………………………………………………………..10
   5.2 Use case diagram……………………………………………..11
   5.3 Flowchart……………………………………………………..11
6. **CONCLUSION………………………………………………12**
7. **FUTURE SCOPE……………………………………………13**

LIST OF FIGURES PAGE NO

Figure 1: Block Diagram……………………………………………………………...8

Figure 2: Architecture…………………………………………………………………8

Figure 3: Dataset table………………………………………………………………10

Figure 4: Usecase diagram…………………………………………………………...11

Figure 5: Flowchart…………………………………………………………………..11

1.INTRODUCTION

1.1.MOTIVATION

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.

1.2.DEFINITION

Online payment is the electronic transfer of funds via the internet, usually between a merchant and a consumer. These payments can be made in various ways, such as via credit and debit cards, banking apps or web pages. Exactly which method of online payment you choose to offer and accept will depend on the specifics of your business and the preferences of your target market.

Online payments are payments that are initiated over the internet for goods or services purchased either online or offline. Common methods to facilitate this include:

- Bank Debits via online mandate (often referred to a Direct debit which is the terminology we'll use in this guide)
- Bank transfers (also referred to as wire transfers)
- Online credit or debit card transactions.
- Digital wallet payments (such as PayPal)

OBJECTIVE OF PROJECT:

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

1.4.PURPOSE

The Main Purpose this Guided Project mainly focuses on applying a machine-learning algorithm for online payment fraud with machine learning, we need to train a machine learning model for classifying fraudulent and non-fraudulent payments. For this, we need a dataset containing information about online payment fraud, so that we can understand what type of transactions lead to fraud. For this task, we should collect dataset from Kaggle, which contains historical information about fraudulent transactions which can be used to detect fraud in online payments. online payment systems has helped a lot in the ease of payments. But, at the same time, it increased in payment frauds.

Minimising friction in your payment process saves you time and money and makes positive cash flow more likely. Therefore, it’s important to choose payment collection methods that encourage prompt payment can be automated as possible.

PROBLEM STATEMENT

Necessary preventive measures can be taken to stop this abuse and the behavior of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future. In other words, this is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated. This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions for outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over time.

These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize. Machine learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent. The investigators provide feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time. So, in this project, what we have tried is to create a Web App for the detection of such types of frauds with the help of Machine Learning.

LITERATURE SURVEY

3.1.EXISTING PROBLEM

With the growth of e-commerce websites & bank transactions people and financial companies rely on online services to carry out their transactions that have led to an exponential increase in the online payment frauds. Fraudulent payment transactions lead to a loss of huge amount of money. The design of an effective fraud detection system is necessary in order to reduce the losses incurred by the customers and financial companies. Research has been done on many models and methods to prevent and detect online payments frauds. Some payments fraud transaction datasets contain the problem of imbalance in datasets. A good fraud detection system should be able to identify the fraud transaction accurately and should make the detection possible in real-time transactions. Fraud detection can be divided into two groups: anomaly detection and misuse detection. Anomaly detection systems bring normal transaction to be trained and use techniques to determine novel frauds. Conversely, a misuse fraud detection system uses the labeled transaction as normal or fraud transaction to be trained in the database history. So, this misuse detection system entails a system of supervised learning and anomaly detection system a system of unsupervised learning. Fraudsters masquerade the normal behavior of customers and the fraud patterns are changing rapidly so the fraud detection system needs to constantly learn and update. Payments frauds can be broadly classified into three categories, that is, traditional card related frauds (application, stolen, account takeover, fake and counterfeit), merchant related frauds (merchant collusion and triangulation) and Internet frauds (site cloning, credit card generators and false merchant sites).

3.2.PROBLEM SOLUTION

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.

we have used the Xgboost algorithm which also works based on the decision-making trees.

This algorithm has recently become popular dues to its advantages like fast, efficient, more accurate etc. the training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. It basically classifies the transaction in only two states that are either frud or transation.

Xg boost 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.

When using gradient boosting for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score. XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

EXPERIMENTAL INVESTIGATIONS

Milestone 1: Data Collection

ML depends heavily on data, without data, a machine can't learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

You can collect datasets from different open sources like kaggle.com, data.gov; UCI machine learning repository etc. The dataset used for this project was obtained from Kaggle.

Milestone 2: Data Pre-processing

Data Pre-processing includes the following main tasks

- Importing the libraries.
- Importing the dataset.
- Analyse the data.
- Taking care of Missing Data.
- Data Visualisation.
- Splitting Data into Train and Test

Milestone 3: Model Building

The model building process involves setting up ways of collecting data, understanding and paying attention to what is important in the data to answer the questions you are asking, finding a statistical, mathematical or a simulation model to gain understanding and make predictions.

Model Building Includes:

- Import the model building libraries.
- Initialising the model.
- Training the model.
- Model Evaluation.
- Save the Model.

Milestone 4: Application Building

Create an HTML File.

- Build python code.
- Run the app in local browser.
- Show casting the prediction on UI.

4.1.BLOCK DIAGRAM

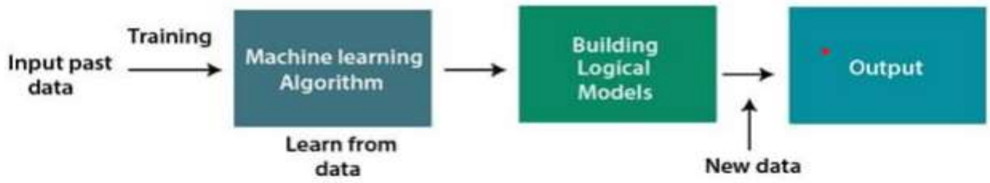
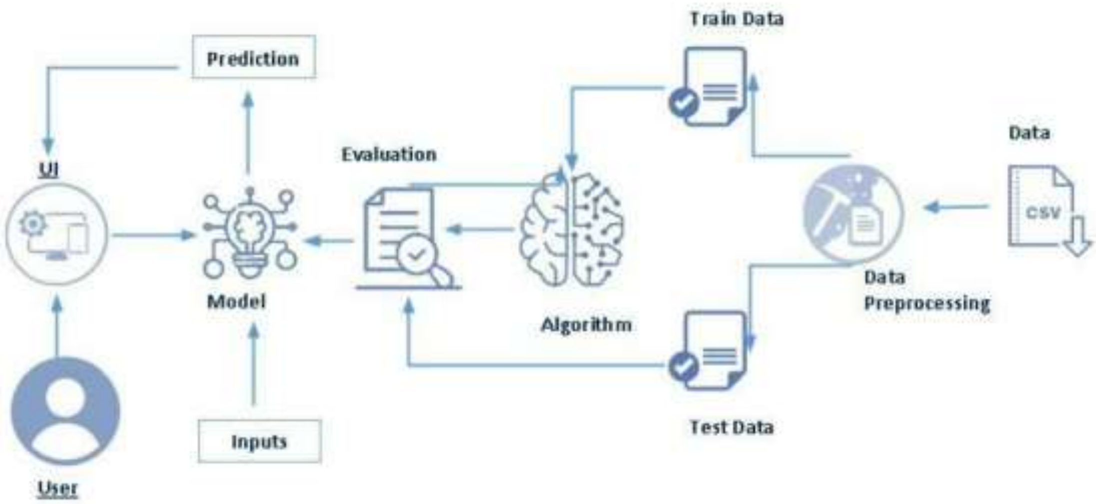


Figure 4 : Block Diagram

4.2.ARCHITECTURE



4.3.SOFTWARE REQUIREMENTS

- Python 3.9 or above:
    - Python is an interpreted high-level general-purpose programming language.
    - Python can be used on a server to create web applications.
- Visual Studio Code:
    - Visual studio code is a source-codeditor made by Microsoft for Windows, linux and macOS.

    - Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
- Anaconda Environment
    - The default environment base (path) is used because it consists of multiple libraries and modules.
- Libraries
    - Pandas and numpy , matplotlib, seaborn, and Algorithms etc..,
- Flask
    - Flask is the module used for web framework.
    - Flask provides you with tools, libraries and technologies that allow you to build a web application.

4.4.PROJECTFLOW:

- User interacts with the UI (User Interface) to upload the input features.
- Uploaded features/input is analysed by the model which is integrated.
- Once a model analyses the uploaded inputs, the prediction is showcased on the UI

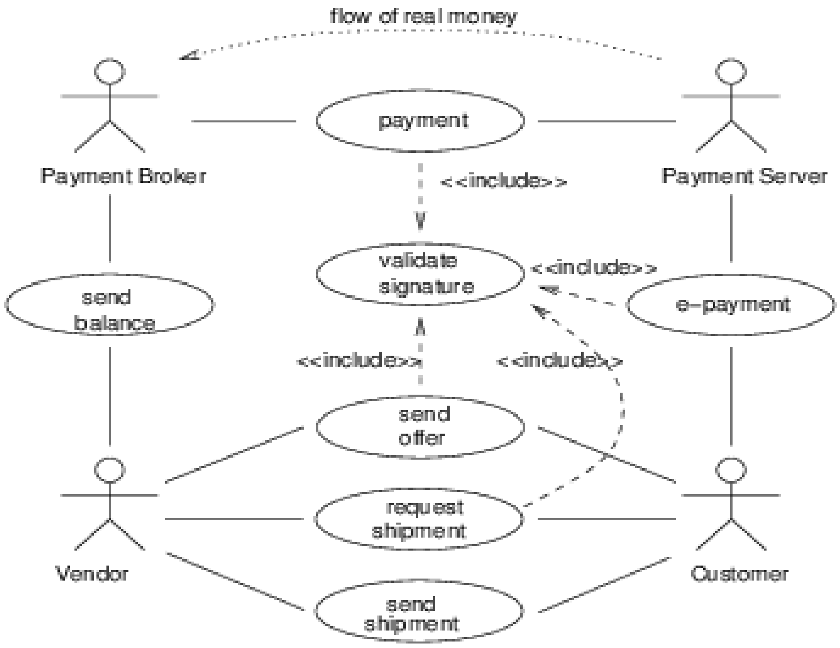
5.DESIGN

5.1.Dataset:

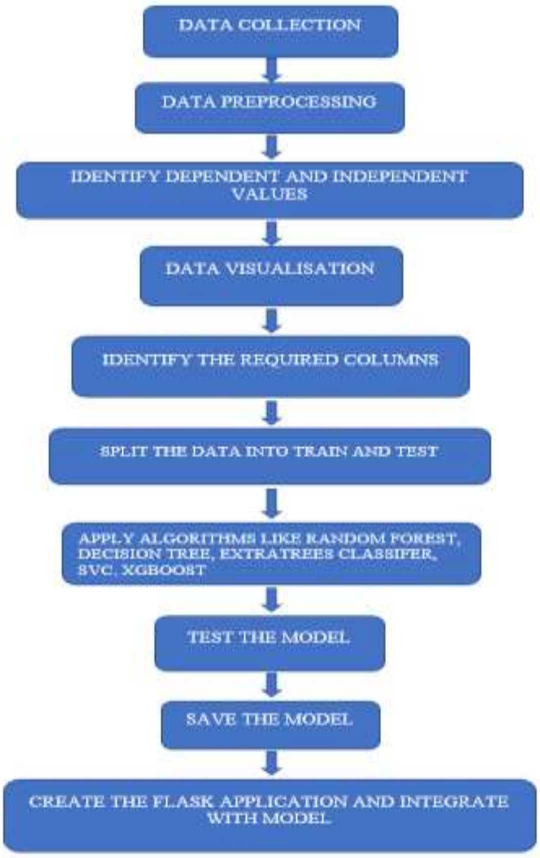
The dataset plays an important role in classifying the model. The dataset has been taken from the official Kaggle website, which is a well-informed data science organization. This dataset has details of millions of transactions out of which some of them are fraud transactions. This makes the development of the system more fluent and reliable. This dataset contains information on the rising risk of digital financial fraud, emphasizing the difficulty in obtaining such data. The main technical challenge it poses to predicting fraud is the highly imbalanced distribution between positive and negative classes in 6 million rows of data. The parameters of this dataset are Transaction type, amount, name Orig, oldbalance Org, newbalance Orig, name Dest, oldbalance Dest, newbalanceDest.

variables	Description	Type
Transaction type	It states the type of the transaction	Categorical
Amount	Transaction amount	Numerical
Name-origin	Senders unique id	ID
Dest-Origin	Receivers unique id	ID
Old-balance-org	Senders balance before transaction	Numercial
New-balance -org	Senders balance after transaction	Numerical
Old-balance-dest	Receivers balance before transaction	Numercial
New-Balance -dest	Receivers balance after transaction	Numerical

5.2.USE CASE DIAGRAM



5.3.FLOWCHART



6.CONCLUSION

In UG Project Phase-1, we have worked on problem statement, literature survey and also done the experimental analyses & design which are required for the project to move forward. In experimental analysis we have discussed about the machine learning concepts and models used in the project. We also discussed about the flowcharts, use case diagram which are used in the project. Based on the experimental analysis we have designed the model for the project. Entire designing part is involved in UG Project Phase-1.

7.FUTURE SCOPE

UG Project Phase-2 is the extension of UG Project Phase-1. 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 will be retrieved in the phase-II. 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.

TABLE OF CONTENTS:

1.INTRODUCTION……………………………………………………….20

2.CODE SNIPPETS……………………………………………………….21-38

2.1 MODEL CODE…………………………………………………21-34

2.2 HTML AND PYTHON CODE………………………………35-38

3.CONCLUSION……………………………………………………….39-41

4.APPLICATIONS……………………………………………………..42

5.ADVANTAGES & DISADVANTAGES…………………………….42-43

6.FUTURE SCOPE………………………………………………………44

7. BIBILOGRAPHY ……………………………………….……………45

8.HELP LINE……………………………………………………………..46

LIST OF FIGURES:

FIGURE 1: .ipynb code importing libraries & mounting dataset from Drive……………21

FIGURE 2: .ipynb code displaying few rows, columns & column names from the dataset………………………………………………………………………………...21

FIGURE 3: .ipynb code describe in detail info using info() method…………………….22

FIGURE 4: .ipynb code for heatmap shows 2 dimensional representation of dataset…...22

FIGURE 5: .ipynb code for univariate analysis of step column…………………………23

FIGURE 6: .ipynb code for different columns present in dataset……………………….24

FIGURE 7:.ipynb code for count of fraud and non fraud transactions………………….25

FIGURE 8: .ipynb code displaying Bi-variate analyasis gives relationship between each variable in dataset……………………………………………………………………….27

FIGURE 9: .ipynb code for descriptive analysis it describes the data………………….27

FIGURE 10: .ipynb code for Data preprocessing, Raw data to processing procedure…28

FIGURE 11: .ipynb code for removing outliers & transformation plot values…………29

FIGURE 12: .ipynb code for transformation plot & graphs……………………………29

FIGURE 13: .ipynb code for object label encoding converts categorical values to numerical……………………………………………………………………………...30

FIGURE 14: .ipynb code splitting data into train and test……………………………...30

FIGURE 15: .ipynb code for Random Forest model…………………………………...31

FIGURE 16: .ipynb code for Decesion tree classifier………………………………….31

FIGURE 17: .ipynb code for extra trees classifier……………………………………..32

FIGURE 18: .ipynb code for support vector machine classifier……………………….32

FIGURE 19: .ipynb code for Label encoding converts categorical columns to numerical columns……………………………………………………………………………….34

FIGURE 20: .ipynb code for xgboost classifier………………………………………..34

FIGURE 21: .ipynb code for comparing the models & accuracy of each model, importing pickle file(.py code)…………………………………………………………………….35

FIGURE 22: .ipynb code for prediction & predicting by giving values…………………35

FIGURE 23: .python code used for rendering all the HTML pages……………………..36

FIGURE 24: home.html page is the code for homepage of our web application………..37

FIGURE 25: predict.html page which predicts the output. By taking the inputs from user………………………………………………………………………………….37-39

FIGURE 26: submit.html is a button when we enter values & click on submit button it displays a message associated with code…………………………………………………...39

FIGURE 27: Home page (which gives introduction to Online payments Fraud Detection)…………………………………………………………………………...39

FIGURE 28: Input page (which takes input from user)………………………………….39

FIGURE 29: Output page (Displays that the payment is fraud)…………………………40

FIGURE 30: Input page (which takes input from user)………………………………….40

FIGURE 31: Output page (Displays that the payment is not fraud)……………………..41

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 the 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.

1. CODE SNIPPETS

1.1 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
```

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

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

```
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)
```

```
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	C827181710	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

```
data.columns
```

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

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.

HEAT MAP

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

```
<ipython-input-40-6c71ac866e2e>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated.
  sns.heatmap(data.corr(),annot=True)
<Axes: >
```

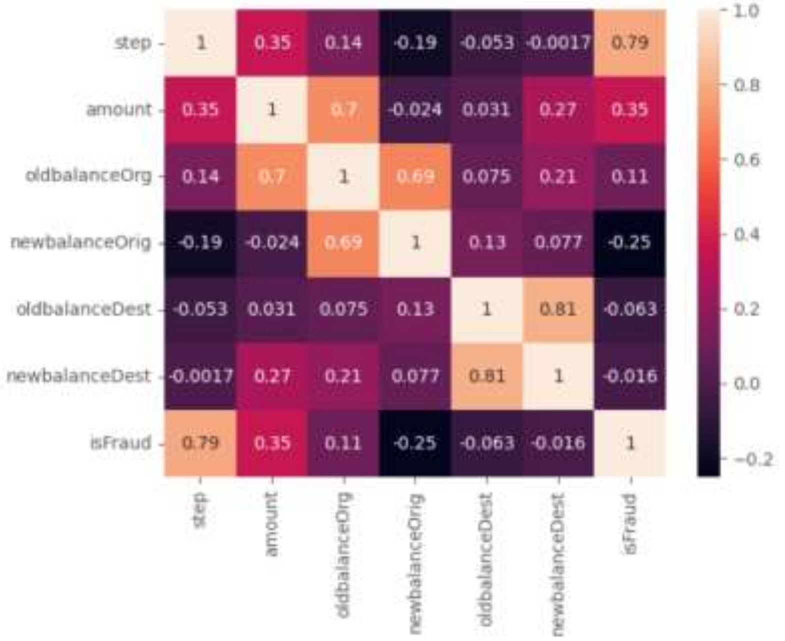


Figure 4: .ipynb code for heatmap shows 2 dimensional representation of dataset.

UNIVARIATE ANALYSIS

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

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

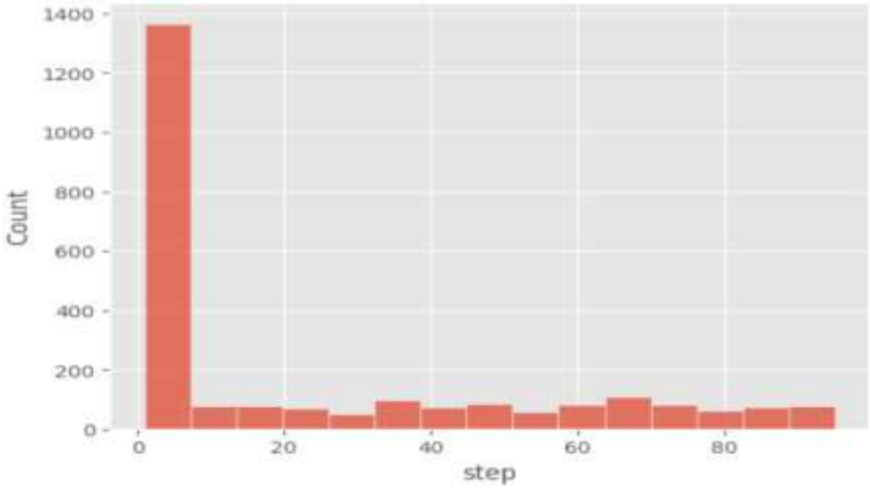
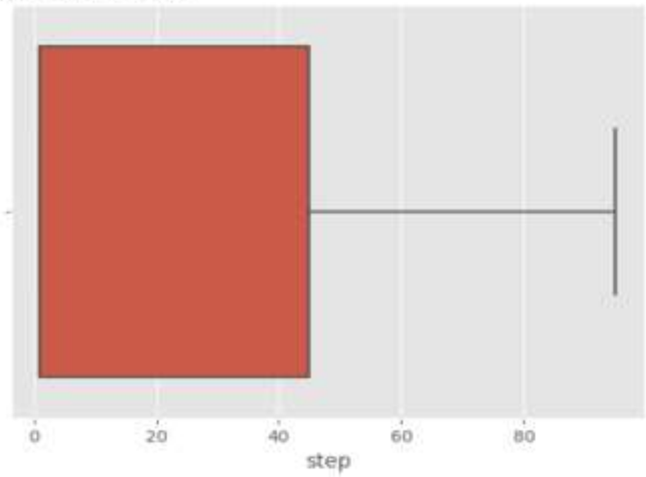


Figure 5: .ipynb code for univariate analysis of step column.

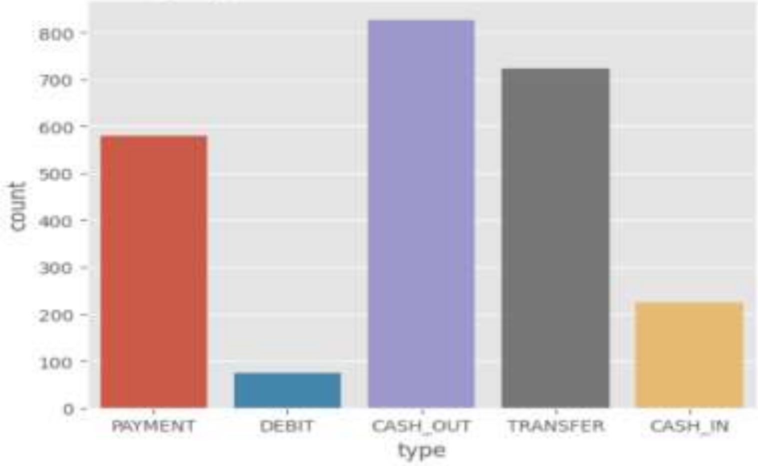
```
sns.boxplot(data=data, x='step')
```

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



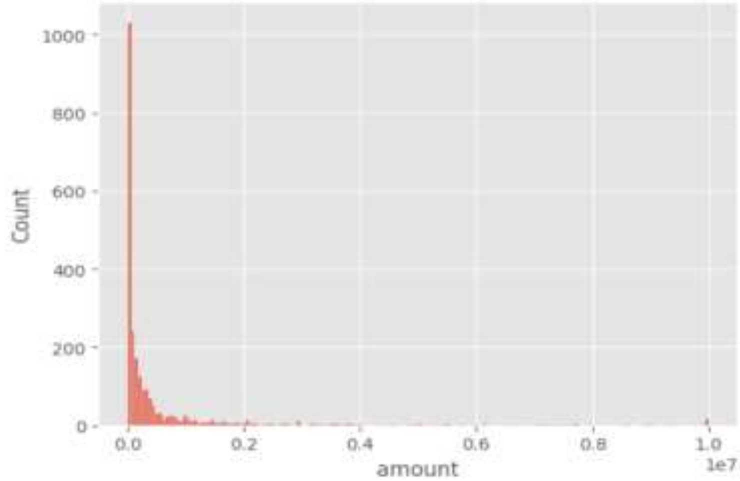
```
sns.countplot(data=data,x='type')
```

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




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

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




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

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

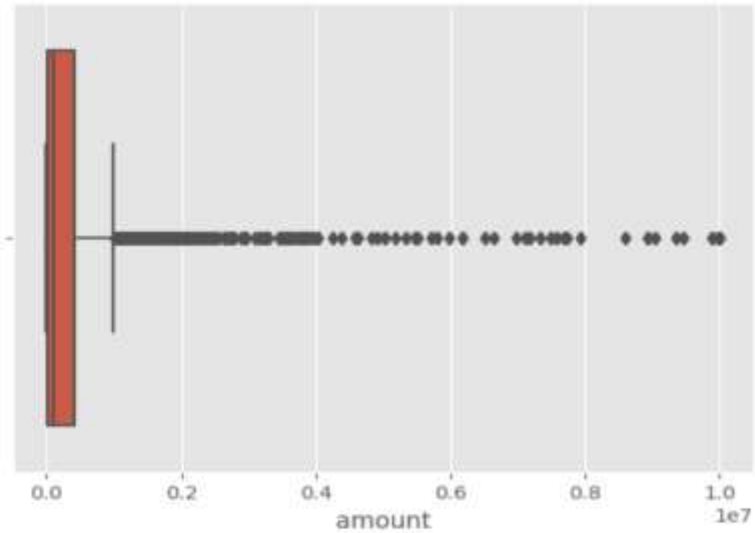


Figure 6: .ipynb code for different columns present in dataset.

```
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

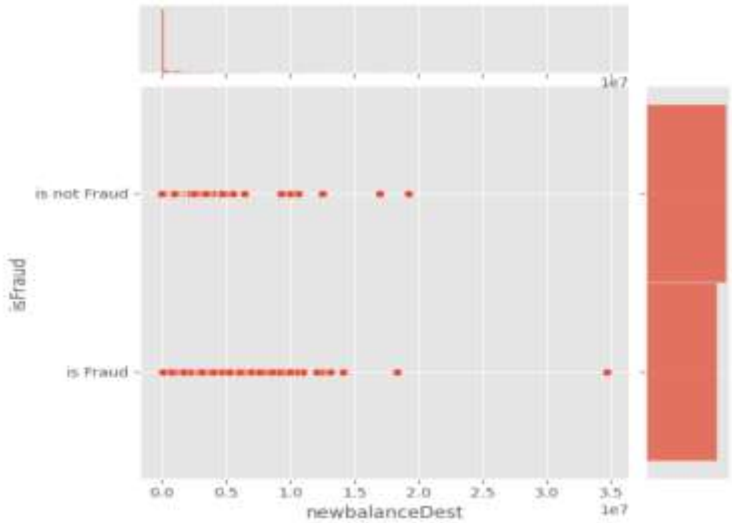
2430 rows × 10 columns

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

Bivariate analysis

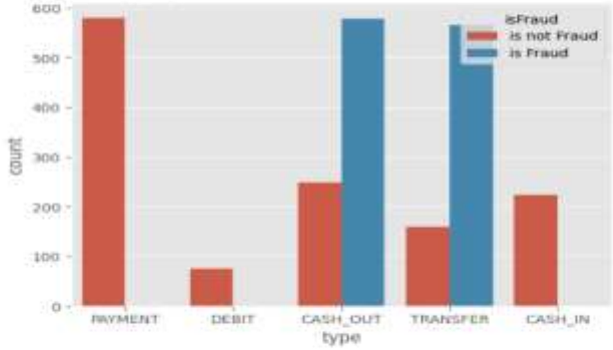
```
sns.jointplot(data=data,x='newbalanceDest',y='isFraud')
```

```
<seaborn.axisgrid.JointGrid at 0x7f49fa5b5ba0>
```



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

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

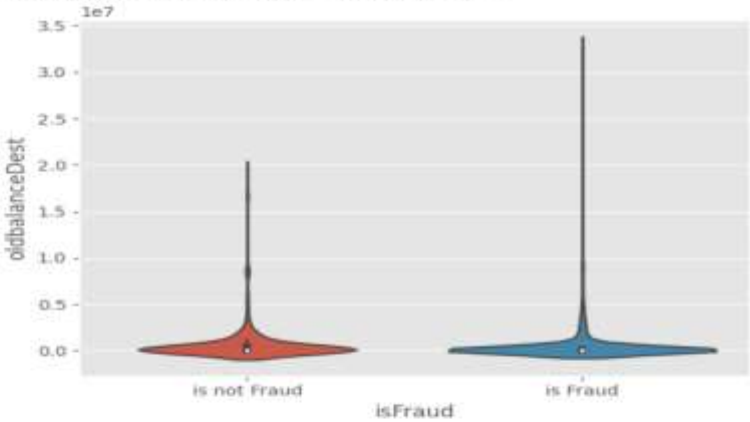



```
sns.boxplot(data=data,x='isFraud',y='step')
```

```
<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'>
```

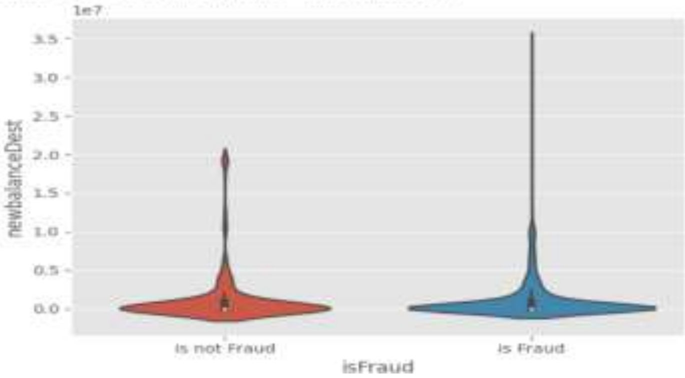


Figure 8: .ipynb code displaying Bi-variate analaysis gives relationship between each variable in dataset.

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	1255
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

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

Data Preprocessing

+ Code + Text

```
[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]))
625836.0974156379
<ipython-input-69-d0edd6481bac>: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)
```

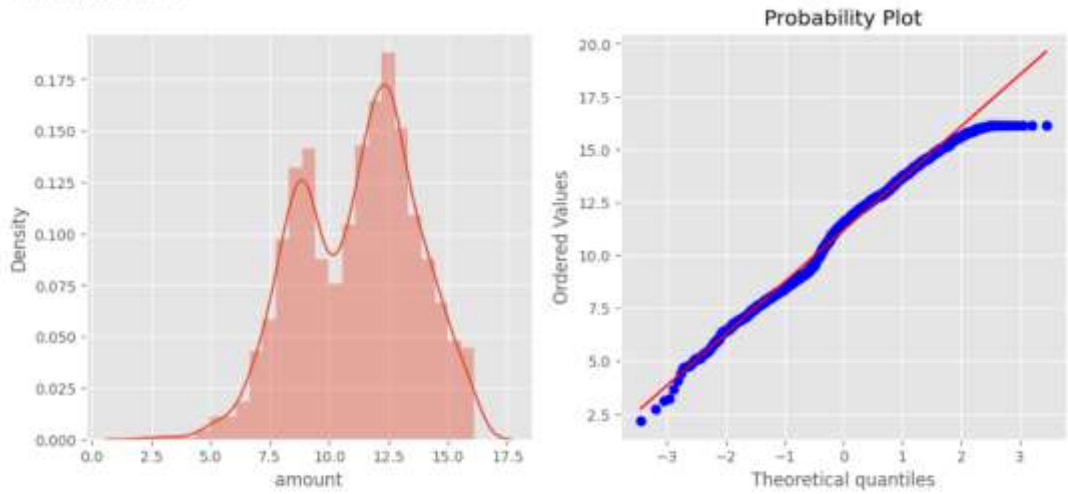


Figure 12: .ipynb code for transformation plot & graphs.

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	is Fraud	is not Fraud
isFraud		
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	is Fraud	is not Fraud
isFraud		
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.

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

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

```
[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])
```

```
[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

```
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
```

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

```
1.0
```

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

col_0	0	1
row_0		
0	233	1
1	0	252

```
[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))
```

+ Code + Text

```
[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.

1.2 HTML CODE AND PYTHON CODE

1. app.py code:

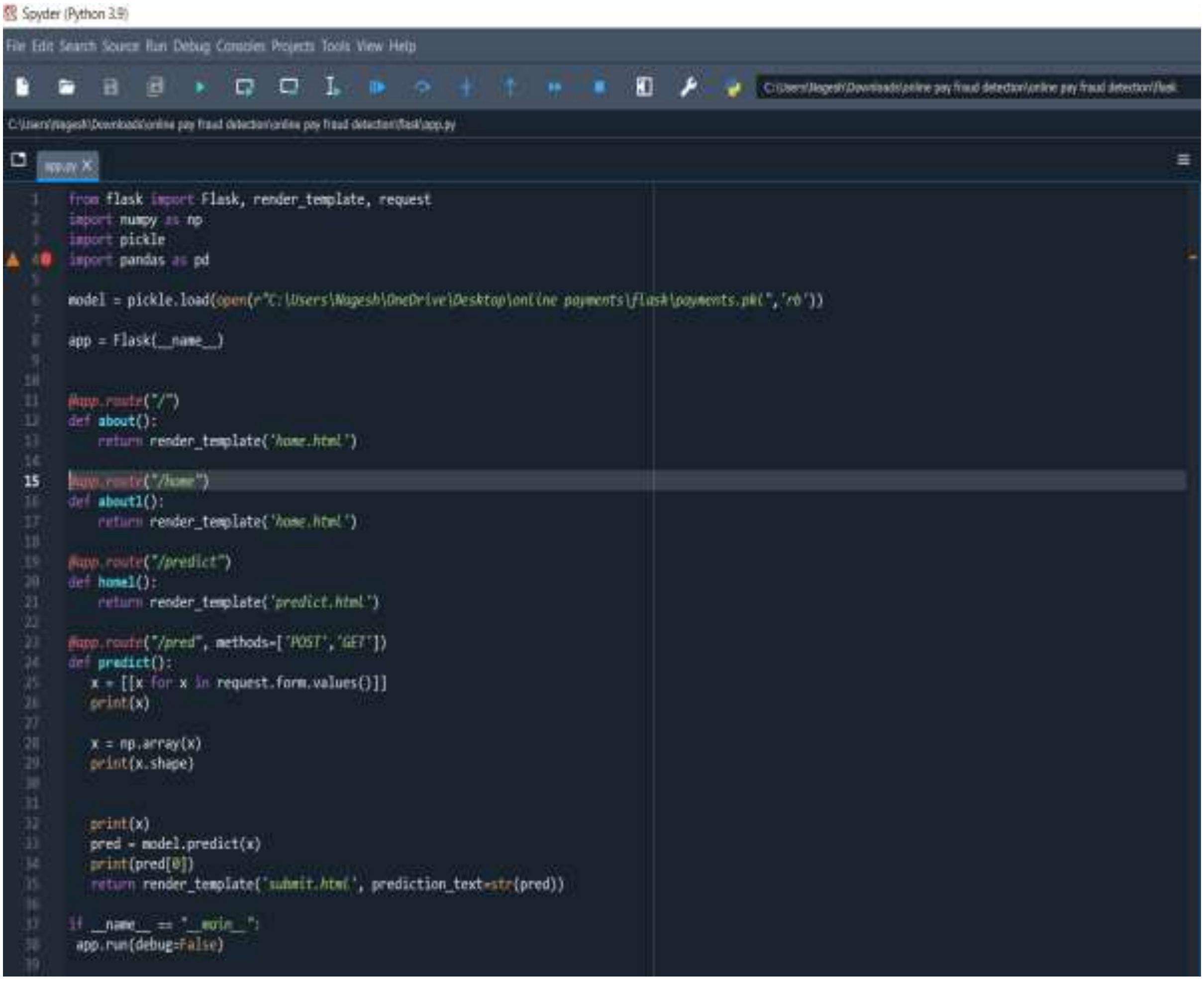


Figure 23: .python code used for rendering all the HTML pages.

2. home.html

```html
<!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("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAUsAAACYCAMAAABatDuZAAABg1BMVEWp4v45tf+q4P+Jt9O33fD///+q4fE
        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 t
 We will be using classification algorithms such as Decision tree, Random forest,  svm, and Extra tree classifier.  We will train and te

         </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>
```

Figure 24: home.html page is the code for homepage of our web application.

3. predict.html:

```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("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAUsAAACYCAMAAABatDuZAAABg1BMVEUqd4v45tf+q4P+Jt9O33fD///+q4fc
        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 whe
                  </div>
              </div>
               <div class="form-group row">
```

```html
                   <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" requi
                   </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="th
                    </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
                    </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"
                     </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="oldbalanceDes
                      </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="newbalanceDe
                       </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 25: predict.html page which predicts the output. By taking the inputs from user.

4. Submit.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 26: submit.html is a button when we enter values & click on submit button it displays a message associated with code.

2. CONCLUSION

Figure 27: Home page (which gives introduction to Online payments Fraud Detection)

Figure 28: Input page (which takes input from user)

Figure 29: Output page (Displays that the payment is fraud)

Figure 30: Input page (which takes input from user)

Figure 31: Output page (Displays that the payment is not fraud)

3. 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).

4. 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.

5. 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.

6. 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-with-machine-learning/
8. https://www.geeksforgeeks.org/online-payment-fraud-detection-using-machine-learning-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.