ONLINE PAYMENTS FRAUD DETECTION USING ML

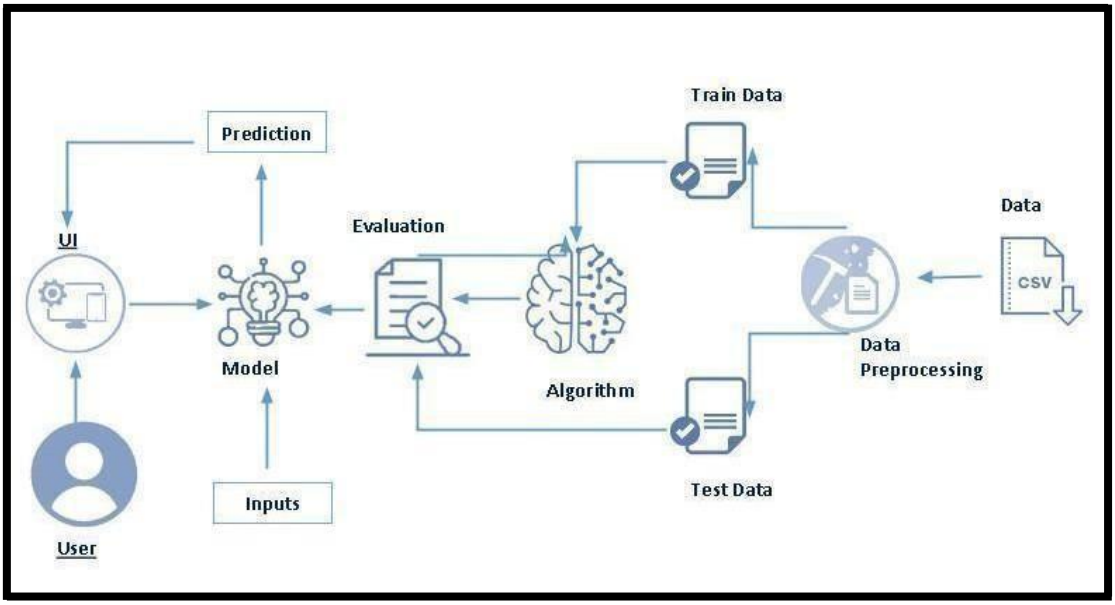
PROJECT MANUAL

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 classifier and Extra tree 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.

Technical Architecture:



Prerequisites:

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

- **Anaconda navigator and PyCharm:** ○ Refer the link below to download anaconda navigator ○ Link: https://youtu.be/1ra4zH2G4o0
- **Python packages:**
  - Open anaconda prompt as administrator ○ 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**
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning ○ Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning ○ SVM: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
  - Decision Tree: https://www.analyticsvidhya.com/blog/2022/03/decision-tree-machine-learning-using-python/
  - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-errormetrics/
- **Flask Basics:** https://www.youtube.com/watch?v=lj4I_CvBnt0

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 visualization 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

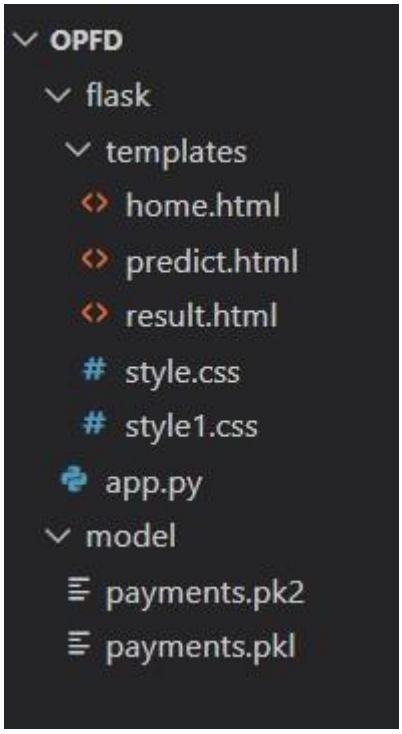
To accomplish this, we have to complete all the activities listed below,

- Data collection – Collect the dataset or create the dataset.
- Visualizing and analysing data – Univariate analysis, Bivariate analysis, Multivariate analysis and Descriptive analysis.

- Data pre-processing – Checking for null values, handling outliers, handling categorical data, and splitting data into train and test.
- Model building – Import the model building libraries, Initializing the model, Training, and testing the model, evaluating performance of model and save the model.
- Application Building – Create an HTML file and Build python code.

Project Structure:

Create the Project folder which contains files as shown below.



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Car prediction.pkl is our saved model. Further we will use this model for flask integration.
- Templates folder contains html files and assets contains css,images,js,scss and vendor files.

Milestone 1: Define Problem / Problem Understanding

1. Define Problem Scope:

Clearly outline the scope of the online payments fraud detection problem. Identify the types of fraud to be addressed (e.g., credit card fraud, identity theft) and specify the range of platforms or systems the solution will cover.

2. Identify Key Stakeholders:

Identify and engage with key stakeholders, including financial institutions, online payment service providers, and end-users. Understand their perspectives, concerns, and requirements to ensure that the solution aligns with their needs.

3. Establish Success Metrics:

Define measurable success metrics that will be used to evaluate the performance of the fraud detection solution. Metrics such as precision, recall, false positive rate, and user satisfaction should be considered to assess the effectiveness and impact of the system.

Social Impact:

Milestone 2: Data Collection and Visualizing and analysing the data.

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

Activity 1: Download the dataset.

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

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

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

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

Note: There is n 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 2: Importing the libraries.

Import the necessary libraries as shown in the image. Here we have used visualization style as five thirty-eight.

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

Activity 3: 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 csv file.

```
# Reading the csv data
df = pd.read_csv(r'/content/PS_20174392719_1491204439457_log.csv')
```

```
df
```

	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

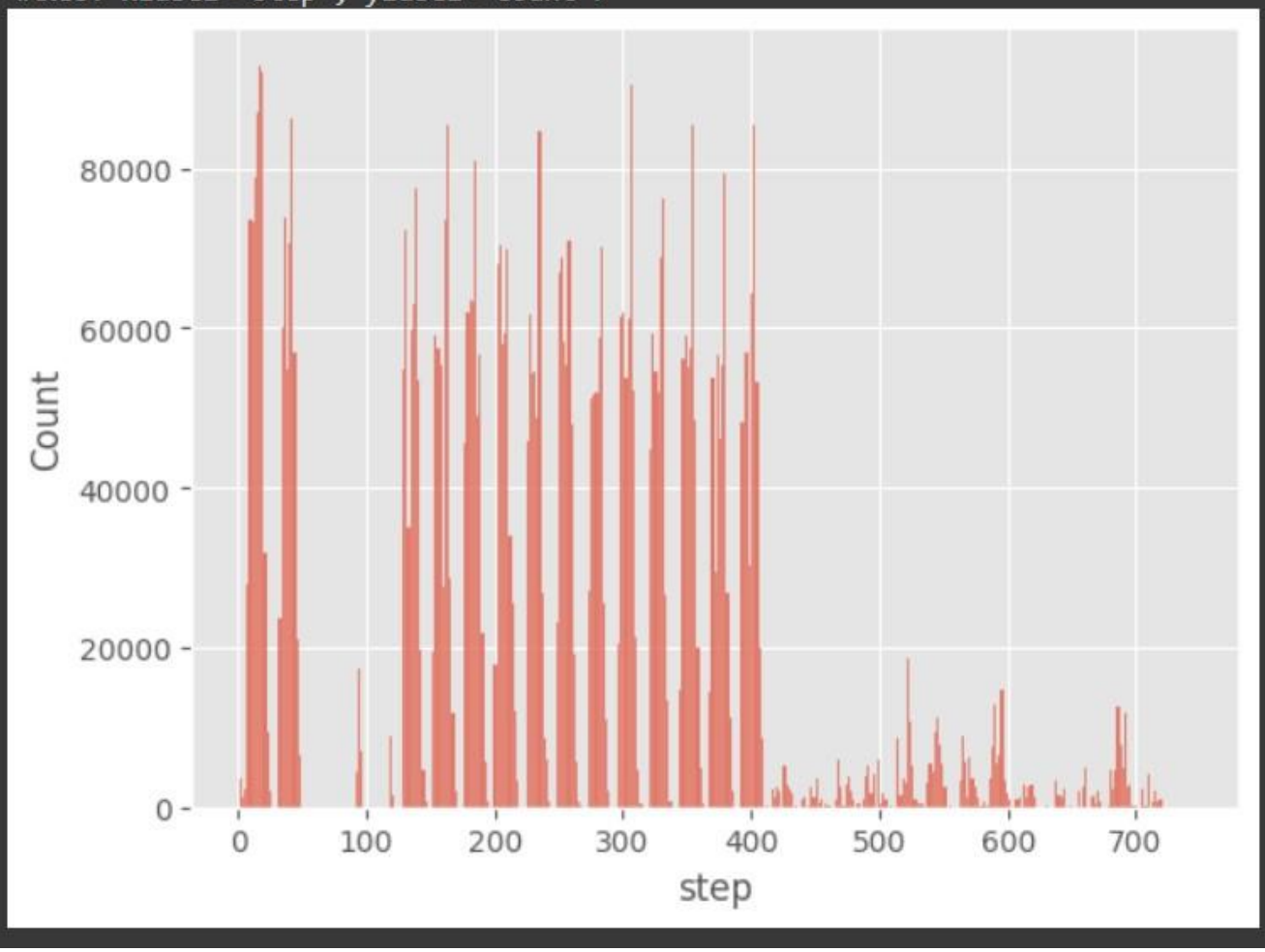
Activity 4: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Histplot and countplot.

- Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

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

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

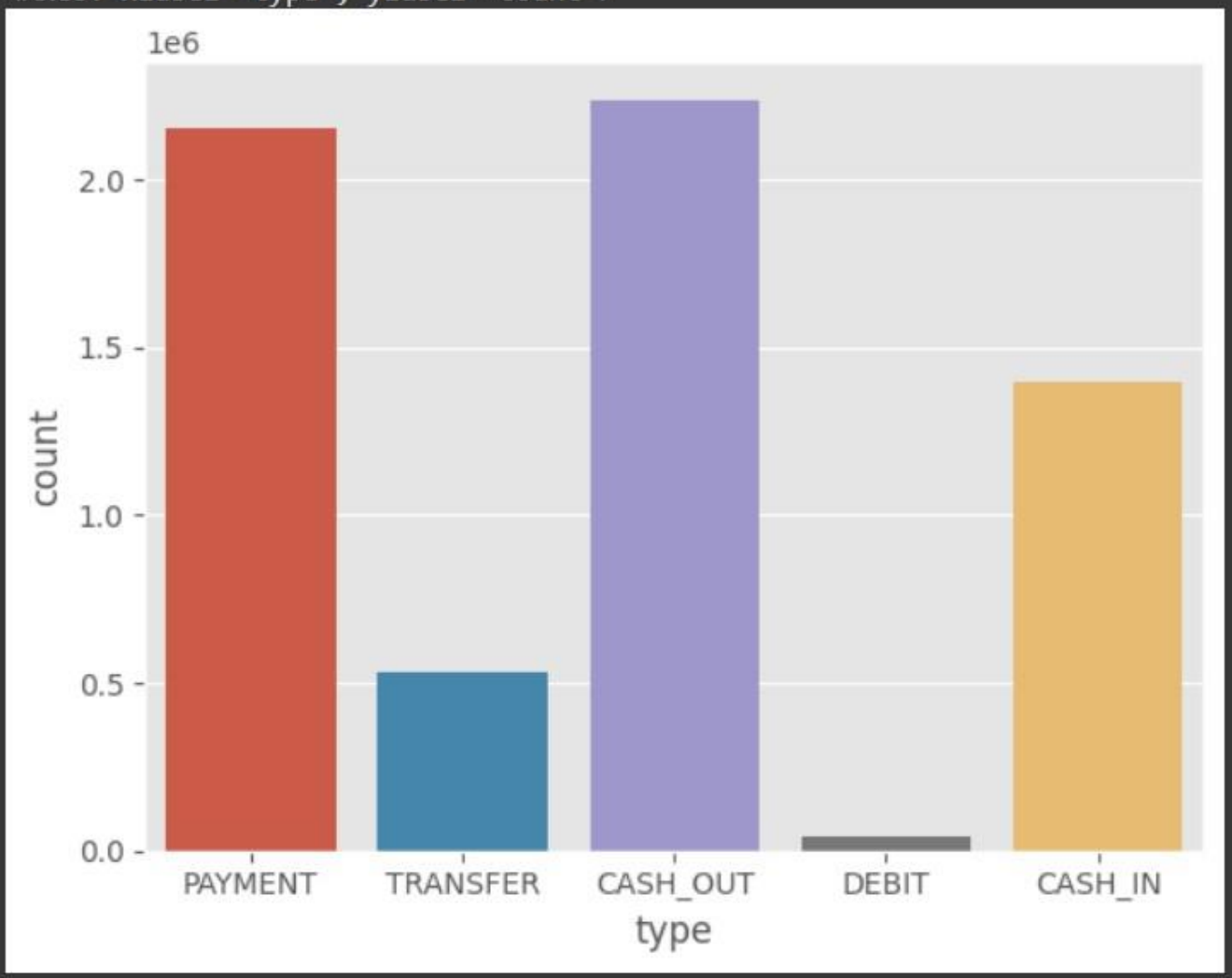


Activity 5: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between ‘Age’ and ‘Annualsalary’ variables using scatterplot.

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

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

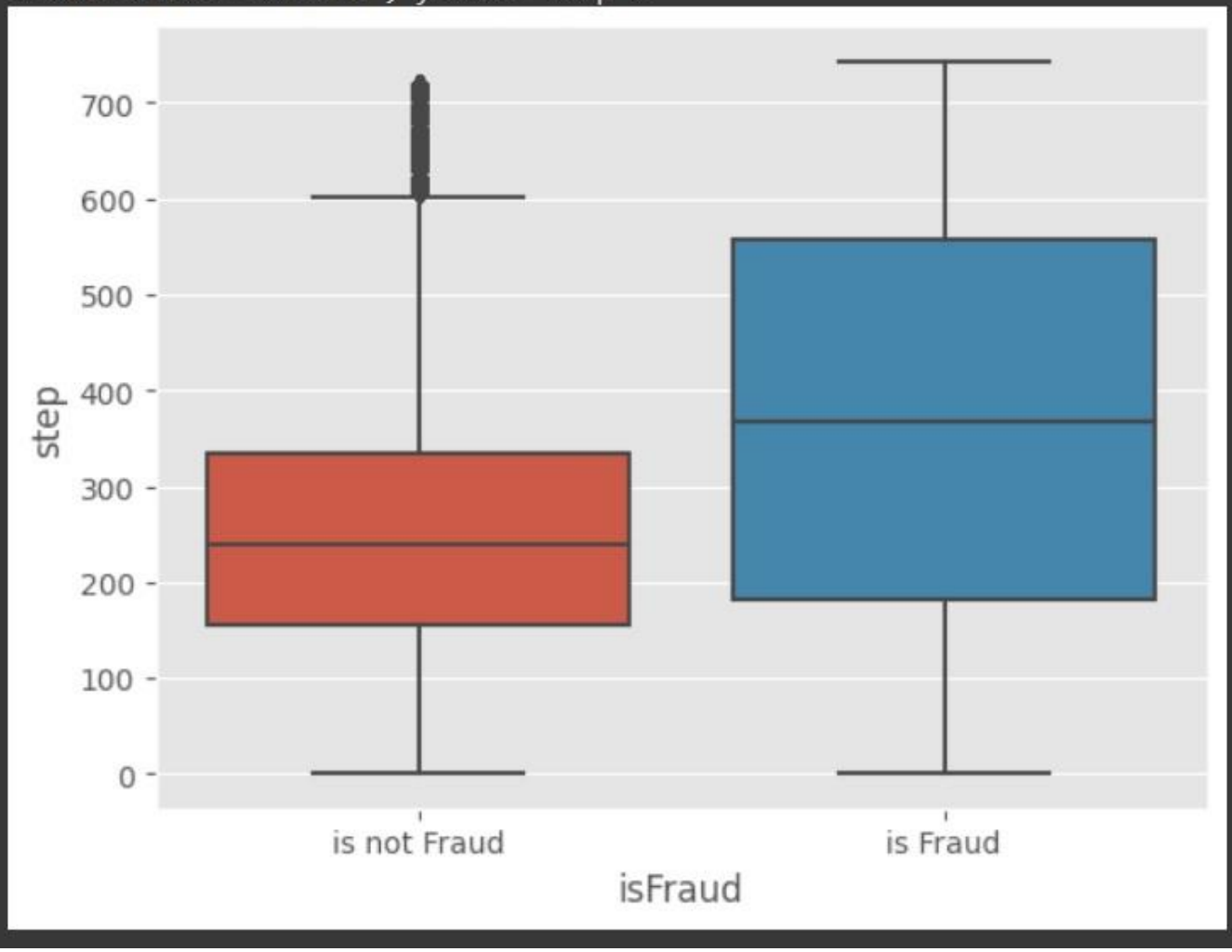


Activity 6: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used boxplot from seaborn package.

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

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



Activity 7: 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.

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

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
count	6.362620e+06	6362620	6.362620e+06	6362620	6.362620e+06	6.362620e+06	6362620	6.362620e+06	6.362620e+06	6362620
unique	NaN	5	NaN	6353307	NaN	NaN	2722362	NaN	NaN	2
top	NaN	CASH_OUT	NaN	C1902386530	NaN	NaN	C1286084959	NaN	NaN	is not Fraud
freq	NaN	2237500	NaN	3	NaN	NaN	113	NaN	NaN	6354407
mean	2.433972e+02	NaN	1.798619e+05	NaN	8.338831e+05	8.551137e+05	NaN	1.100702e+06	1.224996e+06	NaN
std	1.423320e+02	NaN	6.038582e+05	NaN	2.888243e+06	2.924049e+06	NaN	3.399180e+06	3.674129e+06	NaN
min	1.000000e+00	NaN	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
25%	1.560000e+02	NaN	1.338957e+04	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
50%	2.390000e+02	NaN	7.487194e+04	NaN	1.420800e+04	0.000000e+00	NaN	1.327057e+05	2.146614e+05	NaN
75%	3.350000e+02	NaN	2.087215e+05	NaN	1.073152e+05	1.442584e+05	NaN	9.430367e+05	1.111909e+06	NaN
max	7.430000e+02	NaN	9.244552e+07	NaN	5.958504e+07	4.958504e+07	NaN	3.560159e+08	3.561793e+08	NaN

Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of 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 categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 1: Checking for null values

- Let's find the shape of our dataset first, To find the shape of our data, df.shape method is used. To find the data type, df.info() function is used.

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 8 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   oldbalanceOrg   float64
 4   newbalanceOrig  float64
 5   oldbalanceDest  float64
 6   newbalanceDest  float64
 7   isFraud         object
dtypes: float64(5), int64(1), object(2)
memory usage: 388.3+ MB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.

Let's look for any outliers in the dataset.

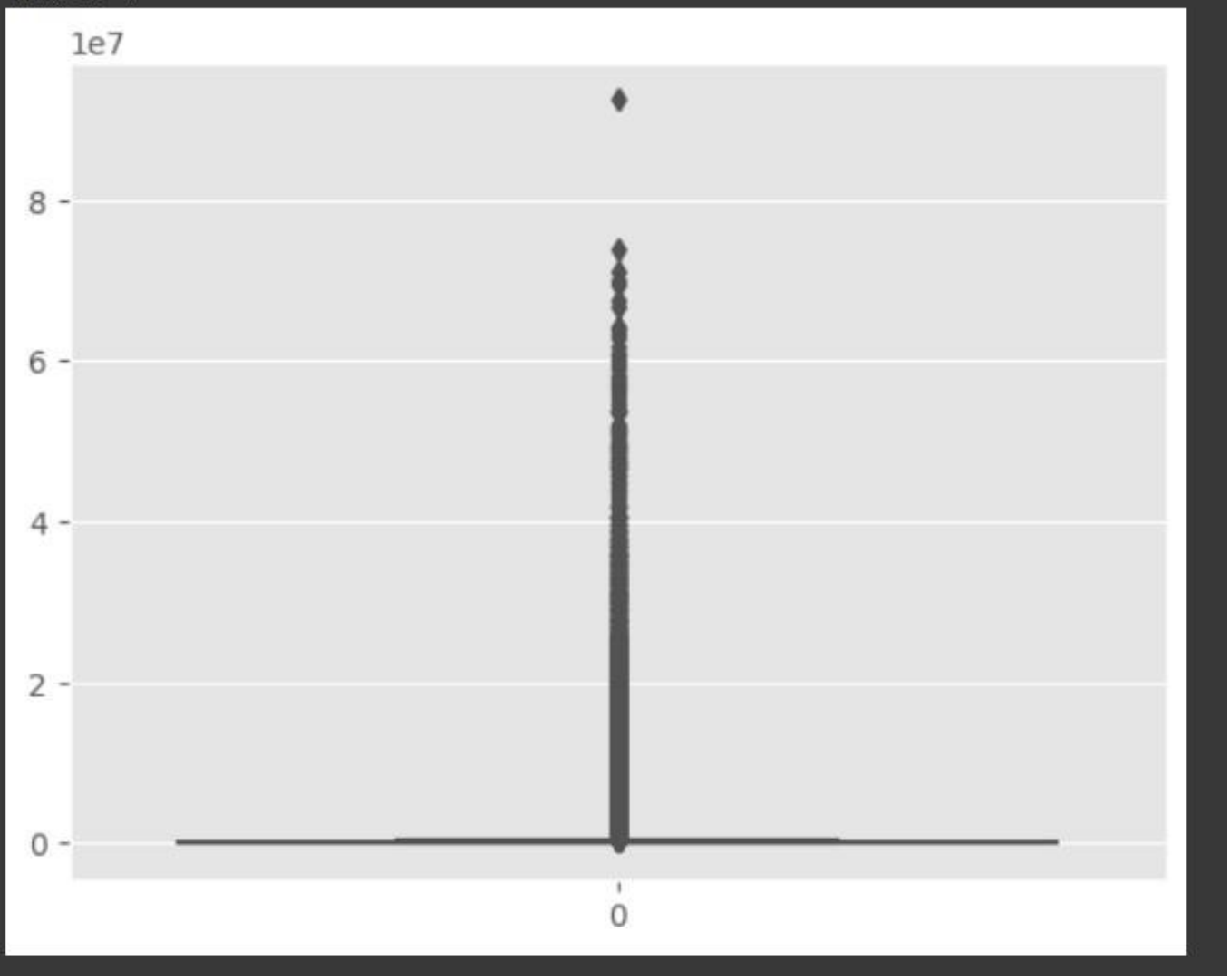
Activity 2: Handling outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of all features with some mathematical formula.

- From the below diagram, we could visualize that Monetary feature has outliers. Boxplot from seaborn library is used here.

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

```
<Axes: >
```



Activity 3: Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
[ ] from sklearn.model_selection import train_test_split

    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
    print(x_train.shape)
    print(x_test.shape)
    print(y_test.shape)
    print(y_train.shape)
```

```
(5090096, 7)
(1272524, 7)
(1272524,)
(5090096,)
```

Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1: Decision Tree Classifier

Decision Tree Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is do

```
[ ] 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
```

```
0.999704524236871
```

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

```
1.0
```

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1445	196
is not Fraud	180	1270703

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

```
              precision    recall  f1-score   support

    is Fraud       0.89      0.88      0.88      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.94      0.94      0.94   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Activity 2: Extra Trees Classifier model

```
[ ] 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
```

```
0.9996990233583021
```

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

```
1.0
```

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

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	1274	367
is not Fraud	16	1270867

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

```
              precision    recall  f1-score   support

    is Fraud       0.99      0.78      0.87      1641
is not Fraud       1.00      1.00      1.00   1270883

    accuracy                           1.00   1272524
   macro avg       0.99      0.89      0.93   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Activity 3: Evaluating performance of the model

```
[ ] from sklearn.preprocessing import LabelEncoder

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

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

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

```
[ ] y_train1
```

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

```
[ ] def compareModel():
      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))
```

```
    compareModel()
```

```
    train accuracy for dtc 1.0
    test accuracy for dtc 0.999704524236871
    train accuracy for etc 1.0
    test accuracy for etc 0.9996990233583021
```

Activity 4: Predict the chance of donation according to given factors.

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

```
    1.0
```

Our model is performing well. So, we are saving the model by pickle.dump().

```
import pickle

pickle.dump(DT_model,open('Car prediction.pkl','wb'))
```

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 HTML file namely • home.html save it in Templates folder.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Payments Fraud Detection</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>

    <header>
        <h1>Online Payments Fraud Detection</h1>
    </header>

    <button id="home-button">Home</button>
    <a
    href="predict.html">
    <button id="predict-button">Predict</button>
    </a>

    <main>
        <p>The objective of this article is to predict online payments fraud given the various parameters. This will be a classification problem
    </main>

</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Payments Fraud Detection - Prediction Input</title>
    <link rel="stylesheet" type="text/css" href="style1.css">

</head>
<body>

    <header>
        <h1>Online Payments Fraud Detection </h1>
    </header>

    <main>
        <form>
            <label for="step">Step:</label>
            <input type="number" id="step" name="step" required><br>

            <label for="type">Type:</label>
            <input type="number" id="type" name="type" required><br>

            <label for="amount">Amount:</label>
            <input type="number" id="amount" name="amount" required><br>

            <label for="oldbalanceOrg">OldbalanceOrg:</label>
            <input type="number" id="oldbalanceOrg" name="oldbalanceOrg" required><br>

            <label for="newbalanceOrig">NewbalanceOrig:</label>
            <input type="number" id="newbalanceOrig" name="newbalanceOrig" required><br>

            <label for="oldbalanceDest">Oldbalance Dest:</label>
            <input type="number" id="oldbalanceDest" name="oldbalanceDest" required><br>

            <label for="newbalanceDest">NewbalanceDest:</label>
            <input type="number" id="newbalanceDest" name="newbalanceDest" required><br>

            <button type="submit">Submit</button>
        </form><br>

        <form action="result.html" method="GET">
            <button type="submit">Submit</button>
        </form>

    </main>
</body>
</html>
```

Activity 2: Build Python code:

Import the libraries

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

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (_name_) as argument.

```
app = Flask(__name__)

model = pickle.load(open('Car prediction.pkl','rb'))
```

Render HTML page:

```
@app.route('/')
def start():
    return render_template('index1.html')

@app.route('/login',methods =["POST","GET"])

def login():

   if request.method == "POST":
      age = request.form["age"]
      annual_income = request.form["annualincome"]
      gender = request.form["gender"]
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the carprediction.html page earlier.

```
t =  np.array([[age,annual_income,gender]])

t_scaled= scale.fit_transform(t)
output =model.predict(t_scaled)
print(output)

if (output == 1):
    return render_template('index1.html', output="Purchasble")
else:
    return render_template('index1.html', output="Not purchasble")
    return render_template("index1.html")

return render_template("index1.html")
```

Main Function:

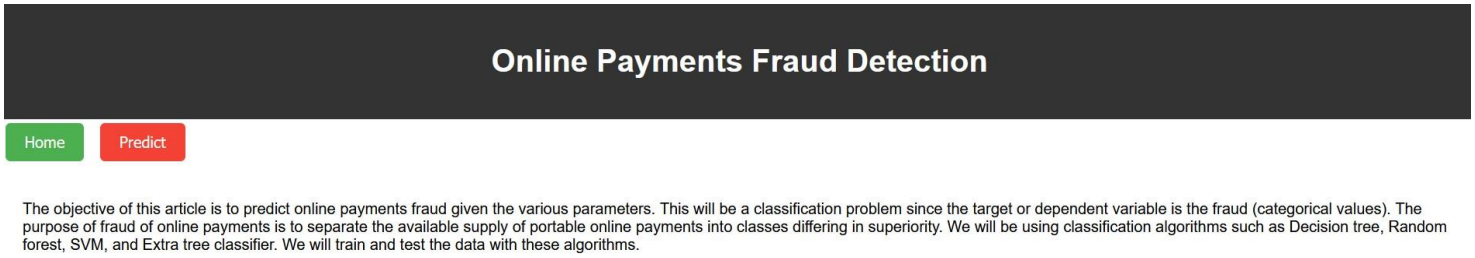
```
if __name__ == '__main__' :

    app.run(debug=True)
```

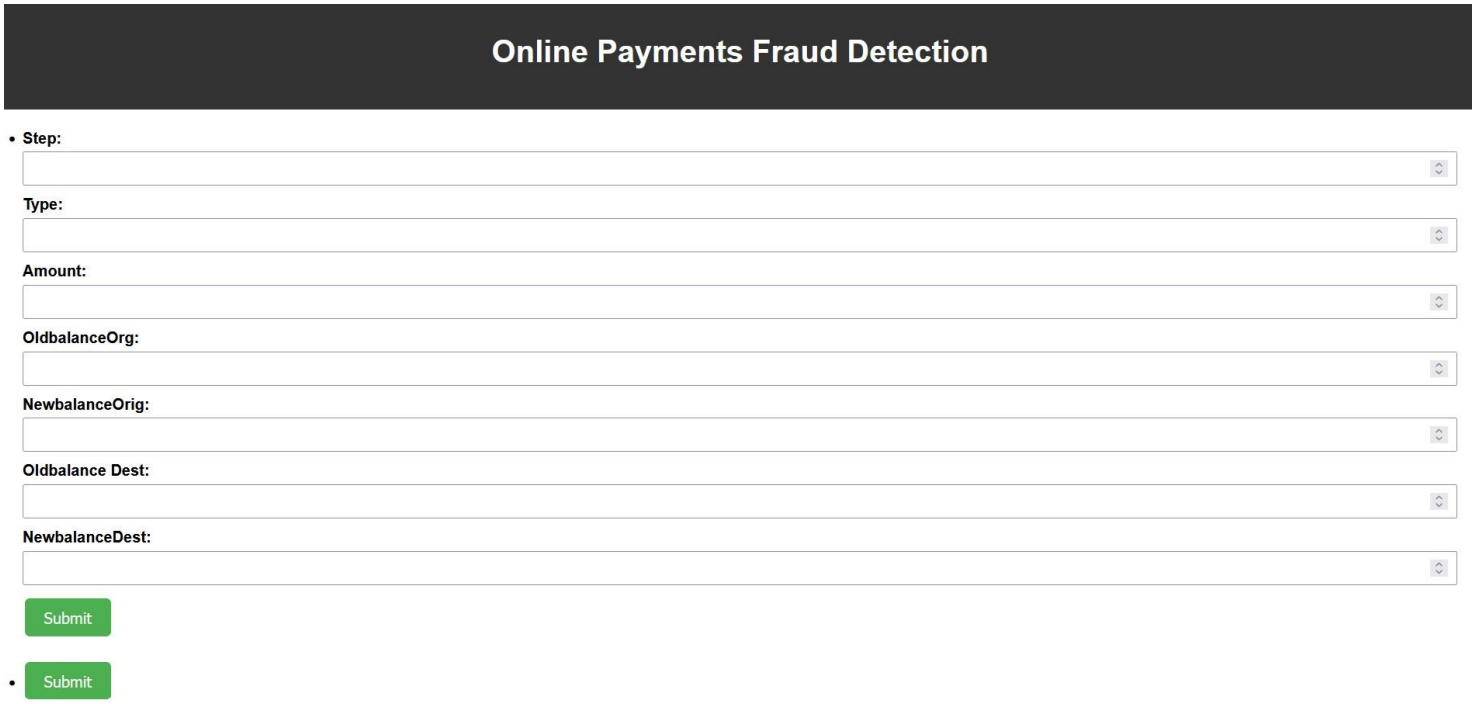
Activity 3: Run the application

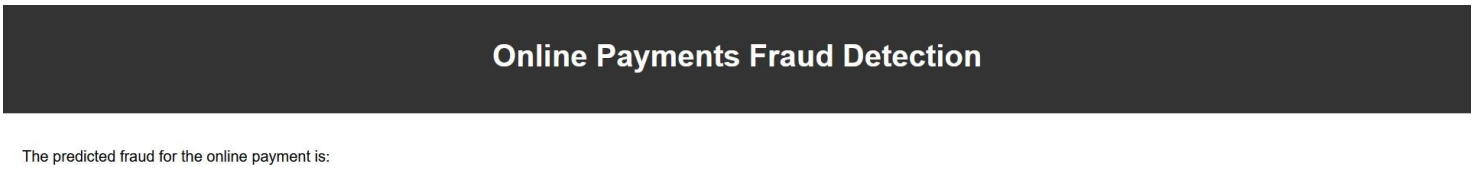
Open Visual studio code and Import all the project folders.

When you run the app.py file and click on the server url in terminal, you will redirected to home page. The home page will looks like:



Prediction page:





Conclusion:

In the realm of online payments, the integration of machine learning for fraud detection represents a crucial stride towards securing digital transactions. By delineating the problem scope, engaging stakeholders, and establishing clear success metrics, we lay the groundwork for a solution adept at identifying and preventing various forms of fraud, from credit card scams to identity theft. The dynamic and evolving nature of online fraud necessitates a solution that can adapt swiftly, striking a delicate balance between accuracy and efficiency to assure not only the integrity of financial transactions but also a seamless user experience.

Collaboration with key stakeholders, including financial institutions and payment service providers, ensures that the developed machine learning solution aligns with industry standards and user expectations. As we navigate the implementation of sophisticated algorithms and leverage diverse data sources, the commitment to transparency, ethical considerations, and legal compliance remains

paramount. Ultimately, the pursuit of excellence in online payments fraud detection using machine learning is not merely a technological advancement; it's a holistic approach that aims to fortify the foundations of digital transactions, instilling confidence among users and contributing to the resilience and security of the broader digital economy.