ONLINE PAYMENTS FRAUD DETECTION USING ML

TEAM ID-592689

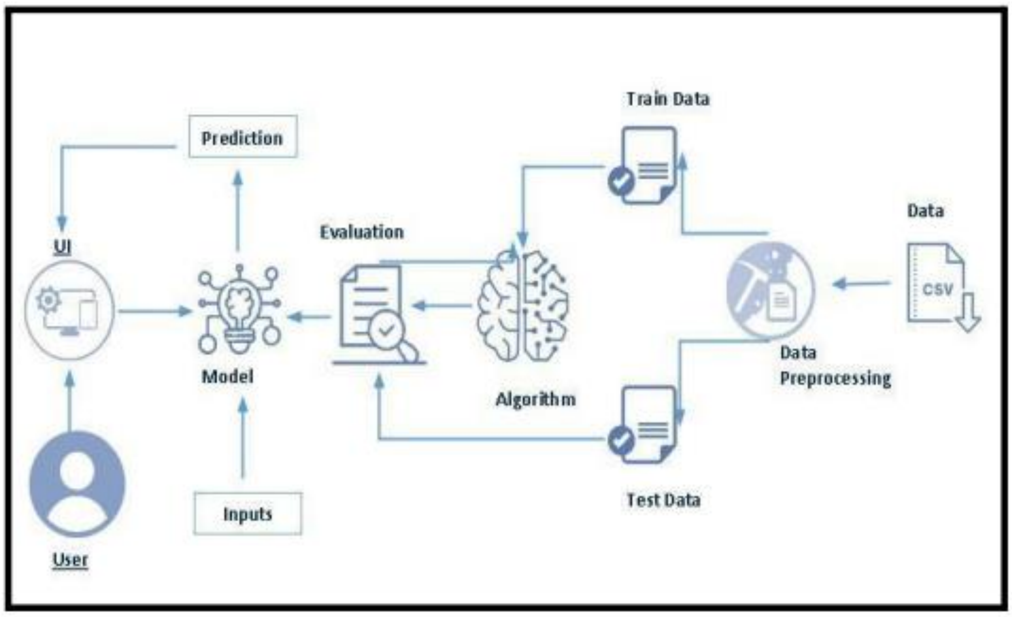
Project Description:

The growth in internet and e-commerce appears to involve the use of online credit/debit card transactions. The increase in the use of credit / debit cards is causing an increase in fraud.

The frauds can be detected through various approaches, yet they lag in their accuracy and its own specific drawbacks. If there are any changes in the conduct of the transaction, the frauds are predicted and taken for further process.

Due to large amount of data credit / debit card fraud detection problem is rectified by the proposed method We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier, xgboost Classifier.We will train and test the data with these algorithms.

TECHNICAL ARCHITECTURE



Pre requisites:

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

packages

● Anaconda navigator and pycharm:

o Refer the link below to download anaconda navigator

o Link: https://youtu.be/1ra4zH2G4o0

● Python packages:

o Open anaconda prompt as administrator

o Type“pip install numpy”and click enter.

o Type“pip install pandas”andclickenter.

o Type“pip install scikit-learn”andclickenter.

o Type”pip install matplotlib”andclickenter.

o Type”pip install scipy”andclickenter.

o Type”pip install pickle-mixin”andclickenter.

o Type”pip install seaborn”andclickenter.

o Type“pipinstallFlask”and click enter.

Prior Knowledge:

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

● ML Concepts

o Supervised learning:

https://www.javatpoint.com/supervised-machine-learning

o Unsupervised learning:

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

o Regression and classification

Decisiontree:

https://www.javatpoint.com/machine-learning-decision-tree-classificatio

n-algorithm

o Randomforest:

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

o xgboost Classifier

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

arning

o Svm:

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

o Extra tree classifier:

Evaluationmetrics:

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-eva luation-error-metrics/ o Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt

PROJECT OBJECTIVES

By the end of this project you will:

● Know fundamental concepts and techniques used for machine learning.

● Gain a broad understanding about data.

● Have knowledge on pre-processing the data/transformation techniques on

outlier and some visualisation concepts.

Project Flow:

● User interacts with the UI to enter the input.

● Entered input is analysed by the model which is integrated.

● Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

● Data collection

o Collect the dataset or create the dataset

● Visualising and analysing data

Importing the libraries

o Read the Dataset

o Univariate analysis

o Bivariate analysis

o Descriptive analysis

● Data pre-processing

o Checking for null values

o Handling outlier

o Handling categorical(object) data

o Splitting data into train and test

● Model building

o Import the model building libraries

o Initialising the model

o Training and testing the model

o Evaluating performance of model

o Save the model

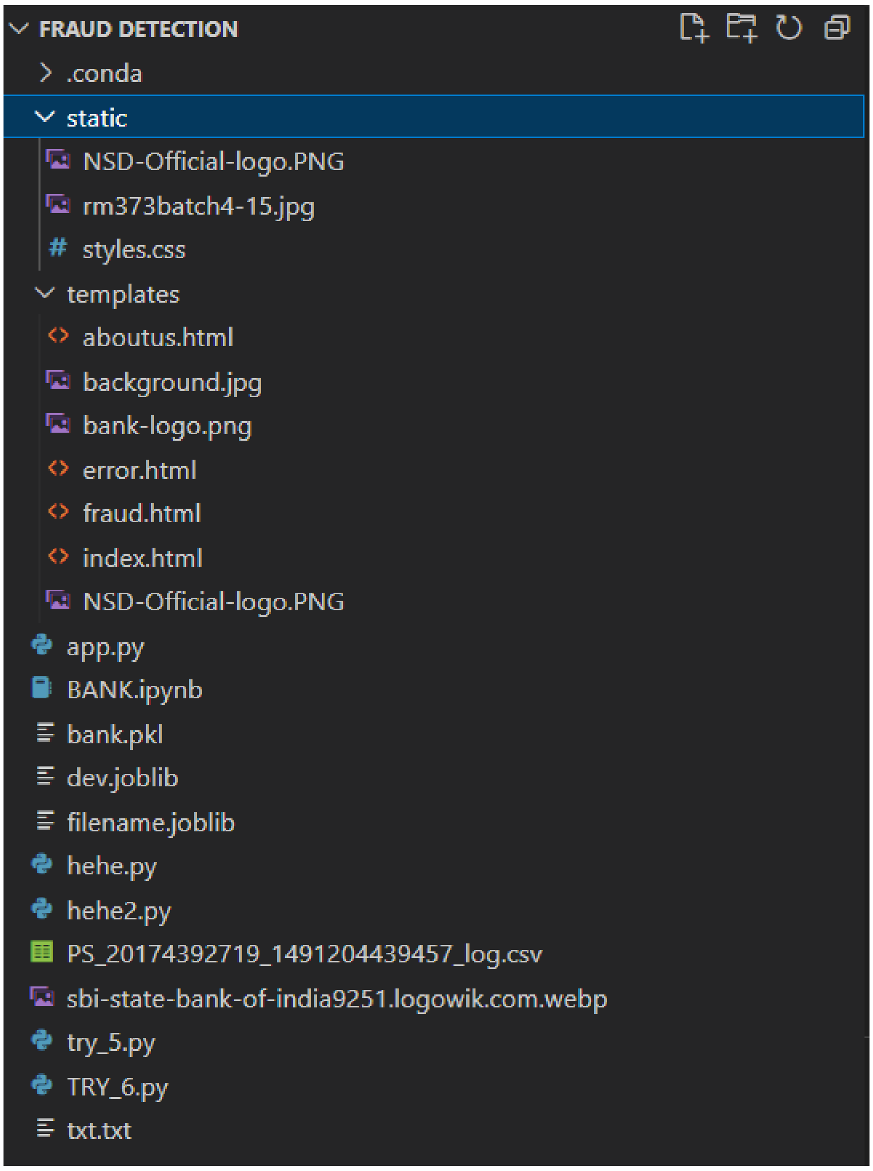
● Application Building

o Create an HTML file

o Build python code

Project Structure:

Create the Project folder which contains files as shown below



```
FRAUD DETECTION
  > .conda
  v static
      NSD-Official-logo.PNG
      rm373batch4-15.jpg
      styles.css
  v templates
      aboutus.html
      background.jpg
      bank-logo.png
      error.html
      fraud.html
      index.html
      NSD-Official-logo.PNG
    app.py
    BANK.ipynb
    bank.pkl
    dev.joblib
    filename.joblib
    hehe.py
    hehe2.py
    PS_20174392719_1491204439457_log.csv
    sbi-state-bank-of-india9251.logowik.com.webp
    try_5.py
    TRY_6.py
    txt.txt
```

Milestone 1: Data Collection

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Collect the dataset or create the dataset or Download the dataset:

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used PS_20174392719_1491204439457_logs.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

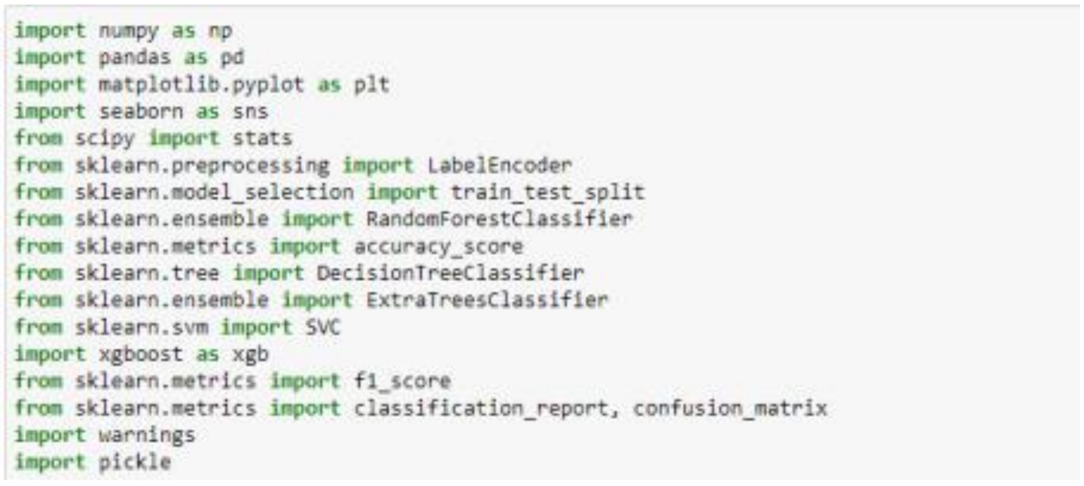
Link: https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-datase

Milestone 2: Visualising and analysing data

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

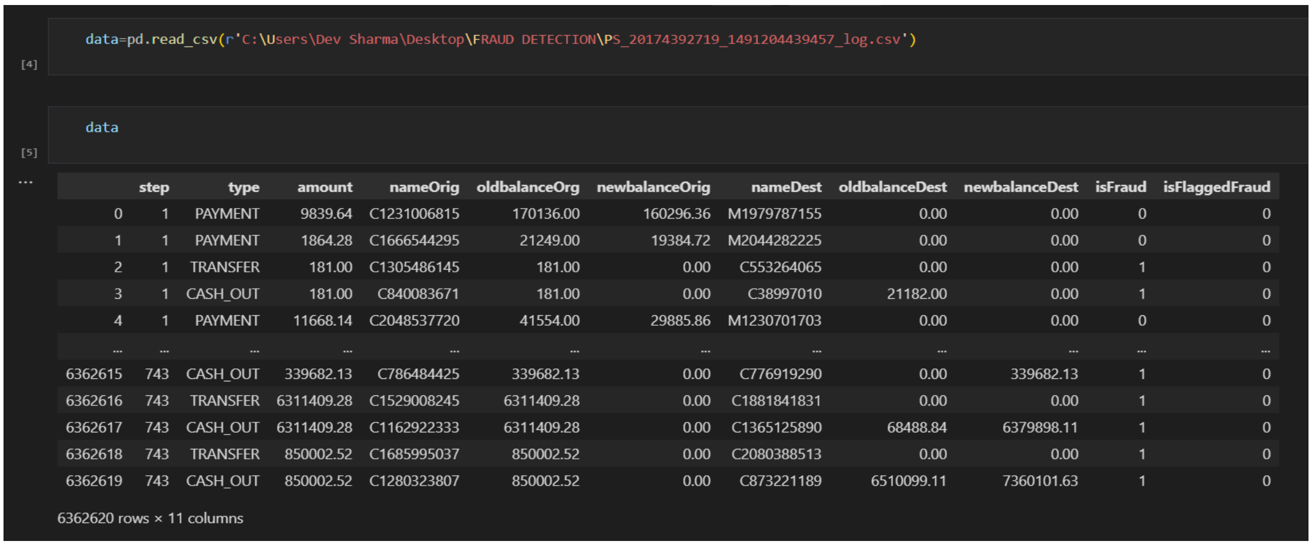
Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries



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

Activity 2: Read the Dataset

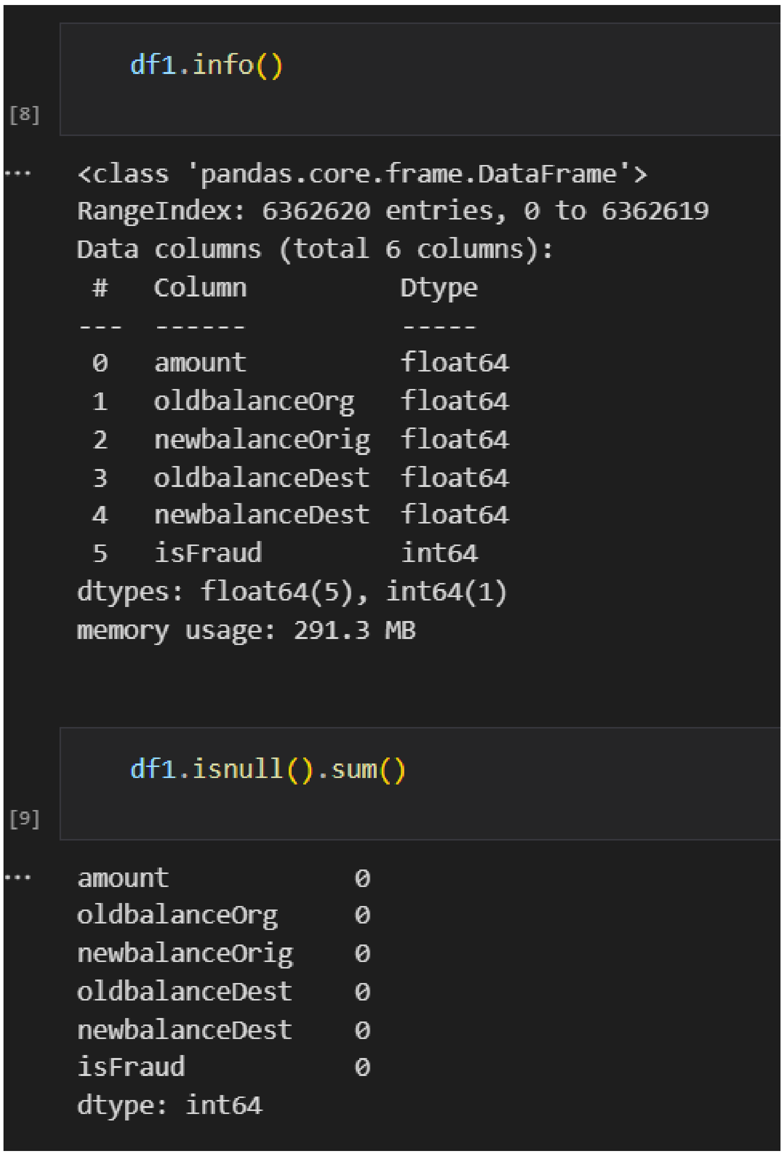


```
[4]  data=pd.read_csv(r'C:\Users\Dev Sharma\Desktop\FRAUD DETECTION\PS_20174392719_1491204439457_log.csv')
```

```
[5]  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	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1	0

6362620 rows × 11 columns



```
[8]  df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   amount          float64
 1   oldbalanceOrg   float64
 2   newbalanceOrig  float64
 3   oldbalanceDest  float64
 4   newbalanceDest  float64
 5   isFraud         int64
dtypes: float64(5), int64(1)
memory usage: 291.3 MB
```

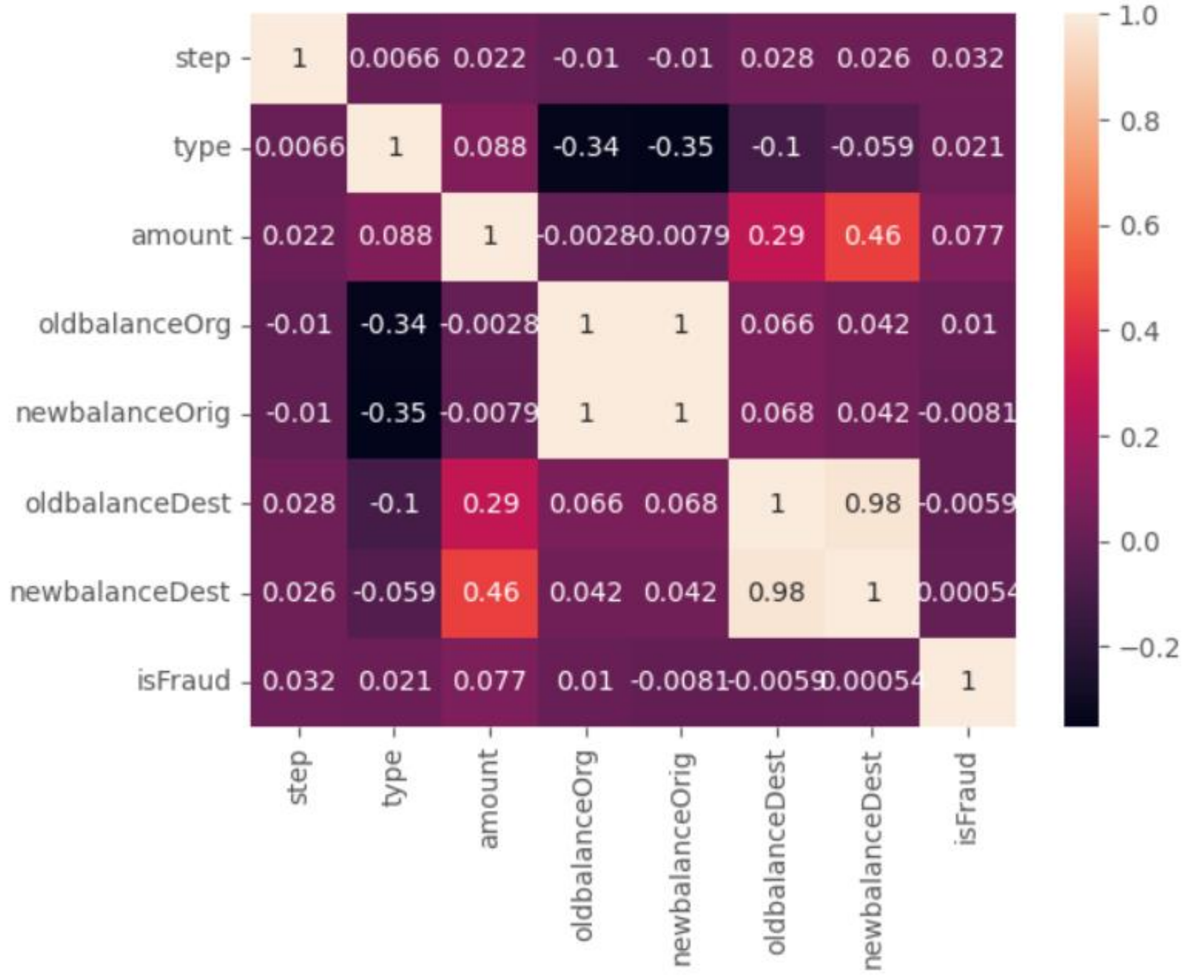
```
[9]  df1.isnull().sum()
```

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

Here, the dataset's superfluous columns are being removed using the drop method

```
df1=df[['amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newbalanceDest',"isFraud"]]
```

ACTIVITY 3 : HEATMAP

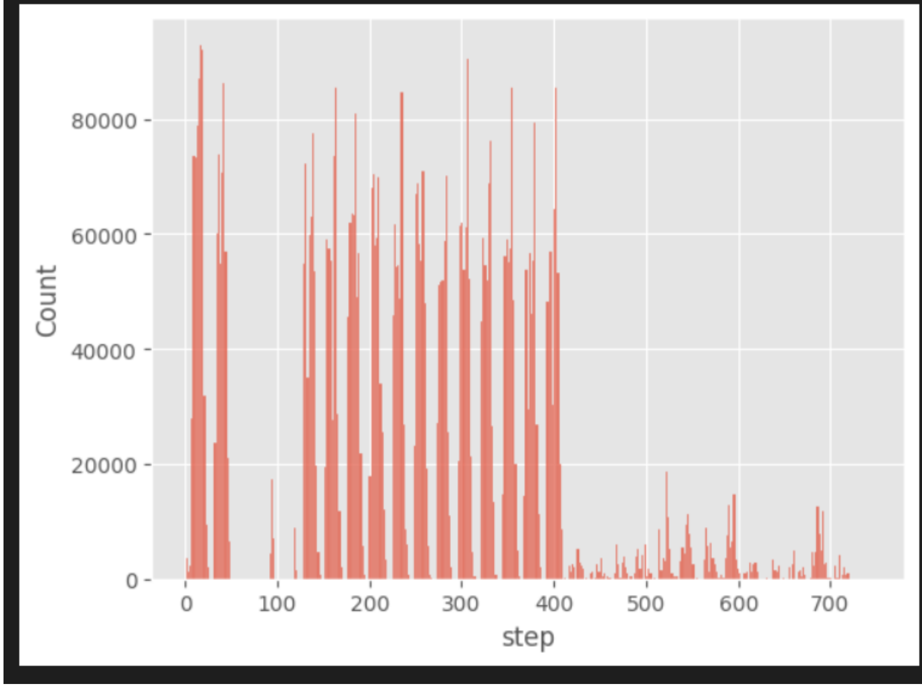


ACTIVITY 4 : UNIVARIATE ANALYSIS

Univariate analysis In simple words, univariate analysis is understanding the data with a single feature. Here I have displayed the graph such as histplot

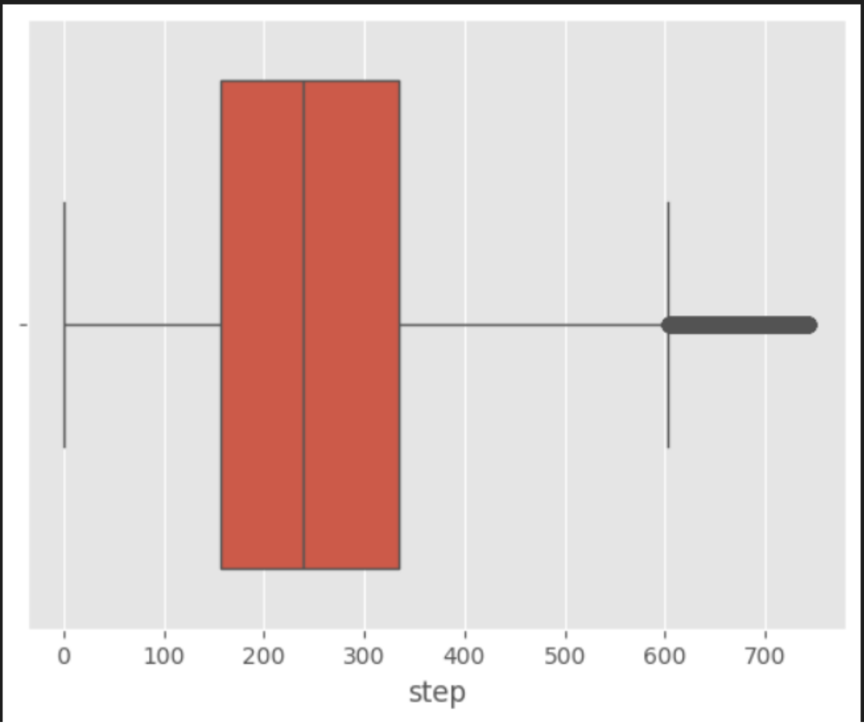
```
sns.histplot(data=df,x='step')
```

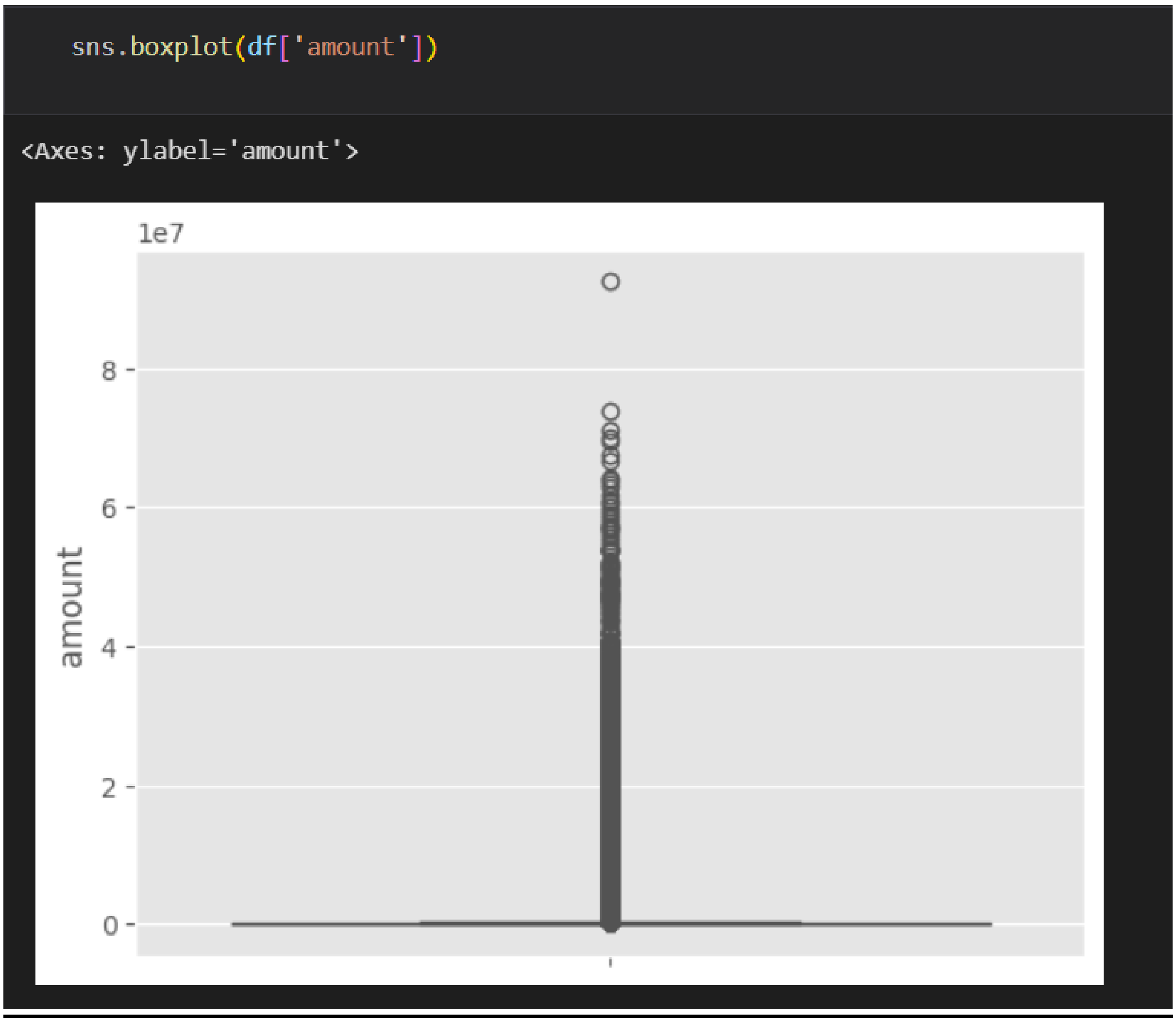
```
<Axes: xlabel='step', ylabel='Count'>
```



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

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





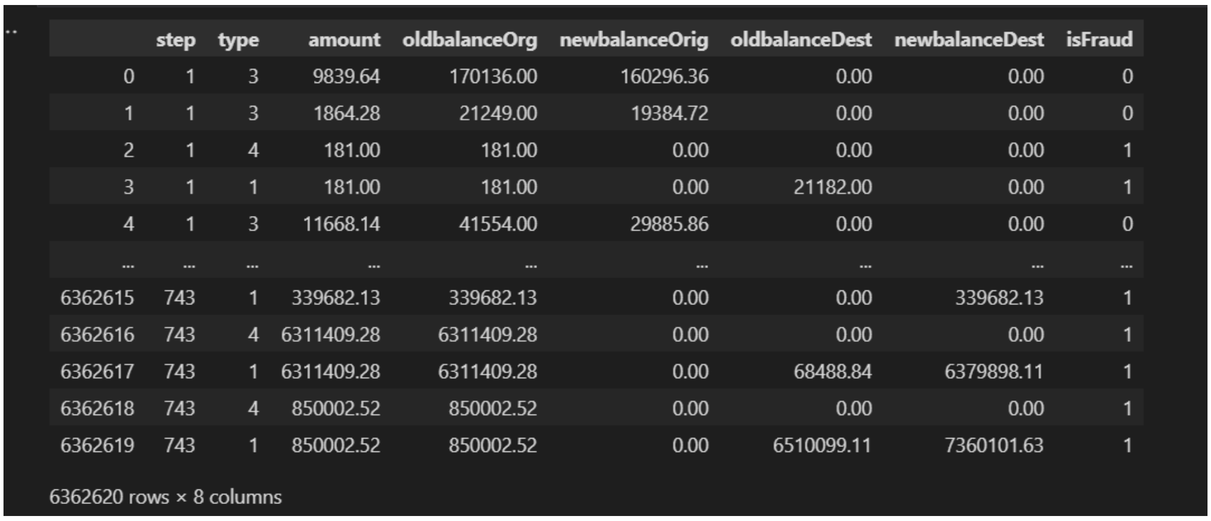
```
sns.boxplot(df['amount'])
```

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

Here, the relationship between the amount attribute and the boxplot is visualise

DATA PROCESSING FOR MODEL BUILDING

DATA :



	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	3	9839.64	170136.00	160296.36	0.00	0.00	0
1	1	3	1864.28	21249.00	19384.72	0.00	0.00	0
2	1	4	181.00	181.00	0.00	0.00	0.00	1
3	1	1	181.00	181.00	0.00	21182.00	0.00	1
4	1	3	11668.14	41554.00	29885.86	0.00	0.00	0
...	...	...	...	...	...	...	...	...
6362615	743	1	339682.13	339682.13	0.00	0.00	339682.13	1
6362616	743	4	6311409.28	6311409.28	0.00	0.00	0.00	1
6362617	743	1	6311409.28	6311409.28	0.00	68488.84	6379898.11	1
6362618	743	4	850002.52	850002.52	0.00	0.00	0.00	1
6362619	743	1	850002.52	850002.52	0.00	6510099.11	7360101.63	1

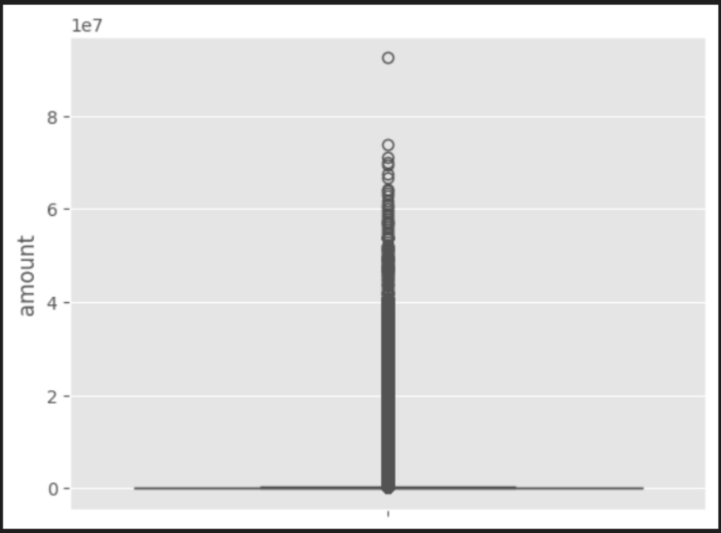
6362620 rows × 8 columns

HANDLING THE OUTLIERS

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

✓ 14.7s

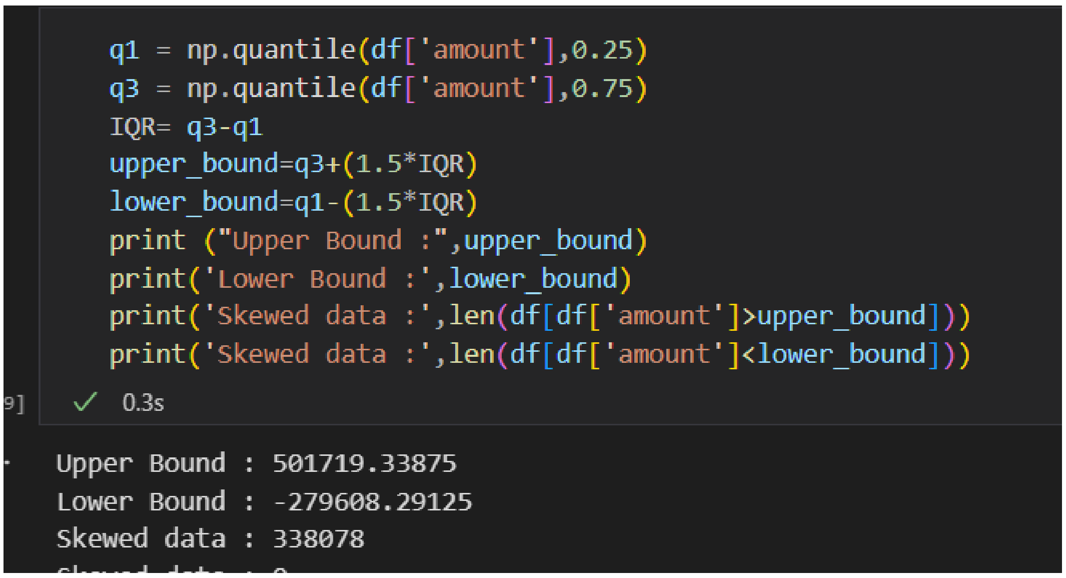
```
<Axes: ylabel='amount'>
```



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

✓ 0.8s

```
ModeResult(mode=10000000.0, count=3207)
179861.90354913071
```

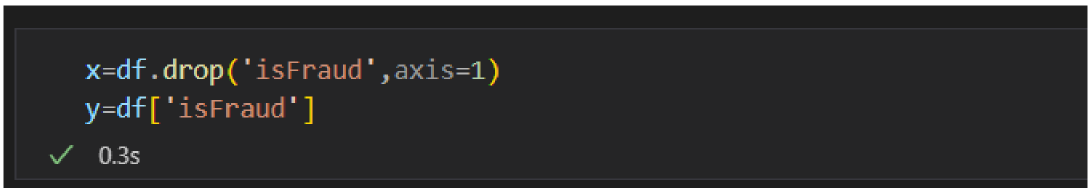


```python
q1 = np.quantile(df['amount'],0.25)
q3 = np.quantile(df['amount'],0.75)
IQR= q3-q1
upper_bound=q3+(1.5*IQR)
lower_bound=q1-(1.5*IQR)
print ("Upper Bound :",upper_bound)
print('Lower Bound :',lower_bound)
print('Skewed data :',len(df[df['amount']>upper_bound]))
print('Skewed data :',len(df[df['amount']<lower_bound]))
```

✓ 0.3s

```
Upper Bound : 501719.33875
Lower Bound : -279608.29125
Skewed data : 338078
```

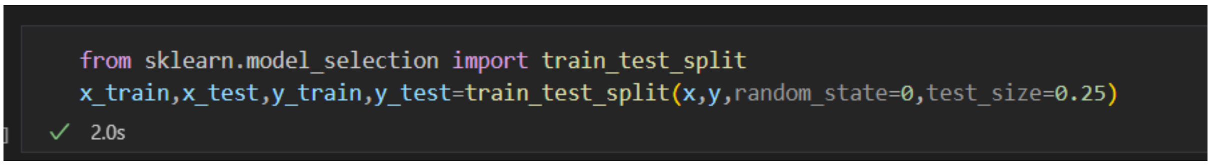
SPLITTING INTO X AND Y



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

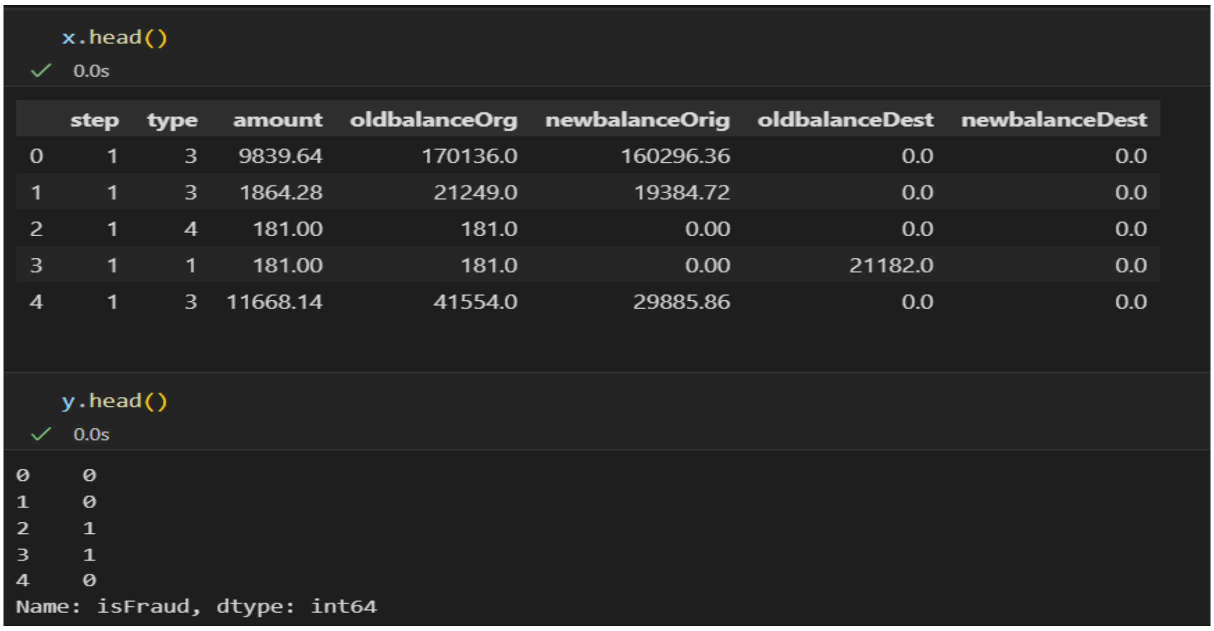
✓ 0.3s

TRAIN TEST SPLIT:



```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.25)
```

✓ 2.0s



```python
x.head()
```

✓ 0.0s

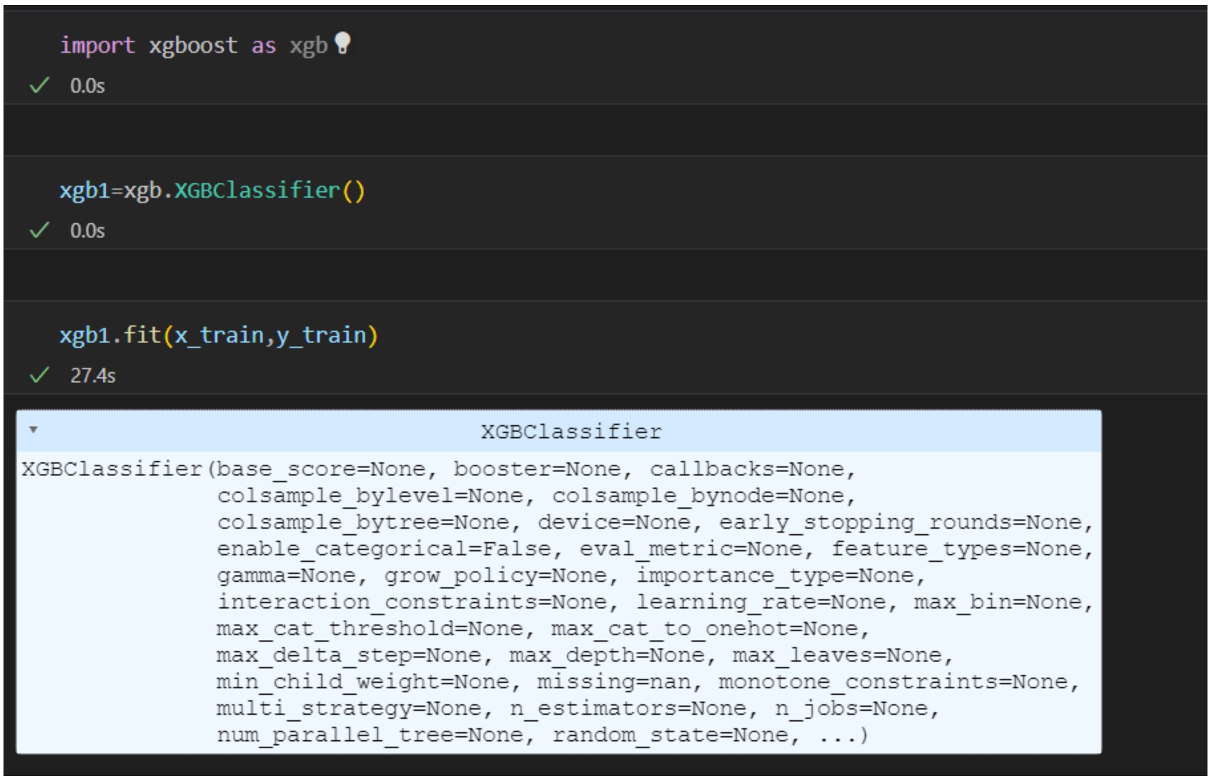
	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	1	3	9839.64	170136.0	160296.36	0.0	0.0
1	1	3	1864.28	21249.0	19384.72	0.0	0.0
2	1	4	181.00	181.0	0.00	0.0	0.0
3	1	1	181.00	181.0	0.00	21182.0	0.0
4	1	3	11668.14	41554.0	29885.86	0.0	0.0

```python
y.head()
```

✓ 0.0s

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

BY evaluating all the classifiers the best Accuracy is getting by used XGboost Classifier

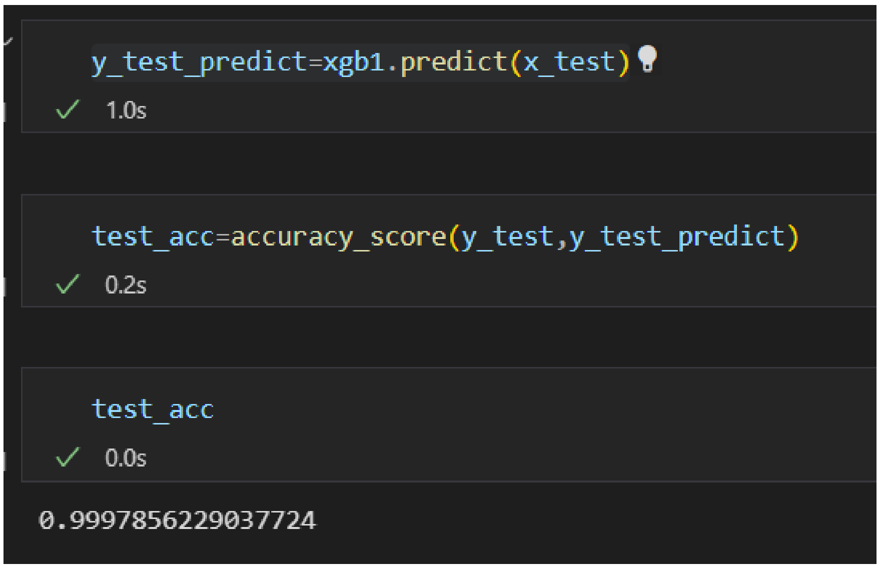


```
import xgboost as xgb
```
✓ 0.0s

```
xgb1=xgb.XGBClassifier()
```
✓ 0.0s

```
xgb1.fit(x_train,y_train)
```
✓ 27.4s

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



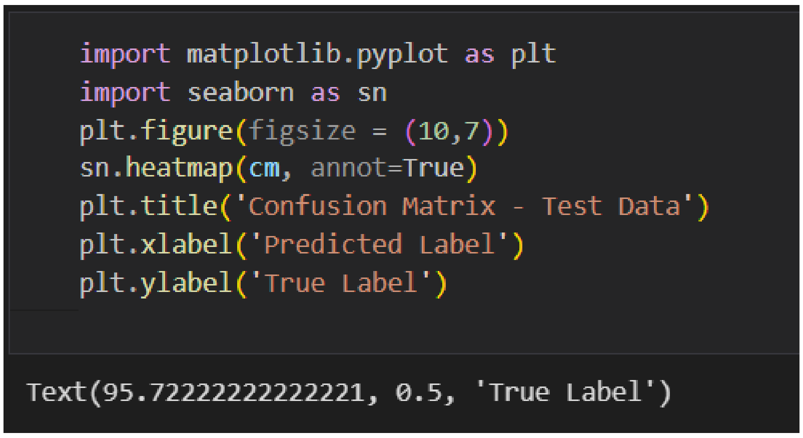
```
y_test_predict=xgb1.predict(x_test)
```
✓ 1.0s

```
test_acc=accuracy_score(y_test,y_test_predict)
```
✓ 0.2s

```
test_acc
```
✓ 0.0s

```
0.9997856229037724
```

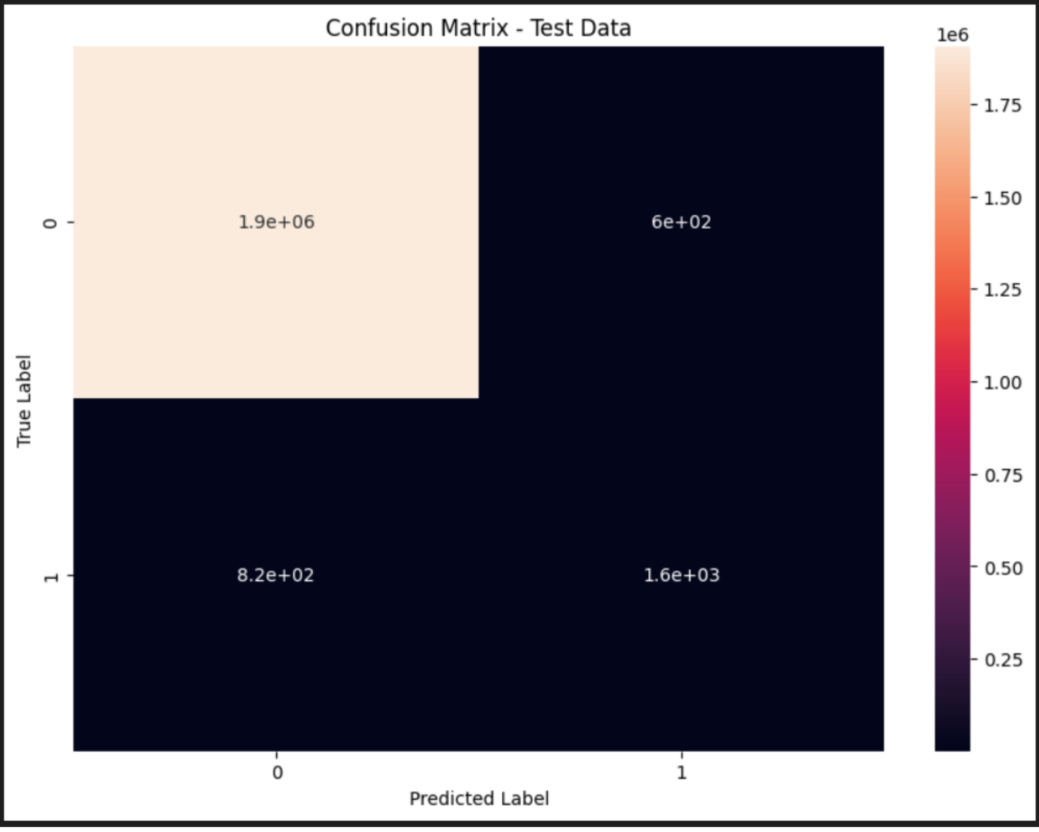
SAVING THE MODEL

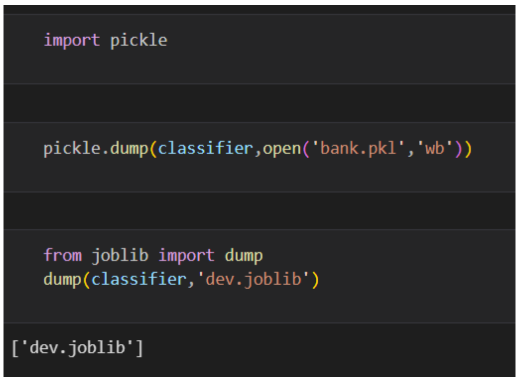


```
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True)
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
```

```
Text(95.72222222222221, 0.5, 'True Label')
```

```
Text(95.72222222222221, 0.5, 'True Label')
```





```
import pickle
```

```
pickle.dump(classifier,open('bank.pkl','wb'))
```

```
from joblib import dump
dump(classifier,'dev.joblib')
```

```
['dev.joblib']
```

Milestone 5: Application Building

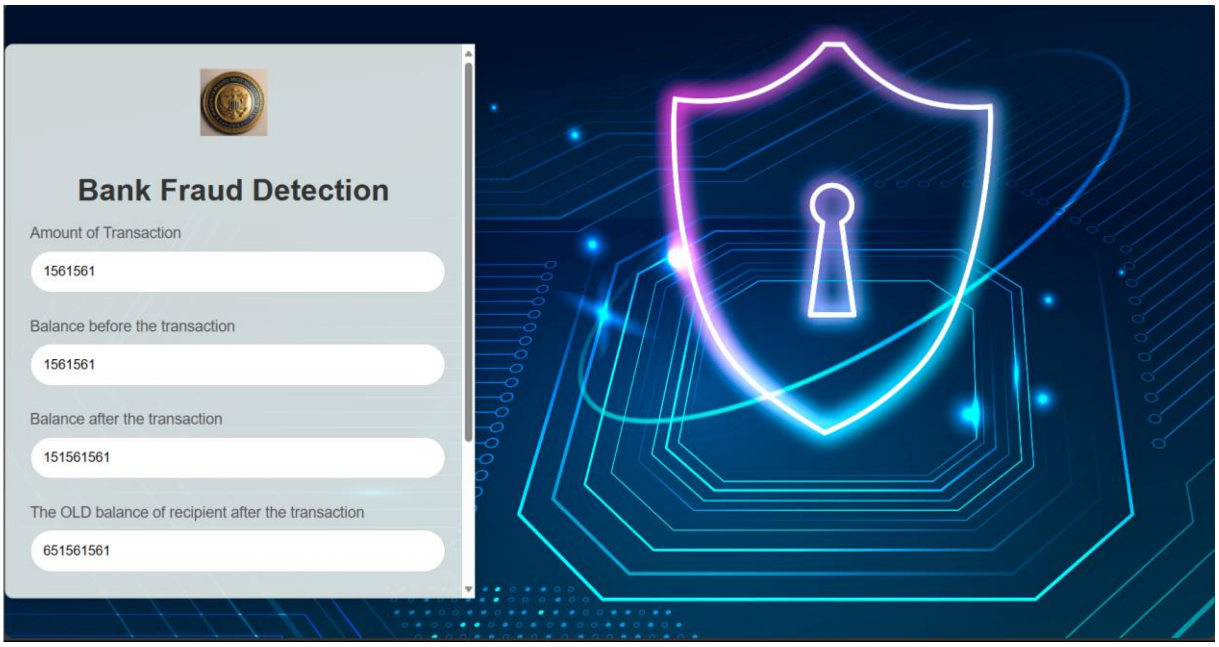
In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

Building HTML Pages
Building server side script
Activity1: Building Html Pages:

For this project create three HTML files namely
● index.html
● fraud.html
● error.html

and save them in the templates folder.
Let's see how our home.html page looks like:

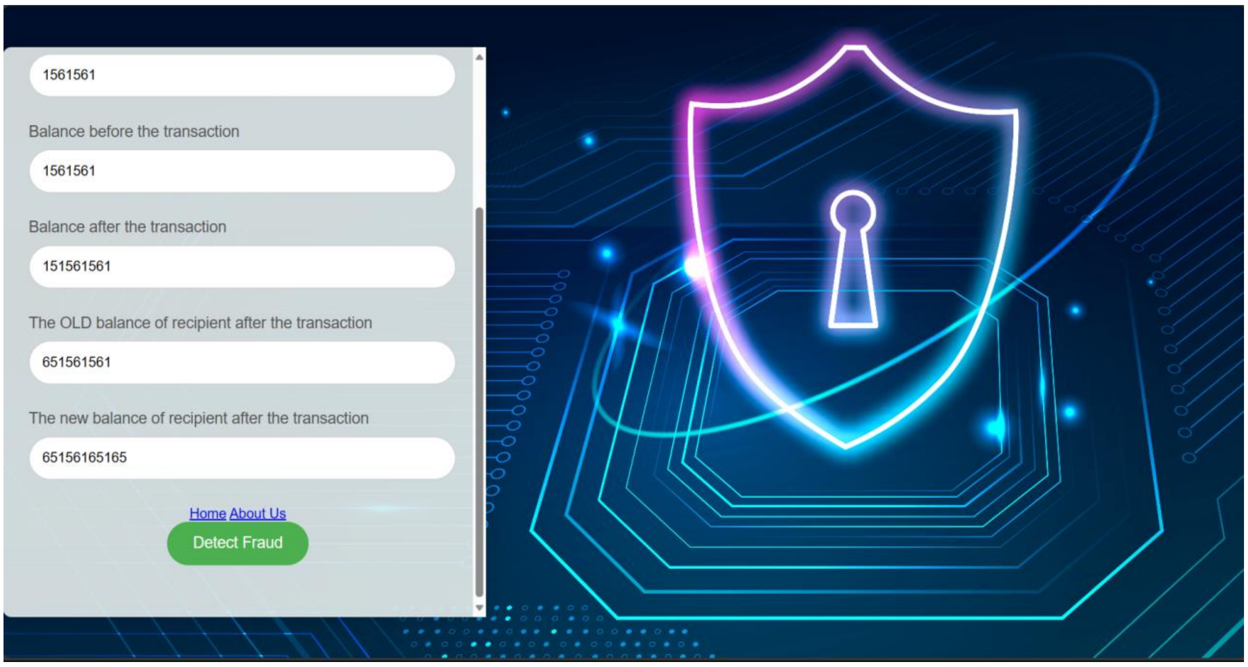

Bank Fraud Detection

Amount of Transaction
1561561

Balance before the transaction
1561561

Balance after the transaction
151561561

The OLD balance of recipient after the transaction
651561561



1561561

Balance before the transaction

1561561

Balance after the transaction

151561561

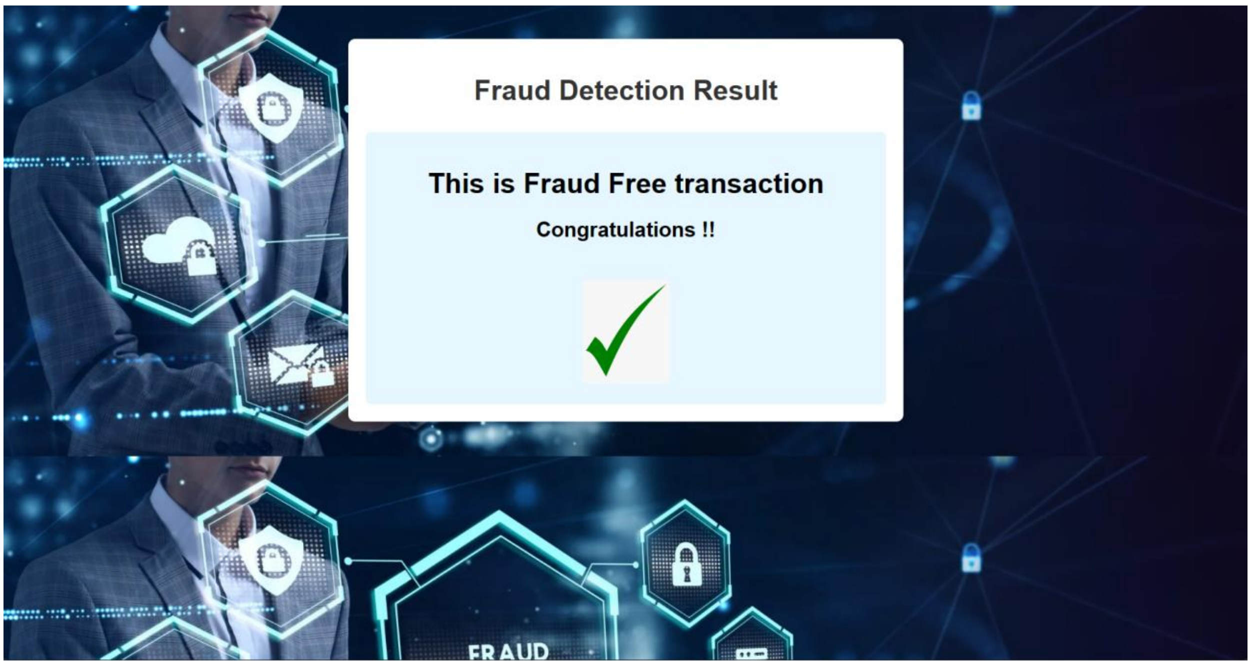
The OLD balance of recipient after the transaction

651561561

The new balance of recipient after the transaction

65156165165

FRAUD HTML PAGE :

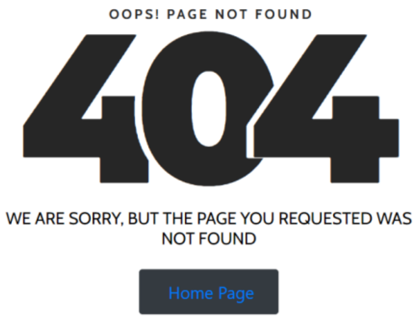


Fraud Detection Result

This is Fraud Free transaction

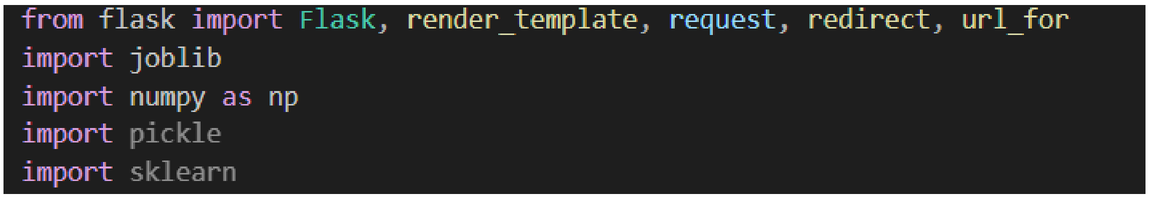
Congratulations !!

ERROR HTML PAGE ::



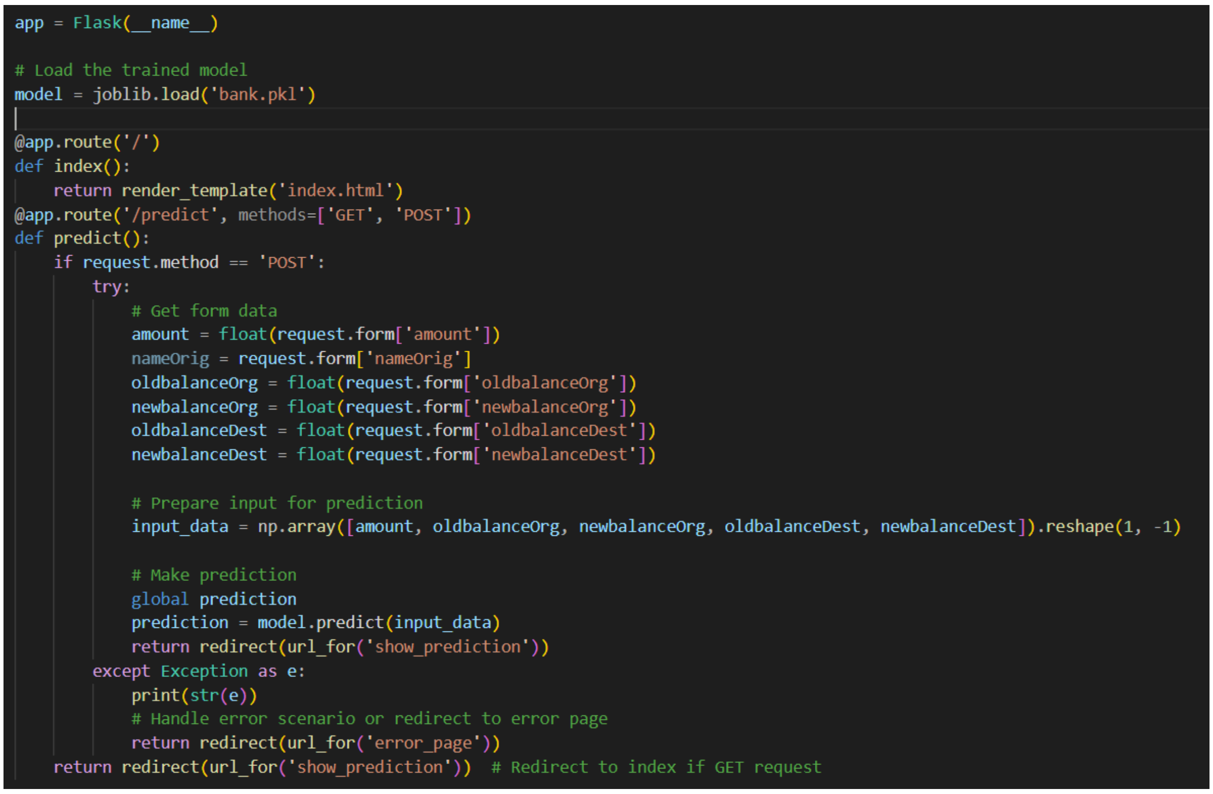
FLASK CODE :

IMPORTING NECESSARY LIBRARIES



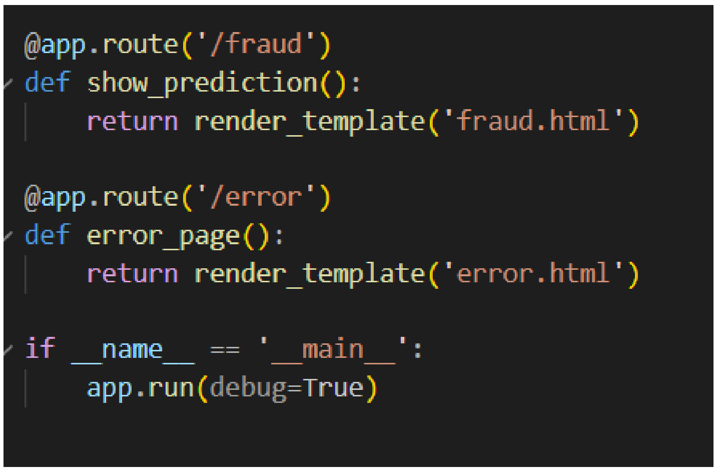
```python
from flask import Flask, render_template, request, redirect, url_for
import joblib
import numpy as np
import pickle
import sklearn
```

MAIN CODE :



```python
app = Flask(__name__)

# Load the trained model
model = joblib.load('bank.pkl')

@app.route('/')
def index():
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        try:
            # Get form data
            amount = float(request.form['amount'])
            nameOrig = request.form['nameOrig']
            oldbalanceOrg = float(request.form['oldbalanceOrg'])
            newbalanceOrg = float(request.form['newbalanceOrg'])
            oldbalanceDest = float(request.form['oldbalanceDest'])
            newbalanceDest = float(request.form['newbalanceDest'])

            # Prepare input for prediction
            input_data = np.array([amount, oldbalanceOrg, newbalanceOrg, oldbalanceDest, newbalanceDest]).reshape(1, -1)

            # Make prediction
            global prediction
            prediction = model.predict(input_data)
            return redirect(url_for('show_prediction'))
        except Exception as e:
            print(str(e))
            # Handle error scenario or redirect to error page
            return redirect(url_for('error_page'))
    return redirect(url_for('show_prediction'))  # Redirect to index if GET request
```

ROUTING :

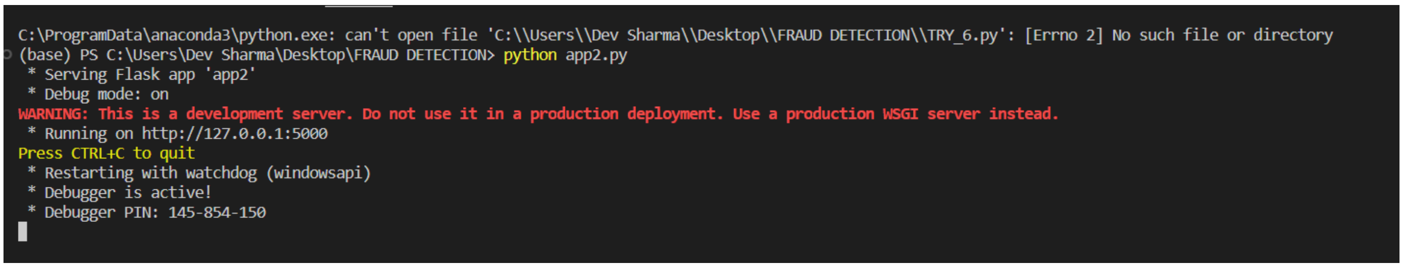


```python
@app.route('/fraud')
def show_prediction():
    return render_template('fraud.html')

@app.route('/error')
def error_page():
    return render_template('error.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.
In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app2.py" command
- Navigate to the localhost where you can view your web page.
- Click on the detect fraud button see the result/prediction on the web on next fraud html page



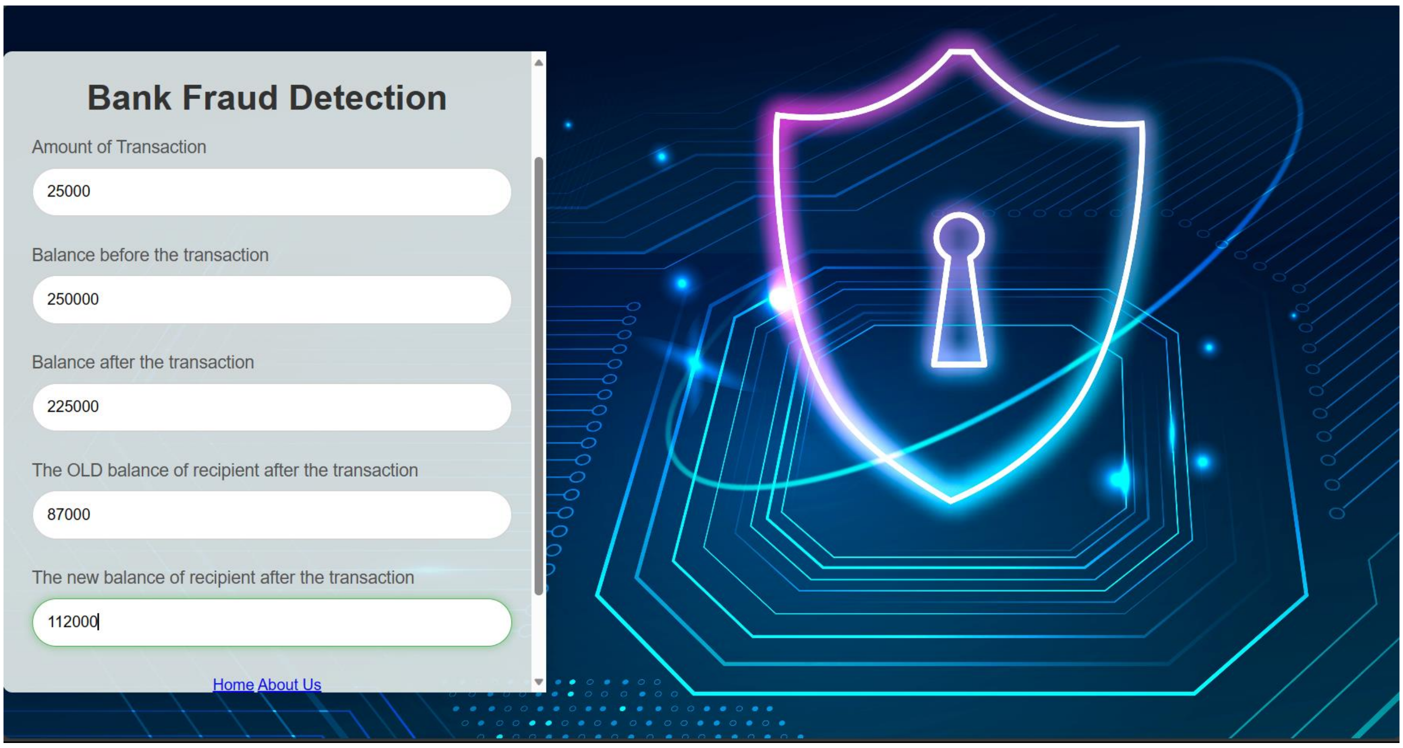
```
C:\ProgramData\anaconda3\python.exe: can't open file 'C:\\Users\\Dev Sharma\\Desktop\\FRAUD DETECTION\\TRY_6.py': [Errno 2] No such file or directory
(base) PS C:\Users\Dev Sharma\Desktop\FRAUD DETECTION> python app2.py
 * Serving Flask app 'app2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 145-854-150
```

OUTPUT SCREENSHOTS ::

ATTEMPT 1

Here we will provide correct details of transaction and accurate deduction of amount for checking

(AFFIRMATIVE CHECKING)



Bank Fraud Detection

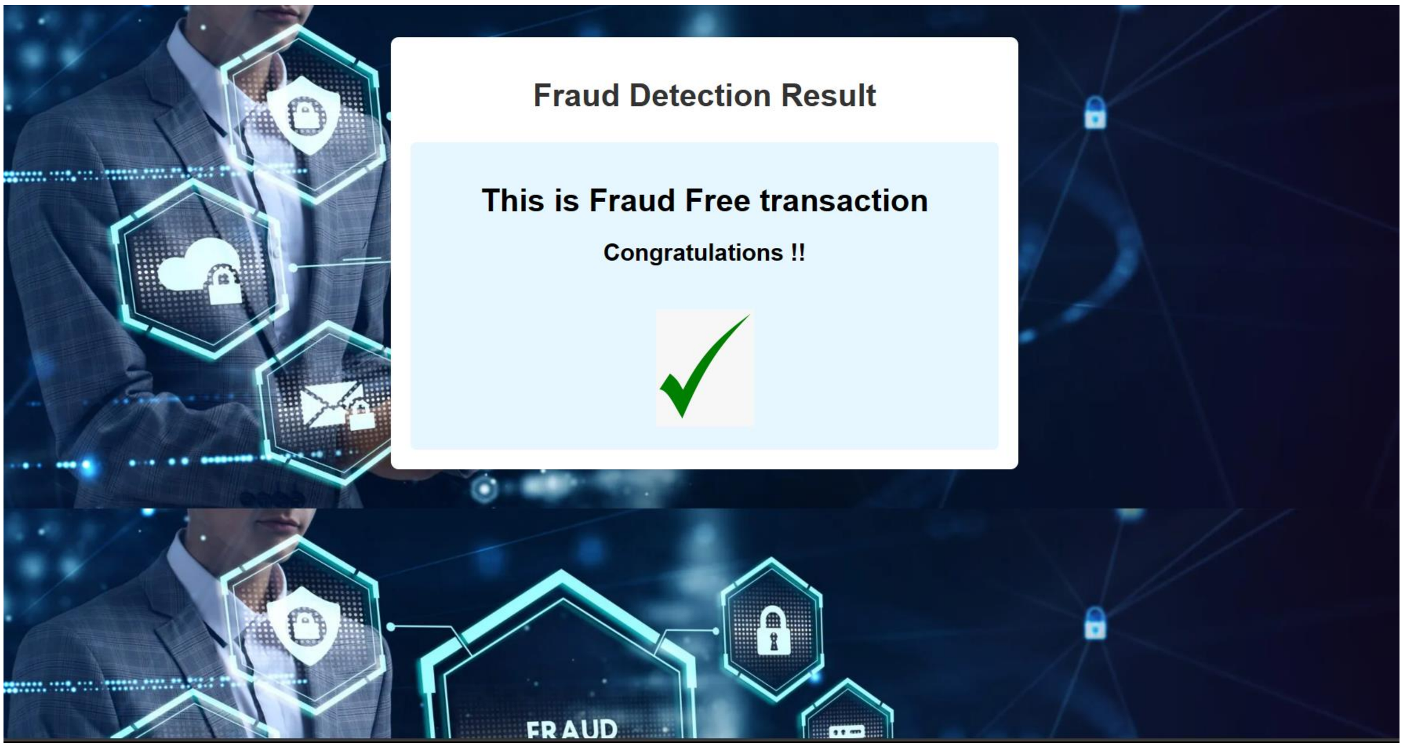
Amount of Transaction: 25000

Balance before the transaction: 250000

Balance after the transaction: 225000

The OLD balance of recipient after the transaction: 87000

The new balance of recipient after the transaction: 112000



Fraud Detection Result

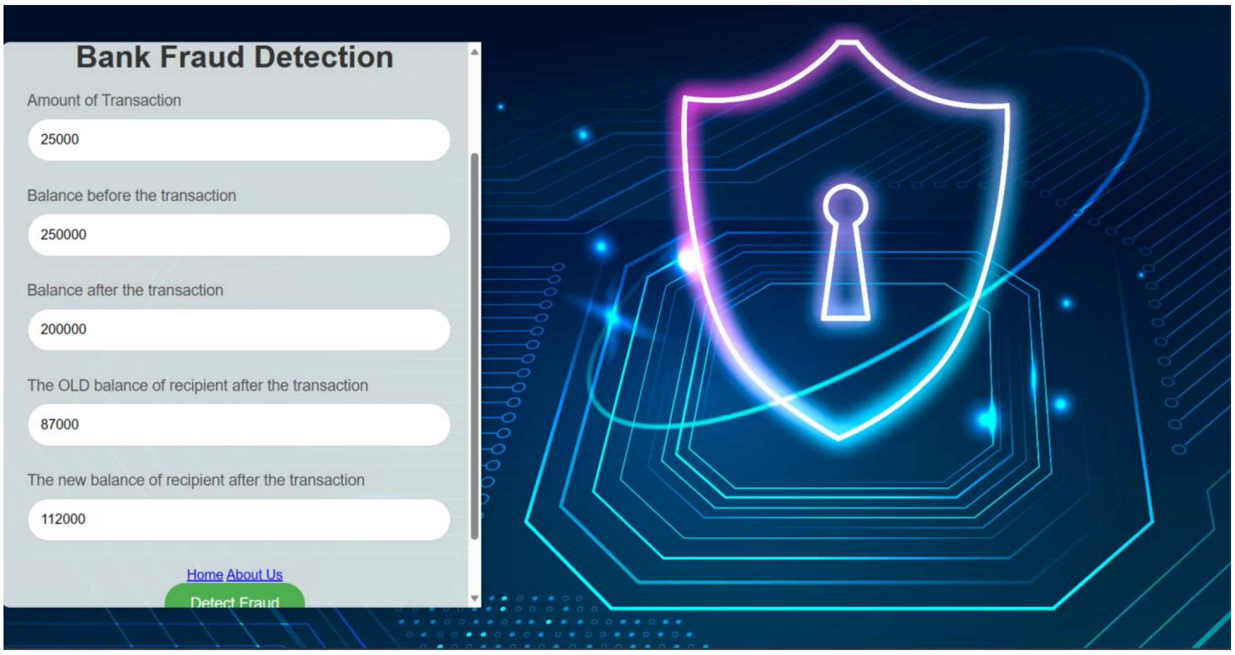
This is Fraud Free transaction

Congratulations !!

ATTEMPT 2

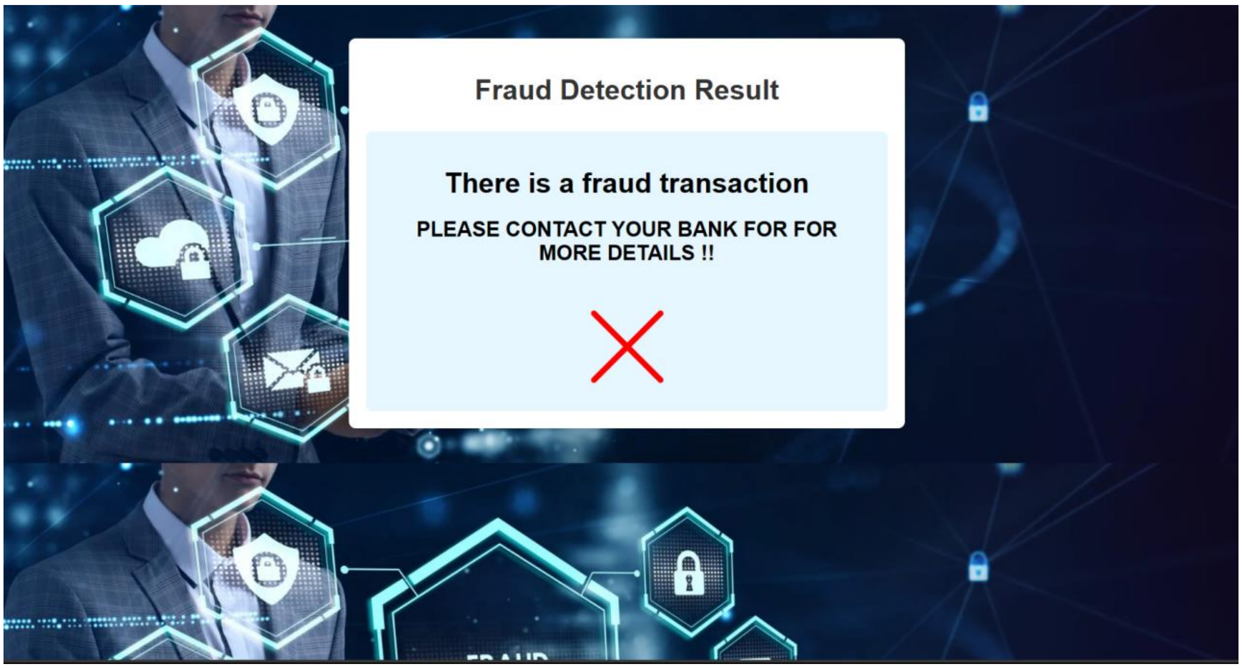
Here we will provide correct details of transaction and non accurate deduction of amount for checking

(NEGATIVE CHECKING)



Bank Fraud Detection

Amount of Transaction
25000
Balance before the transaction
250000
Balance after the transaction
200000
The OLD balance of recipient after the transaction
87000
The new balance of recipient after the transaction
112000



Fraud Detection Result

There is a fraud transaction

PLEASE CONTACT YOUR BANK FOR FOR MORE DETAILS !!