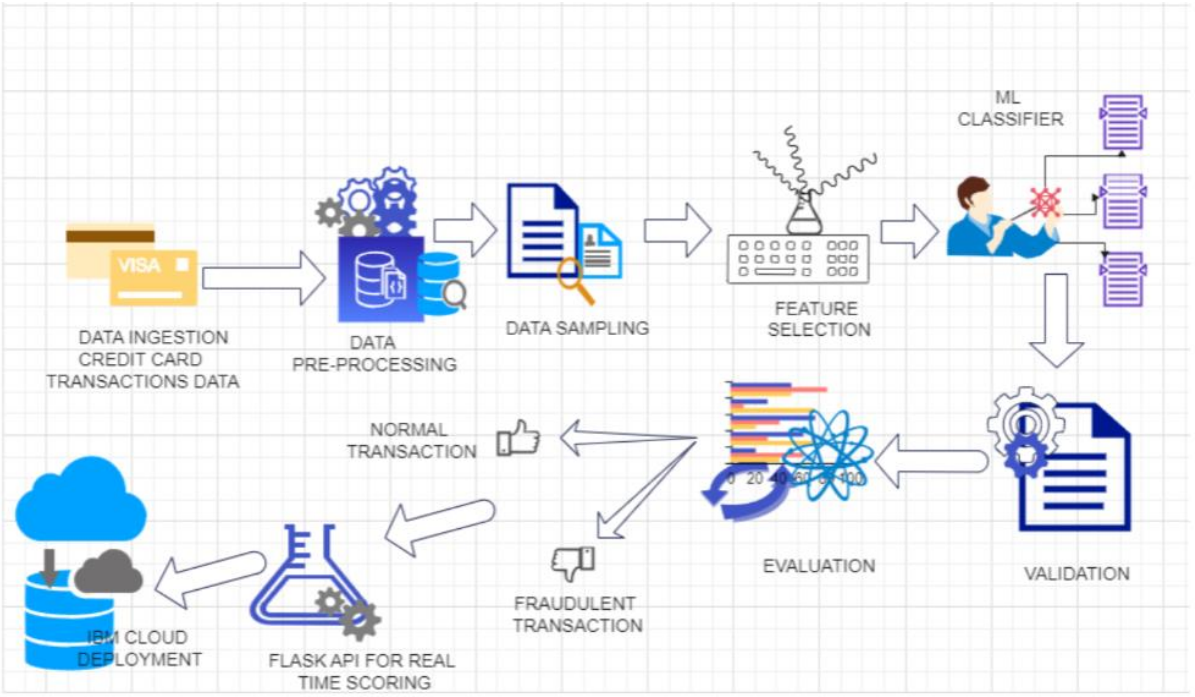
Online Payments Fraud Detection using ML

Project Description:

Online credit and debit card transactions seem to be a contributing factor to the expansion of the internet and e-commerce. Fraud is rising as a result of more people using credit and debit cards. There are several ways to identify the frauds, however they lag in their precision and have unique disadvantages of their own. If any modifications are made to the frauds are detected early in the transaction and dealt with accordingly. Owed the issue of credit/debit card fraud detection is resolved due to the volume of data suggested approach.

Classification techniques such Decision Tree, Random Forest, SVM, Extra Tree Classifier, and xgboost Classifier will be employed.We will use these methods to train and test the data. The optimal model is chosen from this and saved in PKL format. We'll be deploying IBM and integrating Flask.

Technical Architecture:-



Pre requisites:

To complete this project , you must require following software, concepts and packages

- Anaconda Navigator and VS code.
- Python packages
    - Open VS code→view→terminal→and ensure following steps
    - Type “pip install numpy”and click enter.
    - Type “pip install pandas”and click enter.
    - Type “pip install scikit-learn”and click enter.

- Type ”pip install matplotlib” and click enter.
- Type ”pip install scipy” and click enter.
- Type ”pip install pickle-mixin ” and click enter.
- Type ”pip install seaborn” and click enter.
- Type “pip install Flask” and click enter.

Prior Knowledge:

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

- Ml concepts(algos)
  - Supervised Learning
  - Unsupervised Learning
  - Regression and Classification
  - Decision Tree
  - Random Forest
  - XGboost Classifier
  - SVM
  - Extra tree classifier
  - Evaluation metrics
  - Flask Basics

Project Objectives:-

- At the conclusion of this assignment, you will have gained a wide understanding of data and be familiar with the basic principles and techniques used in machine learning.
- Be familiar with outlier transformation techniques, data preprocessing, and basic visualization principles.

Project Flow:-

● The user input is entered via interacting with the UI.

● The integrated model analyzes the entered data.

● The prediction appears on the user interface once the model has analyzed the input.

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

- Bivariate analysis
- Descriptive analysis

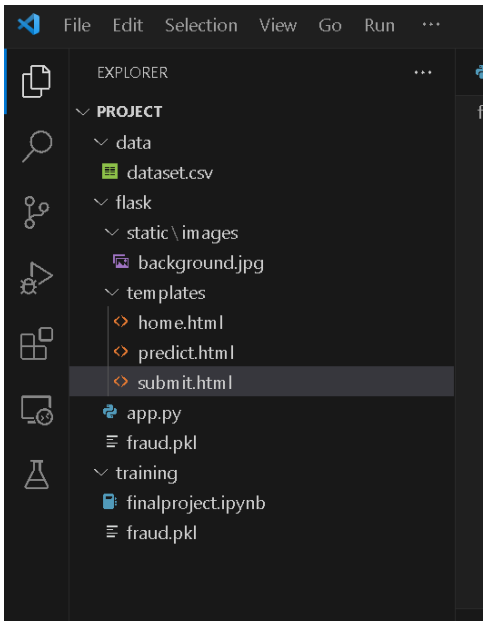
- Data pre-processing
  - Checking for null values
  - Handling outlier
  - Handling categorical(object) data
  - Splitting data into train and test

- Model building
  - Import the model building libraries
  - Initialising the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model

- Application Building
  - Create an HTML file
  - Build python code

Project structure:-

Create the Project folder which contains files as shown below



- We will build a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- fraud.pkl is our saved model. Further we will use this model for flask integration.

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 download it from some trusted source, here we will fetch it from Kaggle

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

Milestone 2: Visualising and analysing data

As the dataset is downloaded. Let us read and understand the data properly with the

help of some visualisation techniques and some analysing techniques.

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
sklearn.__version__
```

```
'1.2.2'
```

```
#Random forest classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import classification_report
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import ExtraTreesClassifier
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
from sklearn.preprocessing import LabelEncoder
```

```
import pickle
```

Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
[2] df=pd.read_csv('/content/dataset.csv')
```

```
[3] df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0	0.0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0	0.0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0	0.0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0	0.0

```
[4] df.columns
```

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

In the next step the dataset's superfluous columns are being removed using the drop method.

```
[ ] #here the last column isFlaggedFraud is not needed so we drop it
    df=df.drop(columns=['isFlaggedFraud'])
    df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0

About Dataset:-

The below column reference:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

Now we will check the correlation which will give the insight that how much one feature is related to other:-

```
[ ] #first check the relativity of features
    df.corr()
```

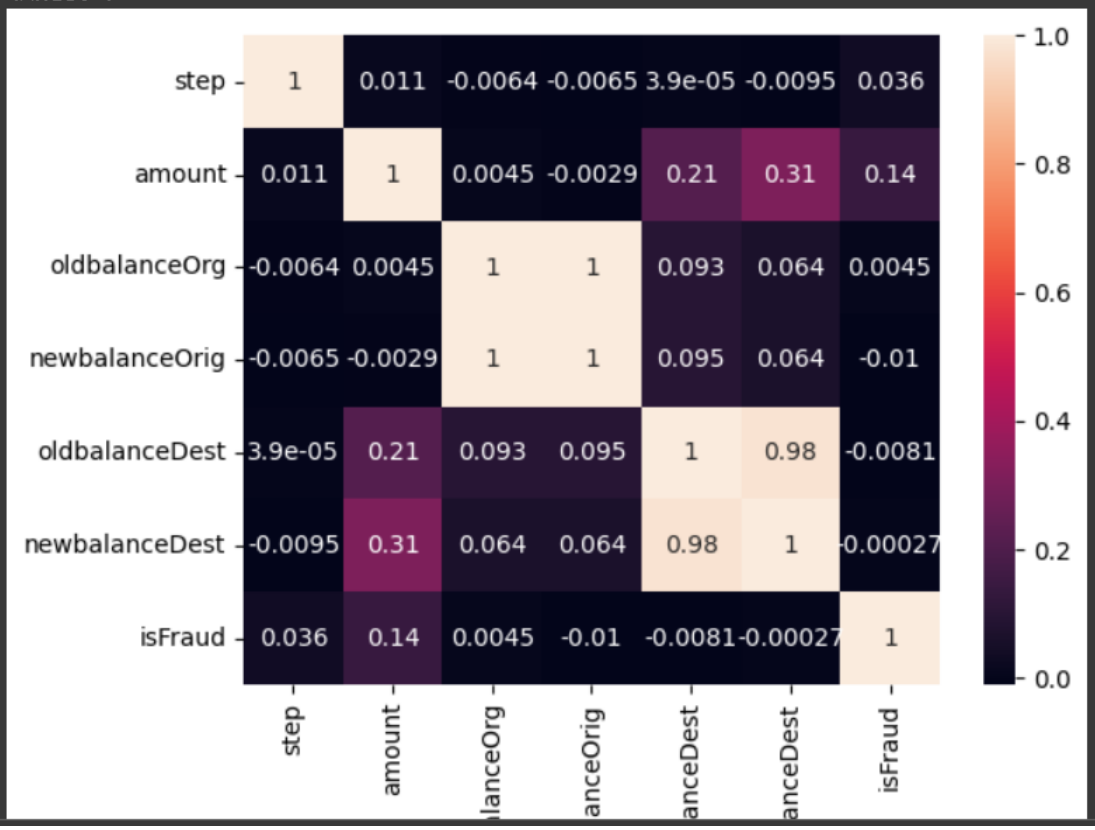
```
<ipython-input-281-5873ad39b12e>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version,
  df.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	0.010905	-0.006399	-0.006545	0.000039	-0.009474	0.035662
amount	0.010905	1.000000	0.004479	-0.002885	0.213497	0.312364	0.139414
oldbalanceOrg	-0.006399	0.004479	1.000000	0.998966	0.093427	0.063983	0.004528
newbalanceOrig	-0.006545	-0.002885	0.998966	1.000000	0.095357	0.063603	-0.010420
oldbalanceDest	0.000039	0.213497	0.093427	0.095357	1.000000	0.978773	-0.008083
newbalanceDest	-0.009474	0.312364	0.063983	0.063603	0.978773	1.000000	-0.000272
isFraud	0.035662	0.139414	0.004528	-0.010420	-0.008083	-0.000272	1.000000

Heat map for graphical visualisation.

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

```
<ipython-input-282-8df7bcac526d>:1: FutureWarning: The default value of numeric_
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```

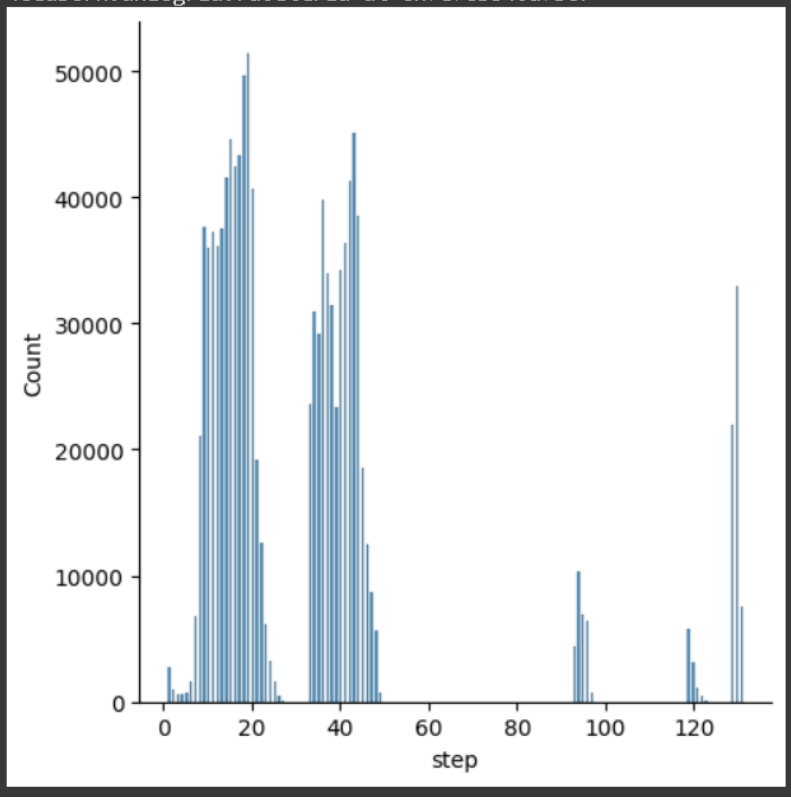


Univariate Analysis:-

Univariate analysis is understanding the data with a single feature.

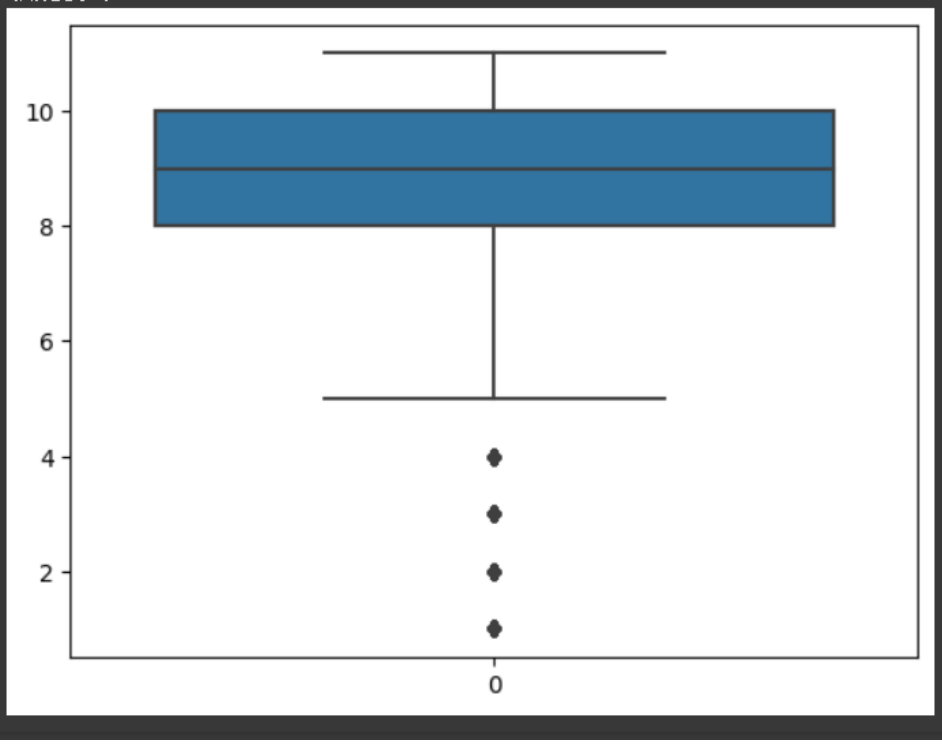
```
sns.displot(df.step)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e703846d7b0>
```



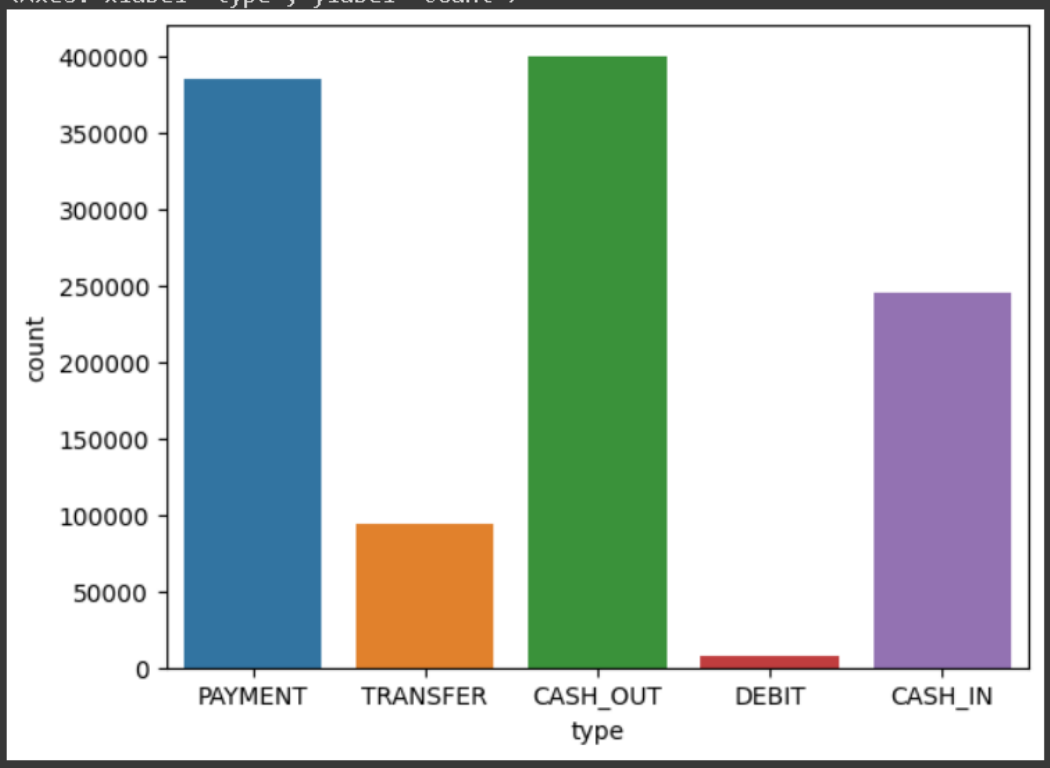
```
sns.boxplot(df.step)
```

```
<Axes: >
```




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

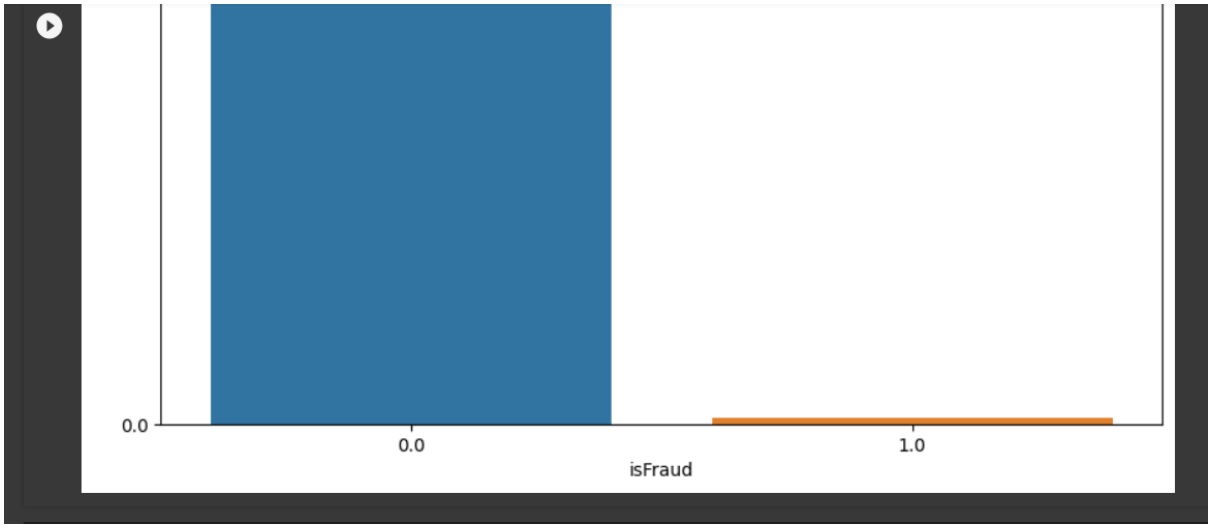
```
<Axes: xlabel='type', ylabel='count'>
```



And similar analysis which could be found in the python notebook

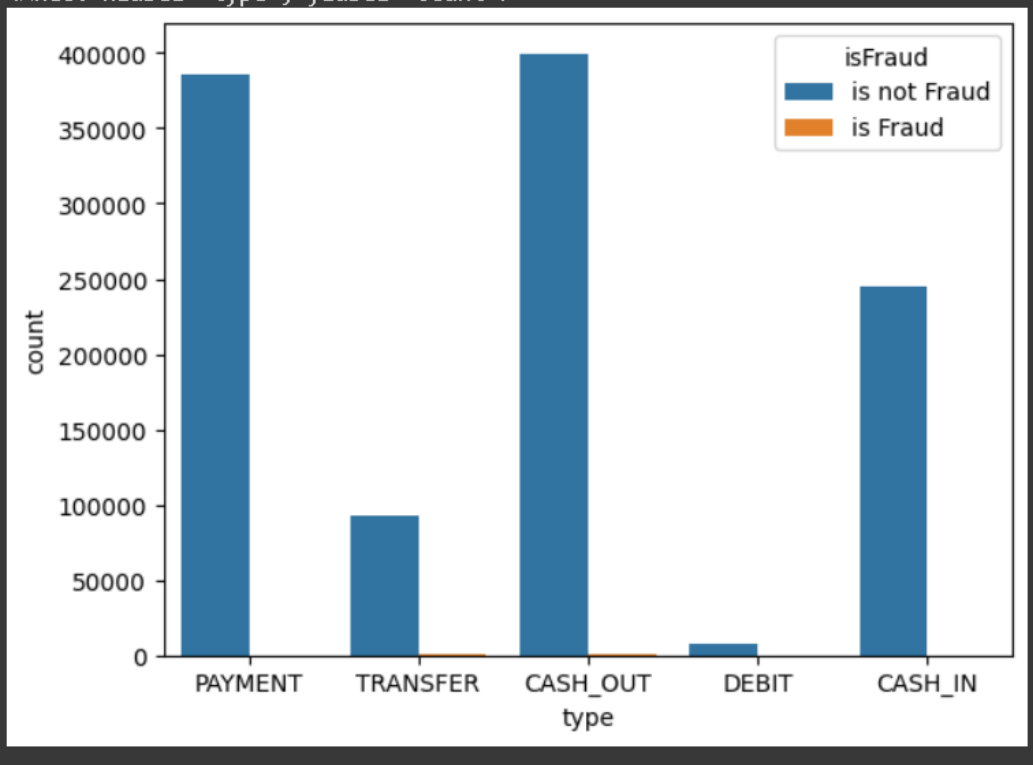
Bivariate analysis.

```
plt.figure(figsize=(10, 50))
sns.countplot(data=df,x='isFraud')
```




```
#fraud against type of payment
sns.countplot(data=df,x='type',hue='isFraud')
```

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



And such similar analysis.

Descriptive analysis:-

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
[11] df.describe()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
count	124457.000000	1.244570e+05	1.244560e+05	1.244560e+05	1.244560e+05	1.244560e+05	124456.000000
mean	8.920945	1.773613e+05	9.047501e+05	9.209941e+05	9.004621e+05	1.185246e+06	0.000964
std	1.857028	3.440304e+05	2.850751e+06	2.887776e+06	2.391423e+06	2.749984e+06	0.031037
min	1.000000	3.200000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
25%	8.000000	1.059726e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
50%	9.000000	5.785259e+04	2.009300e+04	0.000000e+00	2.830632e+04	7.527365e+04	0.000000
75%	10.000000	2.180569e+05	1.952794e+05	2.222311e+05	6.424462e+05	1.097218e+06	0.000000
max	11.000000	1.000000e+07	3.893942e+07	3.894623e+07	3.400874e+07	3.894623e+07	1.000000

Data Pre-processing.

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

Handling missing values

Handling Object data label encoding

Splitting dataset into training and test set

Checking data head and null values after train test split and replacing it with median

```
#dividing dataset in dependent and independent y and x respectively
x=df.drop('isFraud',axis=1)
y=df['isFraud']
```

```
[37] x.head()
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	1.0	3	9839.64	170136.0	160296.36	0.0	0.0
1	1.0	3	1864.28	21249.0	19384.72	0.0	0.0
2	1.0	4	181.00	181.0	0.00	0.0	0.0
3	1.0	1	181.00	181.0	0.00	21182.0	0.0
4	1.0	3	11668.14	41554.0	29885.86	0.0	0.0

```
[38] #splitting data into train and test
     from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

```
x_train.isnull().sum()
```

```
step              0
type              0
amount            0
oldbalanceOrg     1
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
dtype: int64
```

```
[56] x_train.oldbalanceOrg.fillna(x_train.oldbalanceOrg.median(),inplace=True)
```

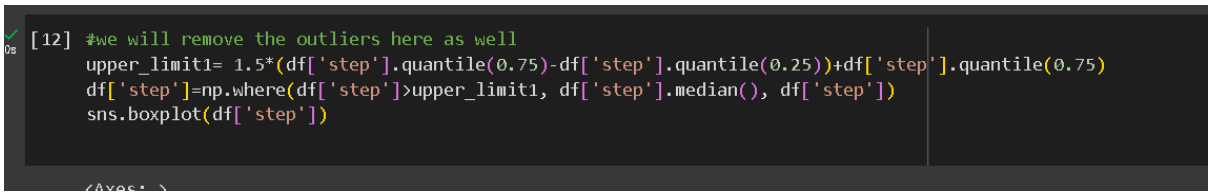
```
[57] print(x_train.shape)
     print(x_test.shape)
     print(y_test.shape)
     print(y_train.shape)
```

```
(99565, 7)
(24892, 7)
(24892,)
(99565,)
```

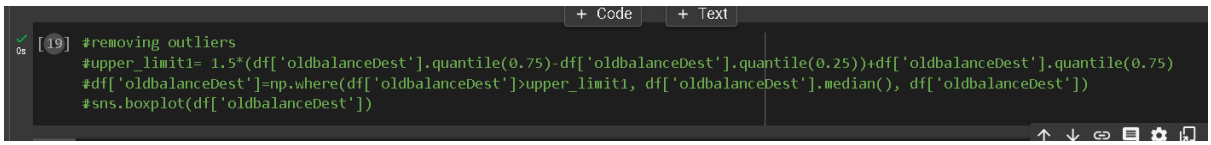
✓ 7s completed a

29°C

Handling outliers:-

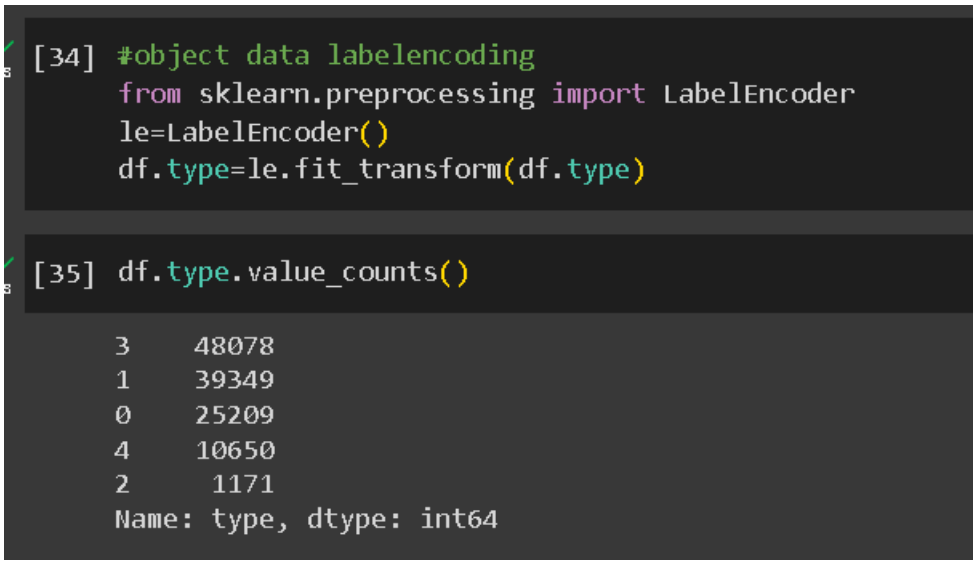


```
[12] #we will remove the outliers here as well
     upper_limit1= 1.5*(df['step'].quantile(0.75)-df['step'].quantile(0.25))+df['step'].quantile(0.75)
     df['step']=np.where(df['step']>upper_limit1, df['step'].median(), df['step'])
     sns.boxplot(df['step'])
```



```
+ Code  + Text
[19] #removing outliers
     #upper_limit1= 1.5*(df['oldbalanceDest'].quantile(0.75)-df['oldbalanceDest'].quantile(0.25))+df['oldbalanceDest'].quantile(0.75)
     #df['oldbalanceDest']=np.where(df['oldbalanceDest']>upper_limit1, df['oldbalanceDest'].median(), df['oldbalanceDest'])
     #sns.boxplot(df['oldbalanceDest'])
```

Object data label encoding



```
[34] #object data labelencoding
     from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
     df.type=le.fit_transform(df.type)

[35] df.type.value_counts()

     3    48078
     1    39349
     0    25209
     4    10650
     2     1171
     Name: type, dtype: int64
```

Model building(Random Forest, Decision tree classifier, extra tree classifier, SVM, xgboost Classifier)

1: Random Forest classifier¶

A function named RandomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

2: Decision tree Classifier

A function named Decisiontree is created and train and test data are passed as the parameters. Inside the function, the DecisiontreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with

the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

3: ExtraTrees Classifier¶

A function named ExtraTree is created and train and test data are passed as the parameters. Inside the function, ExtraTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

4: SupportVectorMachine Classifier¶

A function named SupportVector is created and train and test data are passed as the parameters. Inside the function, the SupportVectorClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, confusion matrix and classification report is done

5: xgboost Classifier¶

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the xgboostClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, confusion matrix and classification report is done

Screenshots :-

```
[58] #Random forest classifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     rfc=RandomForestClassifier(random_state=42)
     rfc.fit(x_train,y_train)
     y_test_predict1=rfc.predict(x_test)
     test_accuracy=accuracy_score(y_test,y_test_predict1)
     test_accuracy
```

```
0.9993572232042424
```

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

```
1.0
```

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	5	14
is not Fraud	2	24871

```
[62] from sklearn.metrics import classification_report
     print(classification_report(y_test,y_test_predict1))
```

```
                precision    recall  f1-score   support

      is Fraud       0.71      0.26      0.38        19
  is not Fraud       1.00      1.00      1.00     24873

      accuracy                           1.00     24892
     macro avg       0.86      0.63      0.69     24892
  weighted avg       1.00      1.00      1.00     24892
```

Decision tree classifier

```
[63] 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
```

```
0.9990358348063635
```

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

```
1.0
```

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	9	10
is not Fraud	14	24859

```
[66] print(classification_report(y_test,y_test_predict2))
```

```
                precision    recall  f1-score   support

      is Fraud       0.39      0.47      0.43        19
  is not Fraud       1.00      1.00      1.00     24873

      accuracy                           1.00     24892
     macro avg       0.70      0.74      0.71     24892
  weighted avg       1.00      1.00      1.00     24892
```

Decision tree classifier

```
[63] 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
```

```
0.9990358348063635
```

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

```
1.0
```

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	9	10
is not Fraud	14	24859

```
[66] print(classification_report(y_test,y_test_predict2))
```

```
                precision    recall  f1-score   support

      is Fraud       0.39      0.47      0.43        19
  is not Fraud       1.00      1.00      1.00     24873

      accuracy                           1.00     24892
     macro avg       0.70      0.74      0.71     24892
  weighted avg       1.00      1.00      1.00     24892
```

Extra tree Classifier

```
[67] 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
```

```
0.999156355455568
```

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

```
[70] train_accuracy
```

```
1.0
```

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	2	17
is not Fraud	4	24869

```
[72] print(classification_report(y_test,y_test_predict3))
```

```
                precision    recall  f1-score   support

      is Fraud       0.33      0.11      0.16        19
  is not Fraud       1.00      1.00      1.00     24873

      accuracy                           1.00     24892
     macro avg       0.67      0.55      0.58     24892
  weighted avg       1.00      1.00      1.00     24892
```

Support vector machine classifier

```
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
```

```
0.9992367025550377
```

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

```
0.9990056746848792
```

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

col_0	is not Fraud
isFraud	
is Fraud	19
is not Fraud	24873

```
[76] print(classification_report(y_test,y_test_predict4))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Un
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

    is Fraud       0.00      0.00      0.00        19
is not Fraud       1.00      1.00      1.00     24873

    accuracy                           1.00     24892
   macro avg       0.50      0.50      0.50     24892
weighted avg       1.00      1.00      1.00     24892

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Un
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Un
  _warn_prf(average, modifier, msg_start, len(result))
```

xgboost Classifier

```
[77] from sklearn.preprocessing import LabelEncoder
     la=LabelEncoder()
     y_train1=la.fit_transform(y_train)
     y_test1=la.transform(y_test)
     y_test1=la.transform(y_test)

     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
```

```
0.9995179174031817
```

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

```
0.9999799126198966
```

```
[79] pd.crosstab(y_test,y_test_predict5)
```

col_0	0	1
isFraud		
is Fraud	10	9
is not Fraud	3	24870

```
[80] print(classification_report(y_test1,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       0.77      0.53      0.62        19
           1       1.00      1.00      1.00     24873

    accuracy                           1.00     24892
   macro avg       0.88      0.76      0.81     24892
weighted avg       1.00      1.00      1.00     24892
```

Comparing the model

```
[81] 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 svc",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))
```

```
train accuracy for rfc 1.0
test accuracy for rfc 0.9993572232042424
train accuracy for dtc 1.0
test accuracy for dtc 0.9990358348063635
train accuracy for etc 1.0
test accuracy for etc 0.999156355455568
train accuracy for svc 0.9990056746848792
test accuracy for svc 0.9992367025550377
train accuracy for xgb1 0.9999799126198966
test accuracy for xgb1 0.9995179174031817
```

Import Pickle dump

```
import pickle
pickle.dump(rfc,open('fraud.pkl','wb'))
```

Application building

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

This section has the following tasks

Building HTML Pages

Building server side script

Activity1: Building Html Pages:

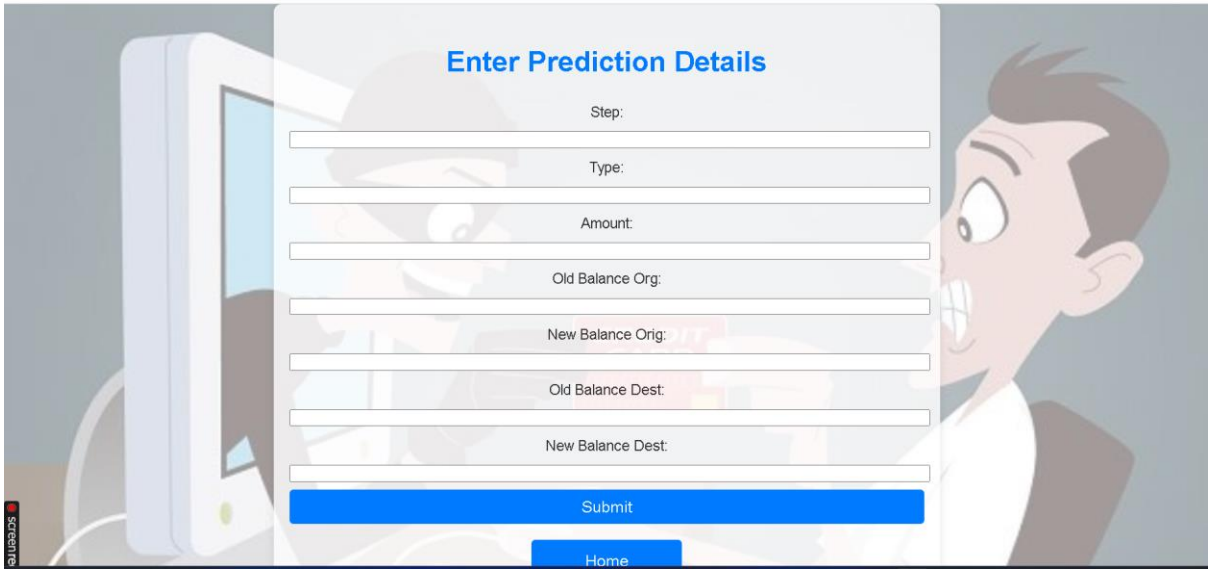
For this project create three HTML files namely

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

and save them in the templates folder.

Home.html

Predict.html



Submit.html

Code for app deployment

App.py

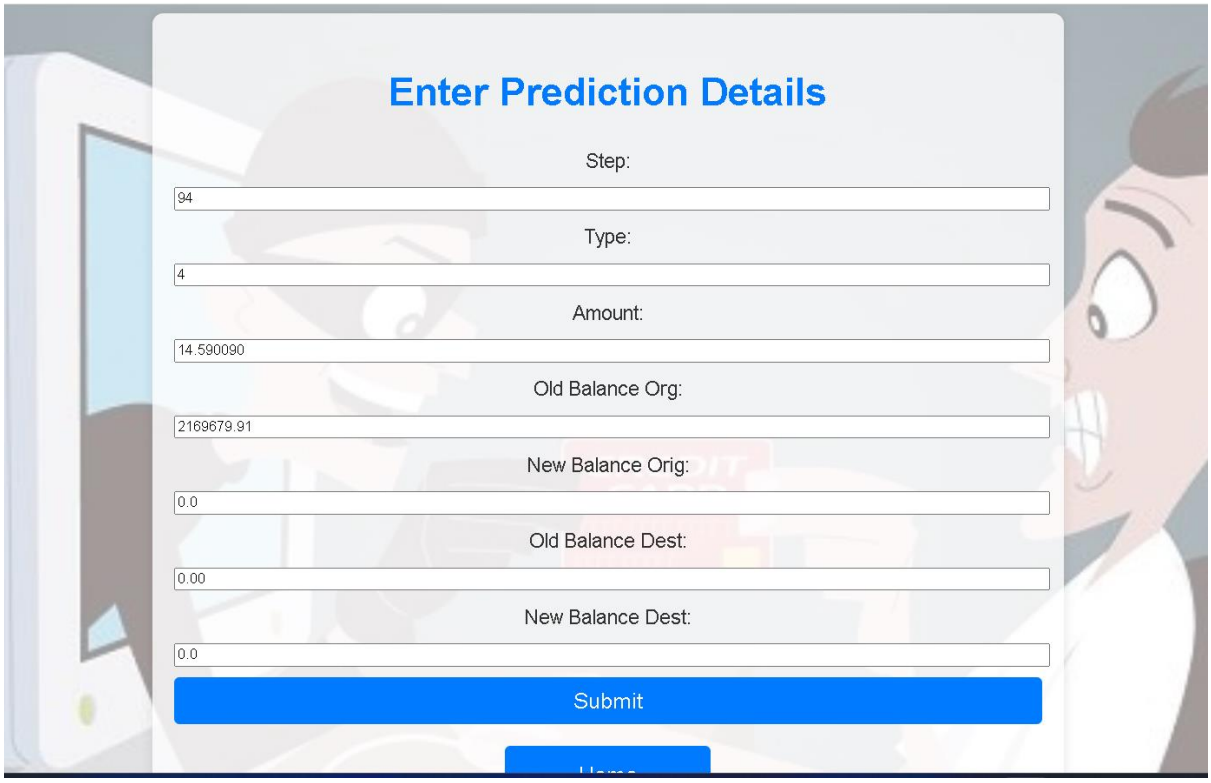
```
import numpy as np
import pickle
import pandas as pd
from flask import Flask, render_template, request

model = pickle.load(open(r"fraud.pkl", 'rb'))
app = Flask(__name__)
```

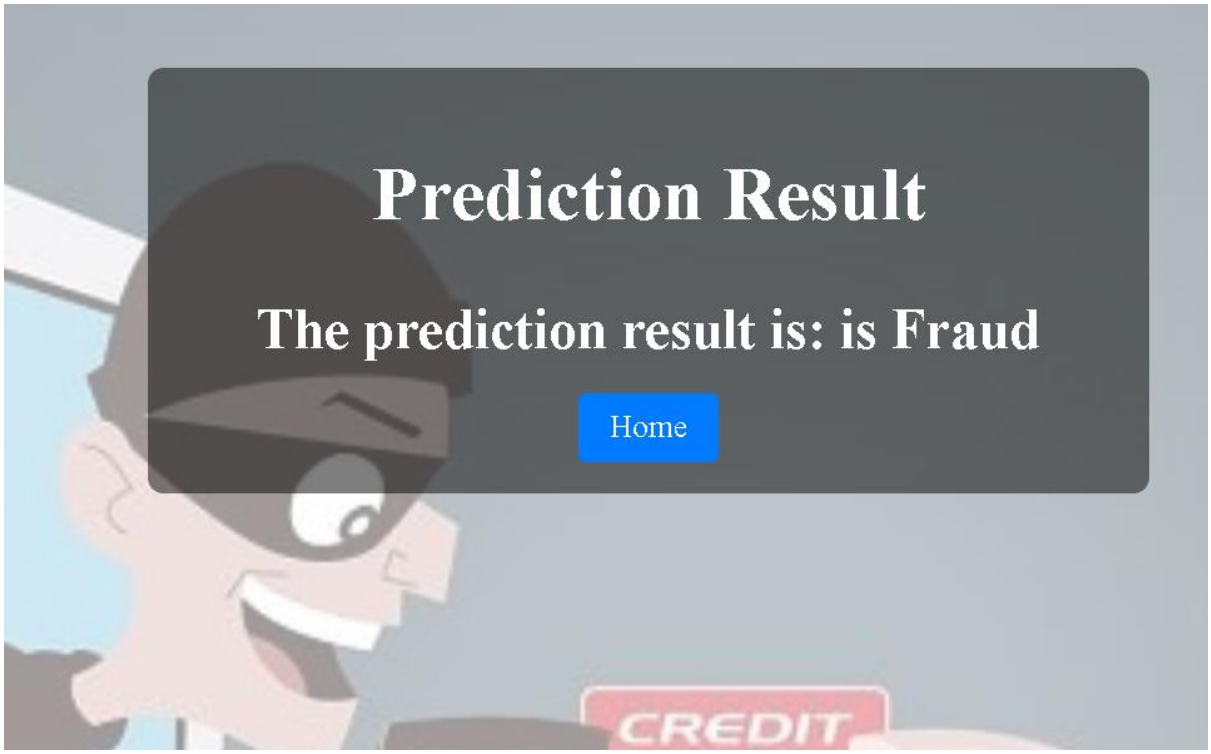
```python
@app.route("/")
def about():
    return render_template('home.html')

@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_str=str(pred))

@app.route("/submit", methods=['POST'])
def submit():
    # Retrieve form data
    step = request.form.get('step')
    Type = request.form.get('Type')
    amount = request.form.get('amount')
    oldbalanceorg = request.form.get('oldbalanceorg')
    newbalanceorig = request.form.get('newbalanceorig')
    oldbalancedest = request.form.get('oldbalancedest')
    newbalancedest = request.form.get('newbalancedest')

    # Process the form data (add your processing code here)
    # For example, you can pass the form data to your model for prediction
    form_data = [[step, Type, amount, oldbalanceorg, newbalanceorig,
oldbalancedest, newbalancedest]]
    form_data = np.array(form_data).astype(float)
    prediction = model.predict(form_data)
    prediction_str = str(prediction[0])

    return render_template('submit.html', prediction_str=prediction_str)

if __name__ == "__main__":
    app.run(debug=False)
```

Run the application

- **Open VScode command prompt.**
- **Navigate to the folder where your python script is.**
- **Now type “python app.py” command**
- **Navigate to the localhost where you can view your web page.**
- **Click on the predict button from the top right corner, enter the inputs,**

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





Enter Prediction Details

Step:

1

Type:

3

Amount:

9.194174

Old Balance Org:

170136.00

New Balance Orig:

160296.36

Old Balance Dest:

0.0

New Balance Dest:

0.0

Submit


Prediction Result
The prediction result is: is not Fraud
Home