Online Payments Fraud Detection Using Machine Learning

Team Members :

- Kajal Pahil (22BCI0039)
- Souparnika Jayagopal (22BBS0089)
- Sonali P Sunny (22BCI0252)
- Sanjay S (22BEC0531)

Mentors:

- Machine Learning GP Mentor1
- Machine Learning GP Mentor2
- Kushal
- Siri
- Saumya
- Kaushik
- Revanth
- Manjunath

Online Payments Fraud Detection Using Machine Learning

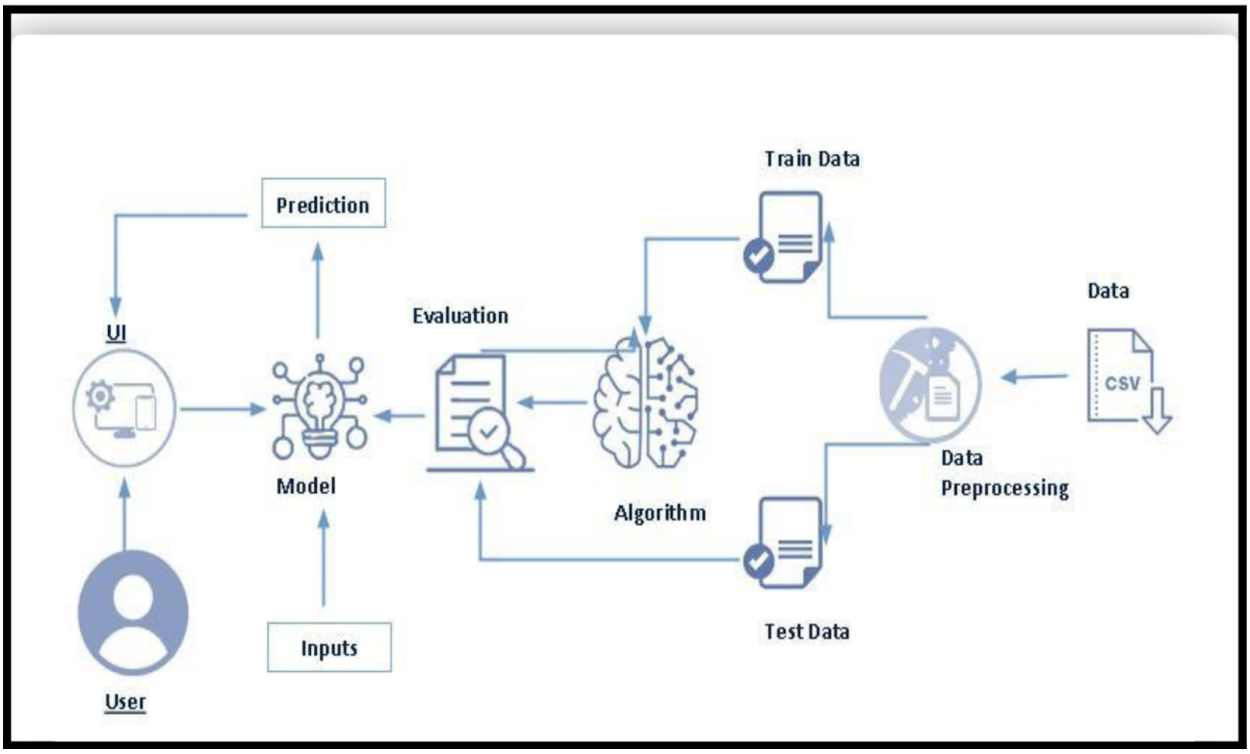
Online Payments Fraud Detection using Machine Learning is a proactive approach to identify and prevent fraudulent activities during online transactions. By leveraging historical transaction data, customer behavior patterns, and machine learning algorithms, this project aims to detect potential fraud in real time, ensuring secure and trustworthy online payment experiences for users and businesses alike.

Scenario 1: Real-time Fraud Monitoring:
The system continuously monitors online payment transactions in real time. By analyzing transaction features such as transaction amount, location, device information, and user behavior, it can flag suspicious transactions for further investigation, preventing fraudulent activities before they occur.

Scenario 2: Fraudulent Account Detection:
Machine learning models can detect patterns indicative of fraudulent accounts or activities. By analyzing user behavior over time, such as unusual login times, multiple failed login attempts, or sudden changes in spending patterns, the system can identify and block potentially fraudulent accounts, protecting legitimate users and businesses.

Scenario 3: Adaptive Fraud Prevention:
The system adapts and improves its fraud detection capabilities over time. By continuously learning from new data and adjusting its algorithms, it can stay ahead of evolving fraud techniques and trends, providing ongoing protection against online payment fraud for businesses and their customers.

Technical Architecture:



Project Flow:

- The user interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- The predictions made by the model are 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
- **Data pre-processing**
    - Removing unnecessary columns
    - Checking for null values
- **Visualizing and analyzing data**
    - Univariate analysis
    - Bivariate analysis
    - Descriptive analysis
- **Model building**
    - Handling categorical values
    - Dividing data into train and test sets
    - Import the model building libraries
    - Comparing the accuracy of various models
    - Hyperparameter tuning of the selected model
    - Evaluating the performance of models
    - Save the model
- **Application Building**
    - Create an HTML file
    - Build python code

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
https://youtu.be/QeKshry8pWQ

- Unsupervised learning:

https://www.javatpoint.com/unsupervised-machine-learning
https://youtu.be/D6gtZrsYi6c

- Decision tree:

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

- Random forest:

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

- Xgboost:

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

- ExtraTrees:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

- SupportVectorMachine:

https://scikit-learn.org/stable/modules/svm.html

- Evaluation metrics:

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
https://youtu.be/aWAnNHXIKww

- Flask Basics :

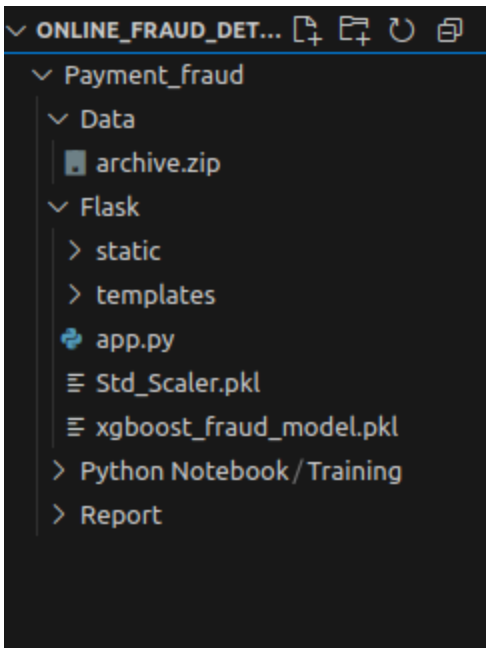
https://www.youtube.com/watch?v=lj4I_CvBnt0

- Literature review:

1. Nandhini, A. (2025). Online Payment Fraud Detection Using Machine Learning. *International Journal of Scientific Research in Engineering and Management*. doi:10.55041/ijsrem42092
2. Deo, S., Adnan, M., Raj, M., Abidi, A., & Bansal, N. (2024). Online Payment Fraud Transaction Detection Using Machine Learning. *2024 IEEE Delhi Section Flagship Conference (DELCON)*, 1-6. doi:10.1109/DELCON64804.2024.10866604
3. Admane, S., Gangrade, D., Patil, N., Nazirkar, S., & Thorat, M. (2023). Credit Card Fraud Detection Using Machine Learning. *International Research Journal of Modernization in Engineering Technology and Science*. doi:10.56726/irjmets33662
4. Anitha, V. (2025). A Survey on Online Payment Fraud Detection Techniques Using Machine Learning Algorithms. *International Journal for Research in Applied Science and Engineering Technology*. doi:10.22214/ijraset.2025.66490
5. Jeyachandran, P., Akisetty, A. S. V. V., Subramani, P., Goel, O., Singh, S. P., & Shrivastav, A. (2024). Leveraging Machine Learning for Real-Time Fraud Detection in Digital Payments. *Integrated Journal for Research in Arts and Humanities*. doi:10.55544/ijrah.4.6.10

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.
- xgboost_fraud_model.pkl is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Dataset used
- The Notebook file contains procedure for building the model.

Milestone 1: Define Problem / Problem Understanding

Business problem:

The project focuses on detecting fraudulent activities in online payments using machine learning. The aim is to identify fraudulent transactions effectively and efficiently to minimize financial losses and improve trust among users and businesses. This involves analyzing transaction patterns, identifying anomalies, and building models to predict fraudulent behavior.

Business requirements:

Accuracy and Reliability

- The system should use the most accurate and reliable data to ensure high precision in detecting fraudulent transactions and minimizing false positives.

Real-Time Detection

- The system should operate in real-time or near-real-time to identify and block fraud before it is executed.

Scalability

- The fraud detection system should be scalable to handle the growing volume of online payment transactions.

Compliance

- The project must comply with legal and regulatory standards for data privacy and financial transactions, such as PCI DSS and GDPR.

Explainability

- The machine learning model should provide insights into why a transaction is flagged as fraudulent, enabling user trust and system debugging.

User-Friendly Dashboard

- The interface should allow business users and fraud analysts to understand the flagged transactions and investigate further.

Integration Flexibility

- The system should be compatible with existing payment gateways and transaction platforms.

Activity 3: Literature Survey :

Recent advancements in machine learning (ML) have significantly improved online payment fraud detection by addressing challenges like imbalanced datasets and evolving fraud patterns.

Ensemble methods, particularly Random Forest and XGBoost, have emerged as highly effective in detecting fraudulent transactions due to their precision and ability to handle complex patterns (Nandhini, 2025). Studies on real-world datasets, such as those by Deo et al. (2024), demonstrate ML models achieving up to 95% accuracy, underscoring their practical utility.

Researchers have also highlighted the importance of addressing skewed datasets to improve sensitivity and specificity, as seen in work by Admane et al. (2023). Comparative analyses further reveal the computational efficiency and superior accuracy of ensemble methods over traditional algorithms (Anitha, 2025).

Additionally, future directions in fraud detection emphasize real-time analytics and the integration of explainable AI and blockchain for scalable, adaptive solutions (Jeyachandran et al., 2024).

Activity 4: Social or Business Impact:

Social Impact

- **Enhanced Trust in Digital Payments**:
  By reducing fraudulent activities, the system builds confidence among users in online payment platforms.
- **Financial Security**:
  Reducing fraud minimizes financial loss for users and businesses, contributing to a more secure financial ecosystem.

Business Model/Impact

- **Reduced Operational Costs**:
  Automating fraud detection lowers the need for manual reviews, saving time and resources.
- **Improved User Retention**:
  Users are more likely to continue using platforms that provide secure transaction services.
- **Revenue Protection**:
  Preventing fraud protects the business’s revenue and reputation.

Milestone 2: Data Collection & Preparation

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.

Activity 1: Collect 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-dataset

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 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

```
[ ]  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
```

Note: Renamed ***PS_20174392719_1491204439457_logs.csv*** to ***fraud.csv*** for ease.

Activity 1.2: Read the Dataset

In the dataset *fraud.csv*, the classes were imbalanced — the majority of transactions were labeled as non-fraudulent (isFraud = 0), while only a small fraction were fraudulent (isFraud = 1). To ensure that the machine learning model didn't become biased toward the majority class, the dataset was balanced using downsampling.

Steps Performed:

- **Loaded the dataset:**
  We read the *fraud.csv* file into a DataFrame using pandas.

- **Separated the classes:**
  We split the data into two subsets: one containing all non-fraudulent transactions (isFraud = 0), and another containing all fraudulent transactions (isFraud = 1).

- **Downsampled the majority class:**
  We randomly selected 10,000 non-fraudulent samples without replacement using sklearn.utils.resample() to reduce the size of the majority class.

- **Combined the classes:**
  We combined the downsampled non-fraudulent data with all the fraudulent samples to create a balanced dataset.

- **Shuffled the data:**
  We shuffled the combined dataset to eliminate any ordering bias and reset the index.

- **Saved the balanced dataset:**
  Finally, we saved the resulting balanced dataset as balanced_fraud_dataset.csv for use in training and evaluating machine learning models.

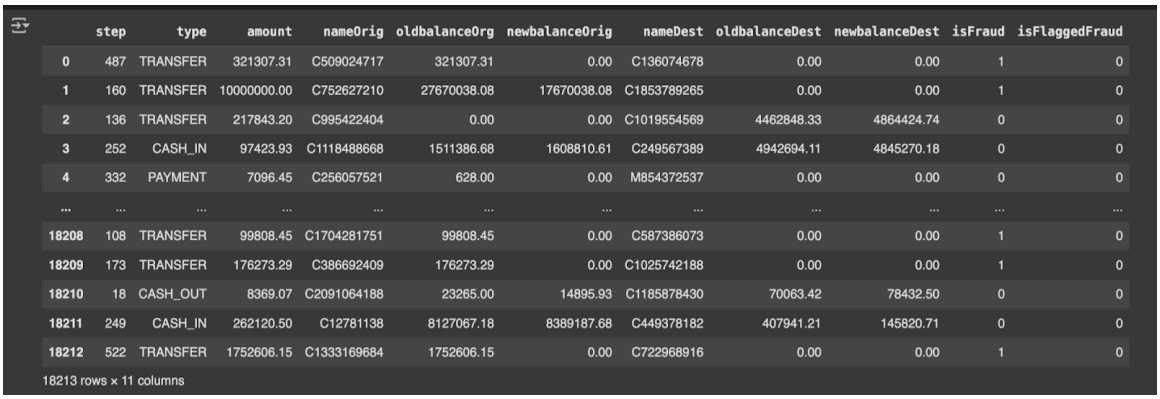
```
[ ]  import pandas as pd
     from sklearn.utils import resample

     df = pd.read_csv("fraud.csv")

     df_not_fraud = df[df['isFraud'] == 0]
     df_fraud = df[df['isFraud'] == 1]

     df_not_fraud_downsampled = resample(
         df_not_fraud,
         replace=False,
         n_samples=10000,
         random_state=42
     )

     df_balanced = pd.concat([df_not_fraud_downsampled, df_fraud])

     df_balanced = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

     df_balanced.to_csv("balanced_fraud_dataset.csv", index=False)
```

Activity 1.3: 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 the csv file.

```
[ ]  df=pd.read_csv("balanced_fraud_dataset.csv")
     df
```



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	487	TRANSFER	321307.31	C509024717	321307.31	0.00	C136074678	0.00	0.00	1	0
1	160	TRANSFER	10000000.00	C752627210	27670038.08	17670038.08	C1853789265	0.00	0.00	1	0
2	136	TRANSFER	217843.20	C995422404	0.00	0.00	C1019554569	4462848.33	4864424.74	0	0
3	252	CASH_IN	97423.93	C1118488668	1511386.68	1608810.61	C249567389	4942694.11	4845270.18	0	0
4	332	PAYMENT	7096.45	C256057521	628.00	0.00	M854372537	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
18208	108	TRANSFER	99808.45	C1704281751	99808.45	0.00	C587386073	0.00	0.00	1	0
18209	173	TRANSFER	176273.29	C386692409	176273.29	0.00	C1025742188	0.00	0.00	1	0
18210	18	CASH_OUT	8369.07	C2091064188	23265.00	14895.93	C1185878430	70063.42	78432.50	0	0
18211	249	CASH_IN	262120.50	C12781138	8127067.18	8389187.68	C449378182	407941.21	145820.71	0	0
18212	522	TRANSFER	1752606.15	C1333169684	1752606.15	0.00	C722968916	0.00	0.00	1	0

18213 rows × 11 columns

Activity 2: Data Preparation

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 Outliers

Activity 2.1: Handling missing values

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

```
[ ]  #finding all null values
     df.isnull().sum()
```

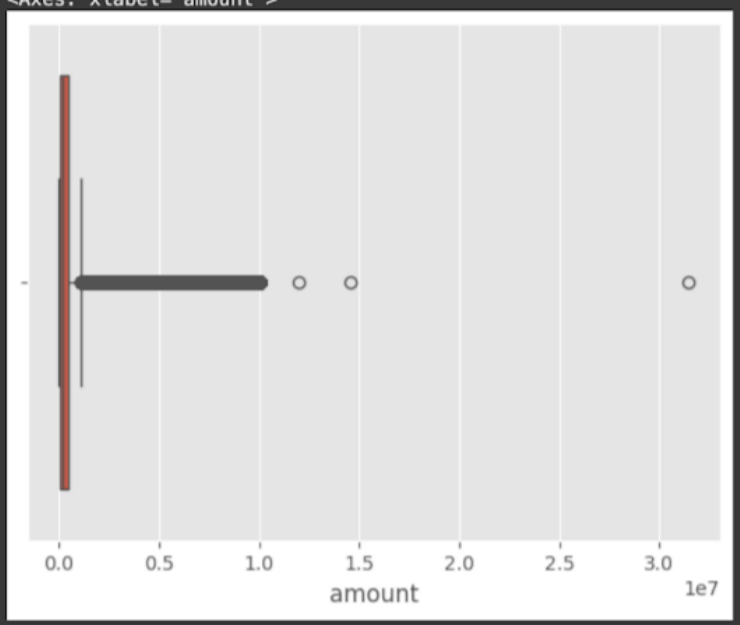
	0
step	0
type	0
amount	0
nameOrig	0
oldbalanceOrg	0
newbalanceOrig	0
nameDest	0
oldbalanceDest	0
newbalanceDest	0
isFraud	0

dtype: int64

Activity 2.2: Handling Outliers

Here, a box-plot is used to identify outliers in the dataset's amount attribute.

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



To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, we have to subtract it with 1st quantile. Shown in image attached below.

remove the outliers ->

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

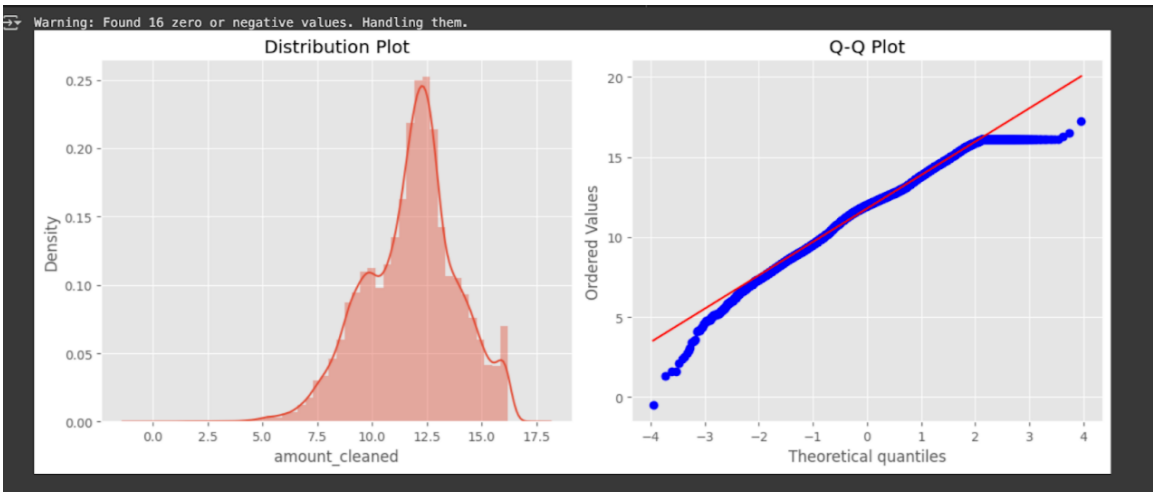
```
ModeResult(mode=np.float64(10000000.0), count=np.int64(289))
760732.1569433921
```

```
[ ] 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('q1 :',q1)
    print('q3 :',q3)
    print( 'IQR :', 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]))
```

```
q1 : 32767.58
q3 : 466422.84
IQR : 433655.26
Upper Bound : 1116905.73
Lower Bound : -617715.31
Skewed data : 2716
Skewed data : 0
```

To handle the outliers transformation technique is used. Here, transformationPlot is used to plot the dataset's outliers for the amount property.

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats

def transformationPlot(feature):
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    sns.distplot(feature, kde=True)
    plt.title("Distribution Plot")

    plt.subplot(1, 2, 2)
    stats.probplot(feature, plot=plt)
    plt.title("Q-Q Plot")

    plt.tight_layout()
    plt.show()

invalid_count = (df['amount'] <= 0).sum()
if invalid_count > 0:
    print(f"Warning: Found {invalid_count} zero or negative values. Handling them.")

df['amount_cleaned'] = df['amount'].apply(lambda x: x if x > 0 else np.nan)

df_cleaned = df.dropna(subset=['amount_cleaned', 'isFraud'])

log_transformed = np.log(df_cleaned['amount_cleaned'])

transformationPlot(log_transformed)
```



Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

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.

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
count	18213.000000	18213	1.821300e+04	18213	1.821300e+04	1.821300e+04	18213	1.821300e+04	1.821300e+04	18213
unique	NaN	5	NaN	18213	NaN	NaN	17972	NaN	NaN	2
top	NaN	CASH_OUT	NaN	C1333169684	NaN	NaN	C1338686176	NaN	NaN	is not Fraud
freq	NaN	7719	NaN	1	NaN	NaN	3	NaN	NaN	10000
mean	299.952781	NaN	7.607322e+05	NaN	1.203497e+06	5.588933e+05	NaN	8.499842e+05	1.248386e+06	NaN
std	189.979179	NaN	1.785152e+06	NaN	3.240780e+06	2.575574e+06	NaN	3.437453e+06	3.801429e+06	NaN
min	1.000000	NaN	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
25%	160.000000	NaN	3.276758e+04	NaN	6.460000e+03	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
50%	280.000000	NaN	1.616653e+05	NaN	1.015620e+05	0.000000e+00	NaN	0.000000e+00	1.363052e+05	NaN
75%	402.000000	NaN	4.664228e+05	NaN	6.991836e+05	0.000000e+00	NaN	5.645954e+05	1.090220e+06	NaN
max	743.000000	NaN	3.147911e+07	NaN	5.958504e+07	4.958504e+07	NaN	2.362305e+08	2.367265e+08	NaN

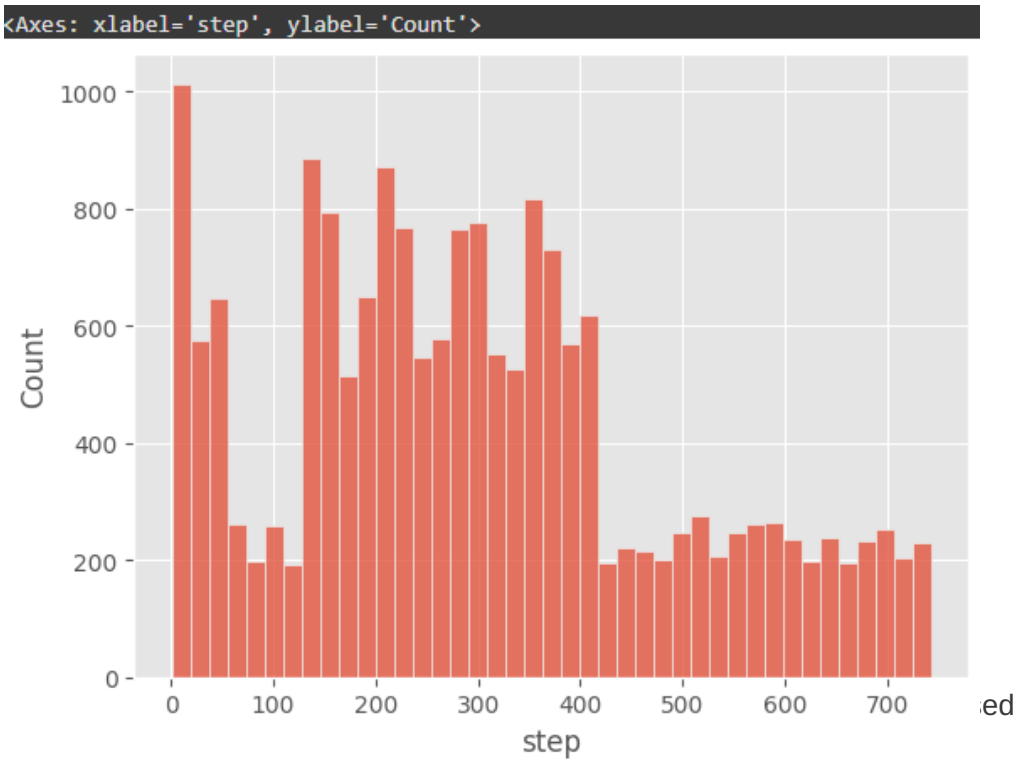
Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

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

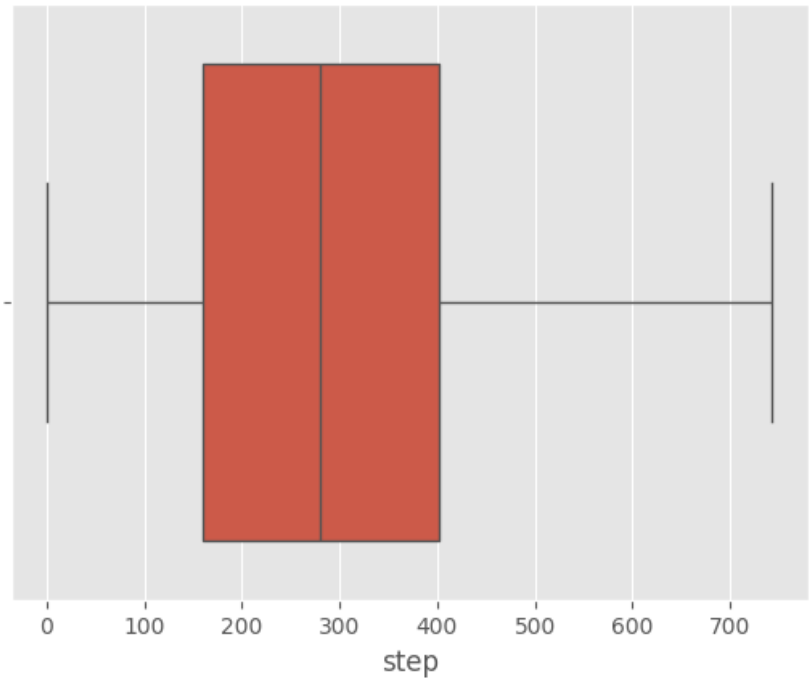
The distribution of one or more variables is represented by a histogram, a traditional visualisation tool, by counting the number of observations that fall within.



This boxplot displays the distribution of the step attribute, which likely represents time steps or transaction time intervals in the dataset.

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

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

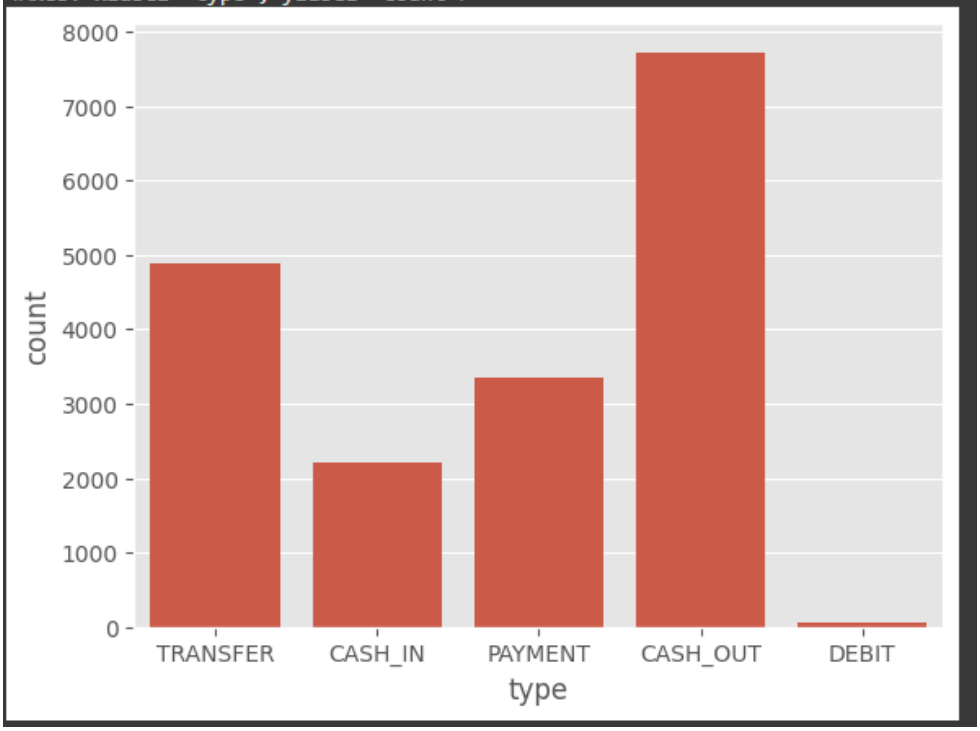


- The majority of transactions are concentrated between step 150 and 450.
- There are no significant outliers, and the distribution appears symmetric and well spread across time.

Here this countplot visualizes the distribution of transaction types (type) in the dataset. Each bar represents the frequency of a transaction category.

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

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

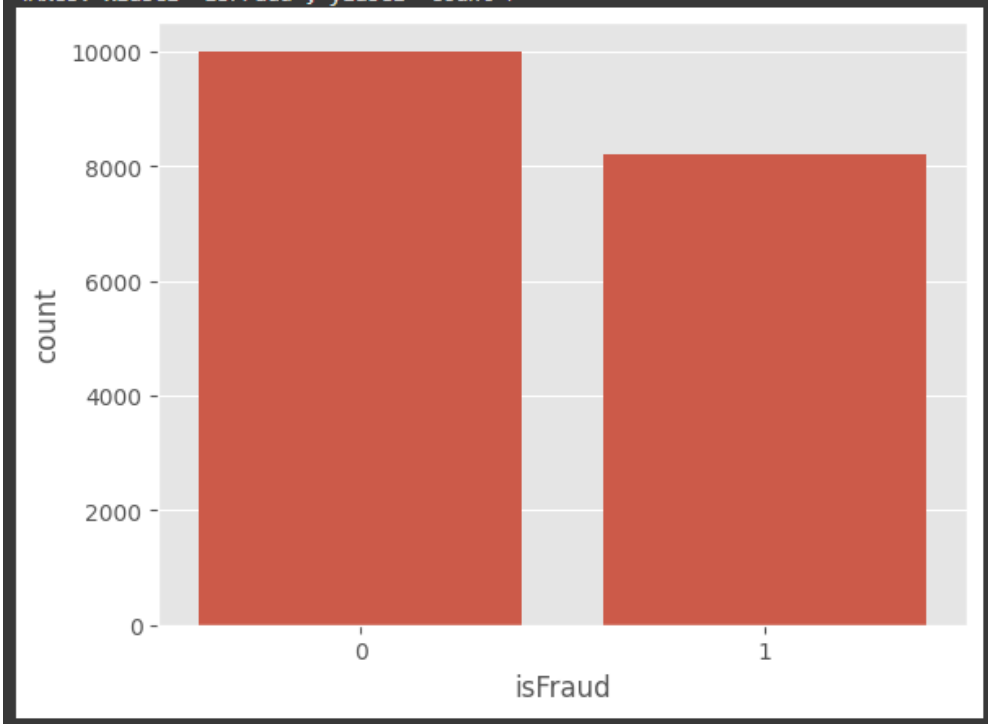


- CASH_OUT and TRANSFER are the most frequent transaction types.
- DEBIT is extremely rare, indicating it may be an outlier or infrequent class.

Using the countplot approach here to count the number of instances in the dataset's target isFraud column.

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

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




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

isFraud	count
0	10000
1	8213

dtype: int64

- The dataset is slightly imbalanced with more non-fraudulent (10,000) than fraudulent (8,213) transactions.
- While not extremely imbalanced, this could still impact model performance and should be addressed.

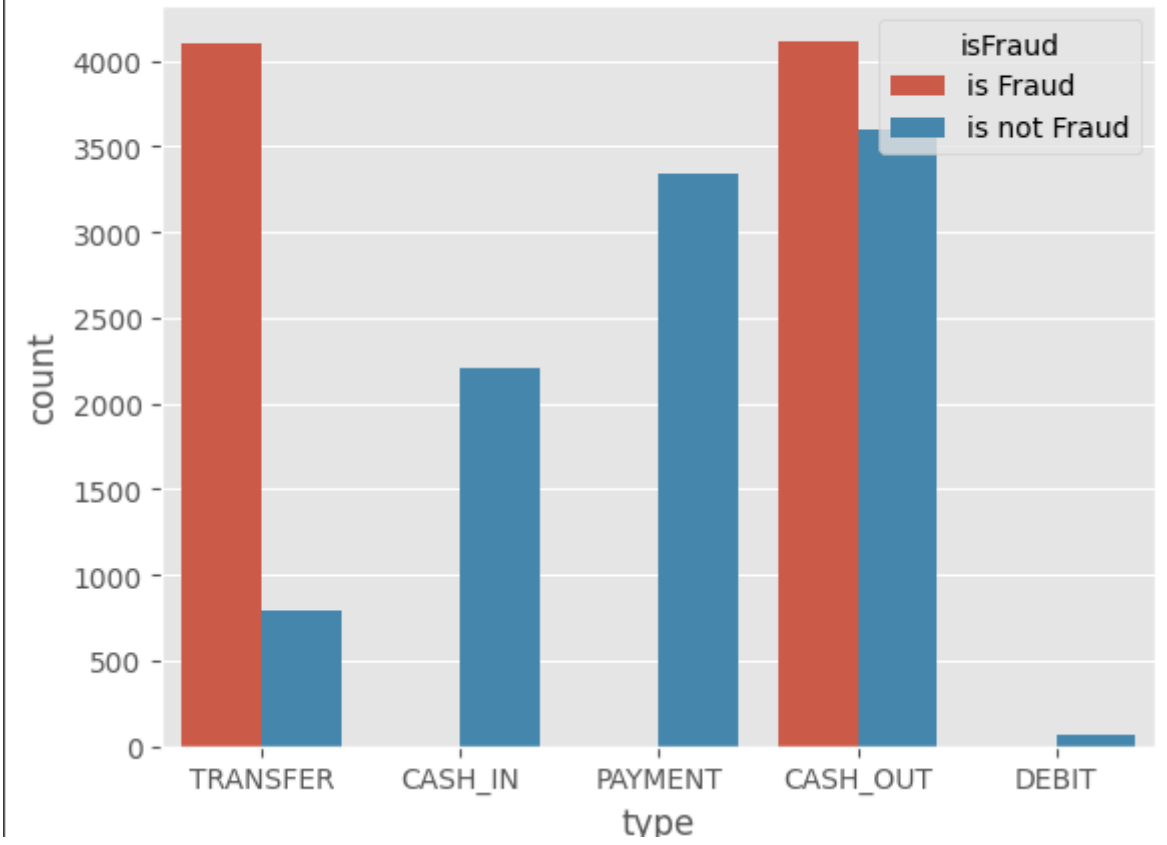
Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between newbalanceDest and isFraud.

Here we are visualising the relationship between type and isFraud.countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

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

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



- TRANSFER and CASH_OUT have significant fraudulent activity, as seen by the red bars.
- CASH_IN, PAYMENT, and DEBIT show almost no fraud, suggesting they are safer transaction types in this dataset.

Here we are visualising the relationship between isFraud and step.boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

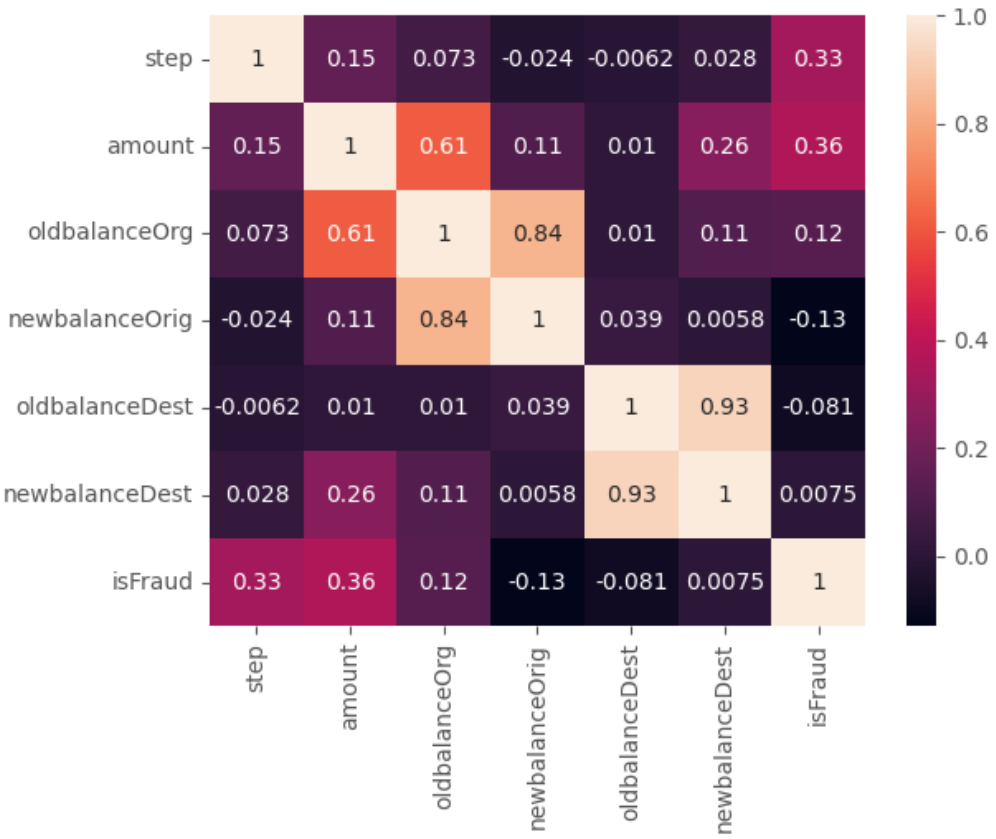
```
sns.boxplot(data=df,x='isFraud',y='step')
```

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

- Fraudulent transactions are more spread across time and tend to occur later (step values are higher on average).
- Non-fraudulent transactions are more concentrated earlier in time with visible outliers, suggesting a potential time-based fraud pattern.

Activity 2.3: Multivariate analysis

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



- From the below image, we can conclude that there are some features with strong correlations.
- The correlation between oldbalanceOrg and newbalanceOrig is high, with a value of **0.84**, indicating these features may be redundant.
- Similarly, oldbalanceDest and newbalanceDest are also highly correlated (**0.93**), suggesting overlapping information.
- The target variable isFraud shows moderate correlation with step (**0.33**) and amount (**0.36**), which could be important features for prediction.
- Highly correlated features like oldbalanceOrg with newbalanceOrig, and oldbalanceDest with newbalanceDest, should be reviewed for multicollinearity and potentially dropped or combined.

Encoding the Categorical Features:

• The categorical Features are can't be passed directly to the Machine Learning Model. So we convert them into Numerical data based on their order. This Technique is called Encoding.
• Here we are applying fit_transform to transform the categorical features to numerical features.

```
from sklearn.preprocessing import LabelEncoder
la = LabelEncoder()
df['type']=la.fit_transform(df['type'])
df['type'].value_counts()
```

type	count
1	7719
4	4888
3	3342
0	2202
2	62

dtype: int64

Splitting data into train and test

Now let's split the Dataset into train and test setsChanges: 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 my target variable is passed. For splitting training and testing data we are using the 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_train.shape)
print(y_test.shape)
```

Scaling

• Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the scale of the input features.

• Here we are using Standard Scaler.

• This scales the data to have a mean of 0 and a standard deviation of 1. The formula is given by: X_scaled = (X - X_mean) / X_std

```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()

x_train = std_scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train,columns=x.columns)

x_test = std_scaler.transform(x_test)
x_test = pd.DataFrame(x_test,columns=x.columns)

import joblib
joblib.dump(std_scaler, 'scaler.pkl')
```

Milestone 4: Model Building

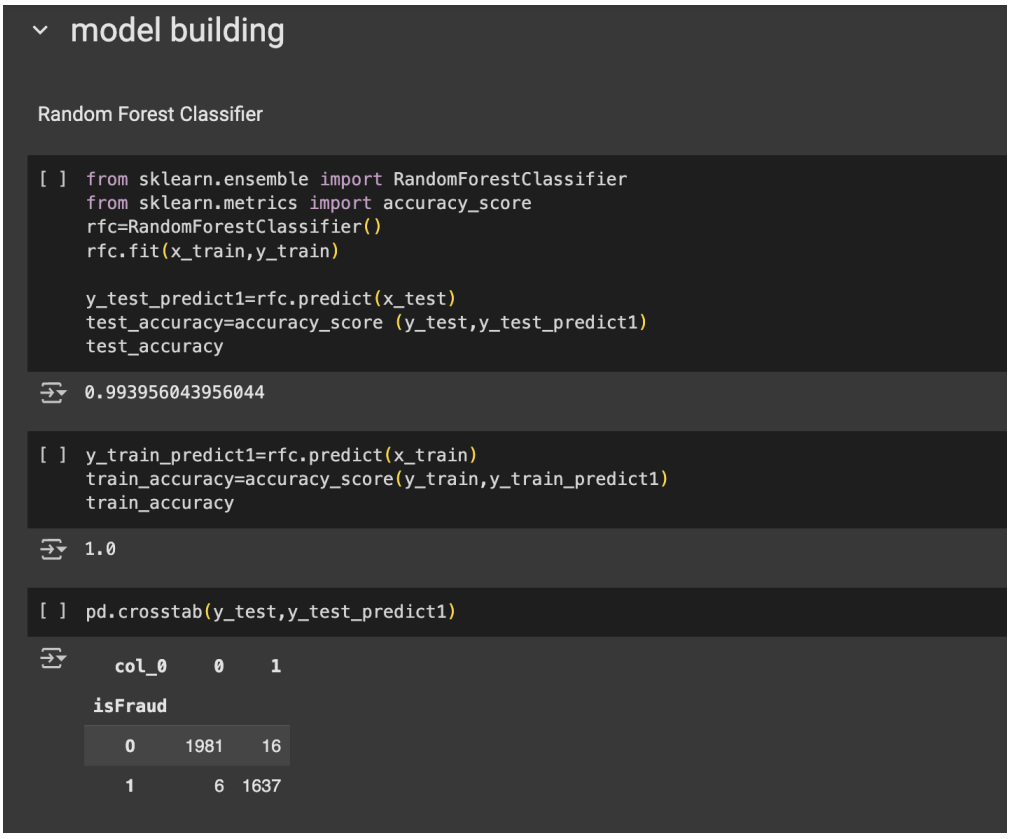
Activity 1: Training the model in multiple algorithms

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 1.1: 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.



```
[ ]  print(classification_report(y_test,y_test_predict1))
```

```
                precision    recall  f1-score   support

             0       1.00      0.99      0.99      1997
             1       0.99      1.00      0.99      1643

      accuracy                           0.99      3640
     macro avg       0.99      0.99      0.99      3640
  weighted avg       0.99      0.99      0.99      3640
```

Activity 1.2: 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.

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
```

```
0.9898351648351649
```

```
[ ]  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	0	1
isFraud		
0	1979	18
1	19	1624

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

```
                precision    recall  f1-score   support

             0       0.99      0.99      0.99      1997
             1       0.99      0.99      0.99      1643

      accuracy                           0.99      3640
     macro avg       0.99      0.99      0.99      3640
  weighted avg       0.99      0.99      0.99      3640
```

Activity 1.3: Extra Trees 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.

ExtraTrees Classifier

```
[ ] 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
```

```
0.9895604395604396
```

```
[ ] 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	0	1
isFraud		
0	1976	21
1	17	1626

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

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1997
           1       0.99      0.99      0.99      1643

    accuracy                           0.99      3640
   macro avg       0.99      0.99      0.99      3640
weighted avg       0.99      0.99      0.99      3640
```

Activity 1.3: Support Vector Machine 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.

SupportVectorMachine 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
```

```
0.910989010989011
```

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

```
0.9106271896682009
```

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

col_0	0	1
isFraud		
0	1932	65
1	259	1384

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

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      1997
           1       0.96      0.84      0.90      1643

    accuracy                           0.91      3640
   macro avg       0.92      0.90      0.91      3640
weighted avg       0.91      0.91      0.91      3640
```

Activity 1.4: 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.

Xgboost Classifier

```
[ ] import xgboost as xgb
    xgb1 = xgb.XGBClassifier()
    xgb1.fit(x_train, y_train)
    y_test_predict5=xgb1.predict(x_test)
    test_accuracy=accuracy_score(y_test,y_test_predict5)
    test_accuracy
```

```
0.9950549450549451
```

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

```
1.0
```

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

col_0	0	1
isFraud		
0	1985	12
1	6	1637

```
[ ] print(classification_report(y_test,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      1997
           1       0.99      1.00      0.99      1643

    accuracy                           1.00      3640
   macro avg       0.99      1.00      1.00      3640
weighted avg       1.00      1.00      1.00      3640
```

Activity 2: Comparing the Models

For comparing the above four models, the compareModel function is defined.

After calling the function, the results of models are displayed as output. From the five models, the XgBoost model was performing the best.

Compare the 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_train))
        print("test accuracy for xgb1", accuracy_score(y_test_predict5,y_test))
```

```
[ ] compareModel()
```

```
train accuracy for rfc 1.0
test accuracy for rfc 0.993956043956044
train accuracy for dtc 1.0
test accuracy for dtc 0.9898351648351649
train accuracy for etc 1.0
test accuracy for etc 0.9895604395604396
train accuracy for svc 0.9106271896682009
test accuracy for svcc 0.910989010989011
train accuracy for xgb1 1.0
test accuracy for xgb1 0.9950549450549451
```

Activity 3: Evaluating Performance

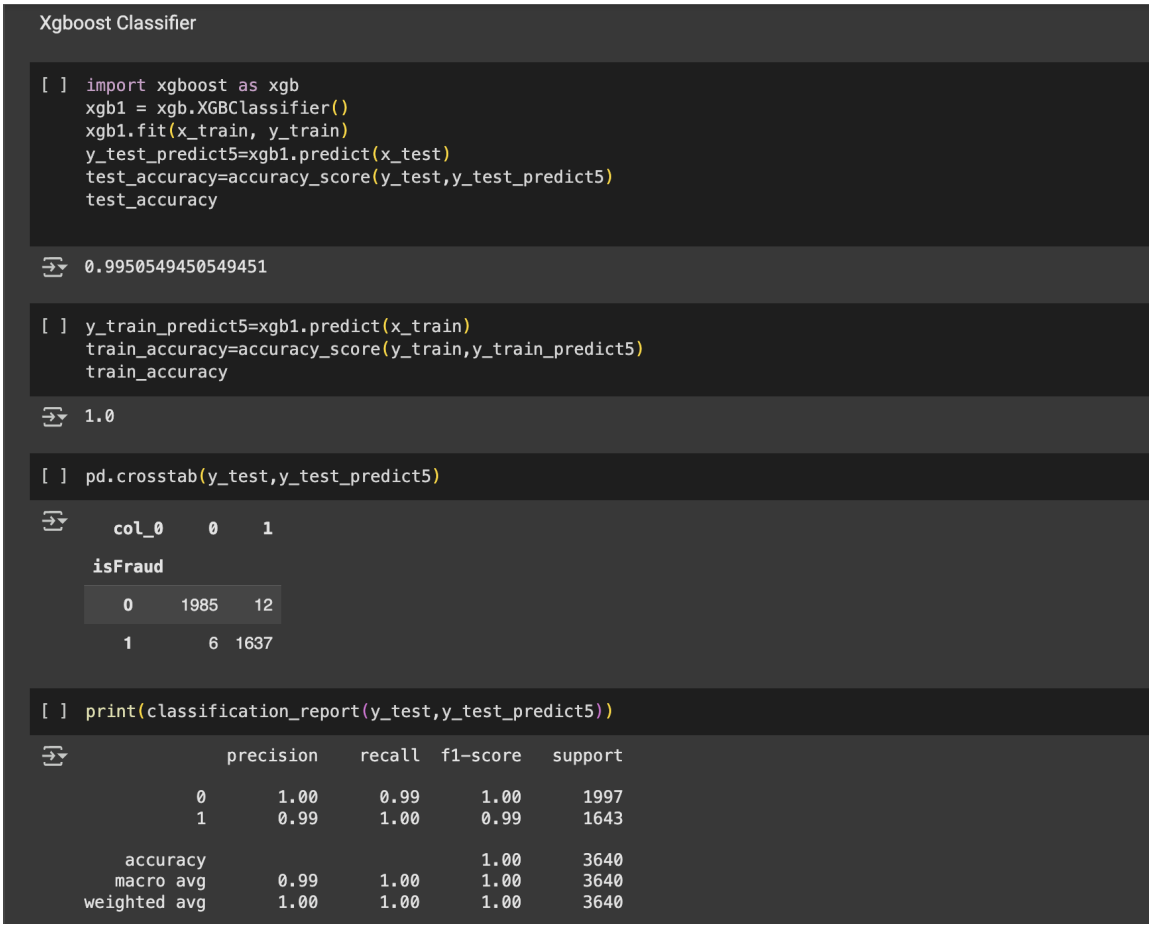
From sklearn, accuracy_score is used to evaluate the score of the model.

The XGBoost classifier was trained on the processed dataset and evaluated using both the training and test data. It achieved a high test accuracy of 99.51%, indicating strong performance on unseen data. The training accuracy was 100%, which could suggest overfitting; however, the model still generalized well based on the test performance.

From the confusion matrix, the model correctly predicted 1985 non-fraud cases and 1637 fraud cases, with only 12 false positives and 6 false negatives.

The classification report further confirms the model's effectiveness, with precision, recall, and F1-scores all close to 1.00 for both classes.

These results highlight that XGBoost is a highly effective model for fraud detection in this dataset.



Xgboost Classifier

```
[ ] import xgboost as xgb
    xgb1 = xgb.XGBClassifier()
    xgb1.fit(x_train, y_train)
    y_test_predict5=xgb1.predict(x_test)
    test_accuracy=accuracy_score(y_test,y_test_predict5)
    test_accuracy
```

```
0.9950549450549451
```

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

```
1.0
```

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

col_0	0	1
isFraud		
0	1985	12
1	6	1637

```
[ ] print(classification_report(y_test,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      1997
           1       0.99      1.00      0.99      1643

    accuracy                           1.00      3640
   macro avg       0.99      1.00      1.00      3640
weighted avg       1.00      1.00      1.00      3640
```

Milestone 5: Model Deployment

Activity 1: Saving the Model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

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

Saving the model

```
[ ]  import pickle

     # Save the trained model
     pickle.dump(xgb1, open('xgboost_fraud_model.pkl', 'wb'))
```

Activity 2: 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

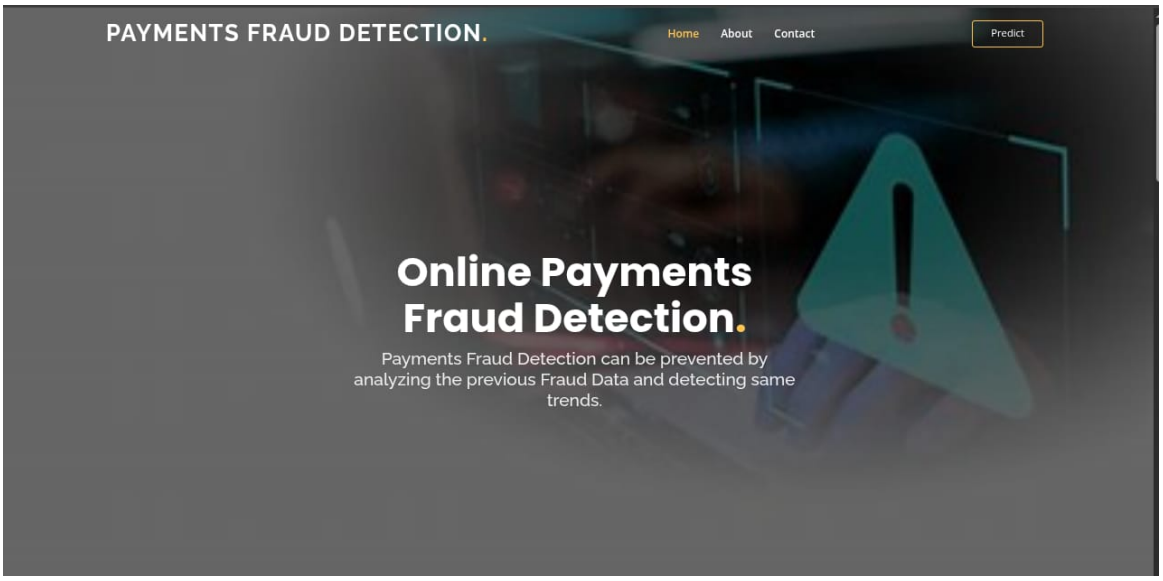
Activity 2.1: Building HTML Pages

For this project, two HTML files were created namely -

home.html

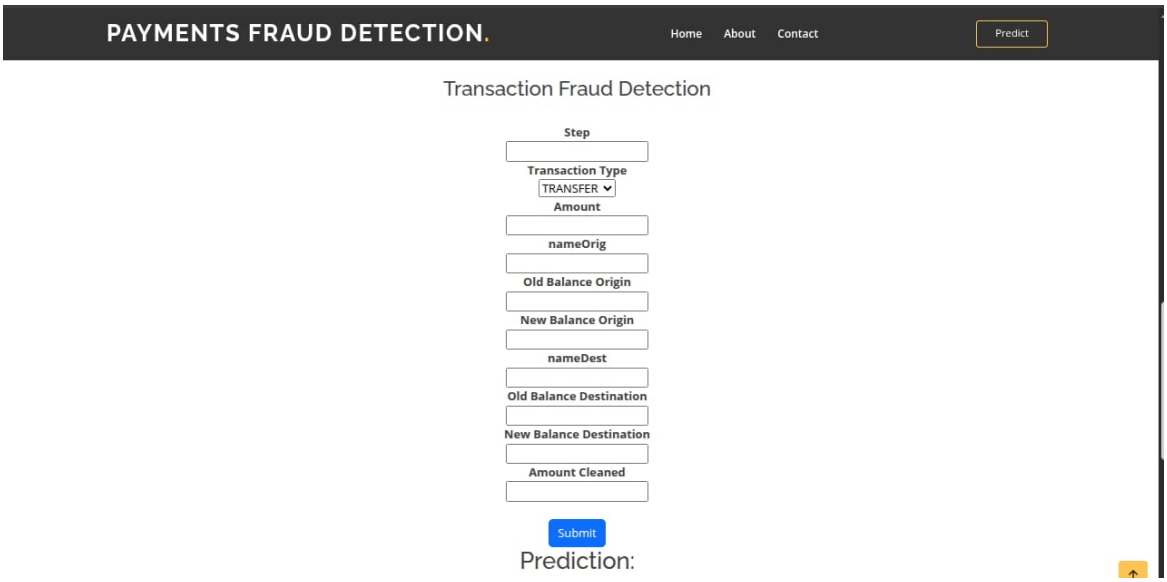
predict.html

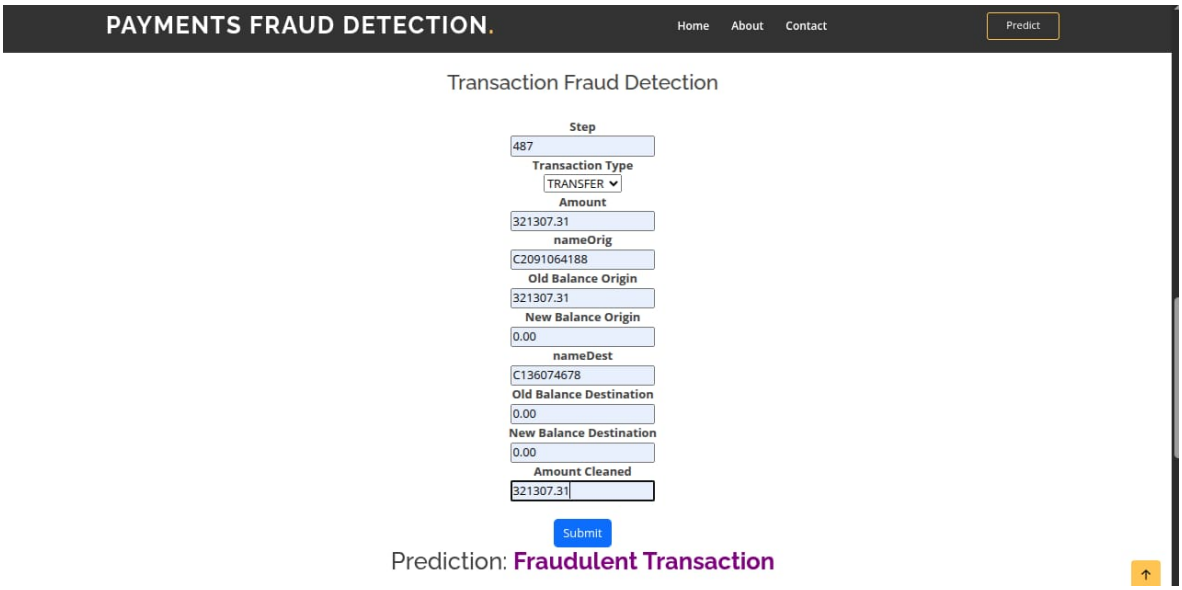
This is how our home.html page looks like:



Now when you click on predict button from top right corner you will get redirected to predict.html.

This is how our predict.html file looks like:





Activity 2.2: Building Python Code

Import the libraries

```
from flask import Flask, render_template, request
import pickle

model = pickle.load(open("xgboost_fraud_model.pkl", "rb"))

app = Flask(__name__)

```

Load the saved model. Importing the 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.route('/')
def welcome():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    step = float(request.form['step'])
    trans_type = request.form['type']
    amount = float(request.form['amount'])
    nameOrig = request.form['nameOrig']
    oldbalanceOrg = float(request.form['oldbalanceOrg'])
    newbalanceOrig = float(request.form['newbalanceOrig'])
    nameDest = request.form['nameDest']
    oldbalanceDest = float(request.form['oldbalanceDest'])
    newbalanceDest = float(request.form['newbalanceDest'])
    amount_cleaned = float(request.form['amount_cleaned'])
```

Codes for rendering HTML page and retreiving the value from UI is shown above.. 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.

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 be rendered to the text that we have mentioned in the submit.html page earlier.

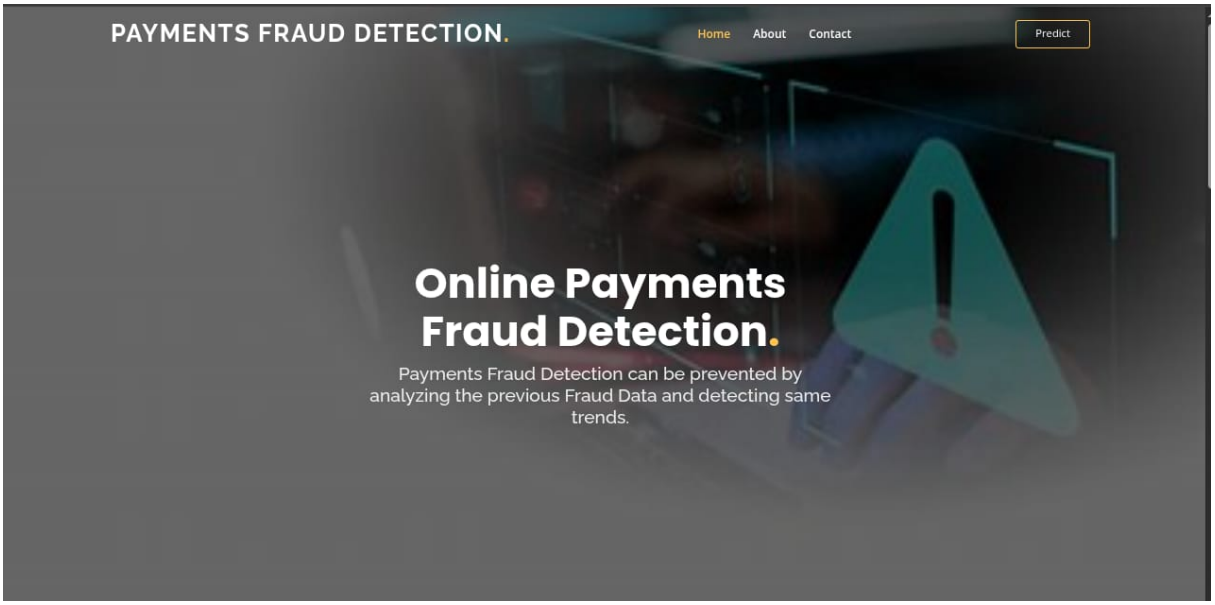
Main Function:

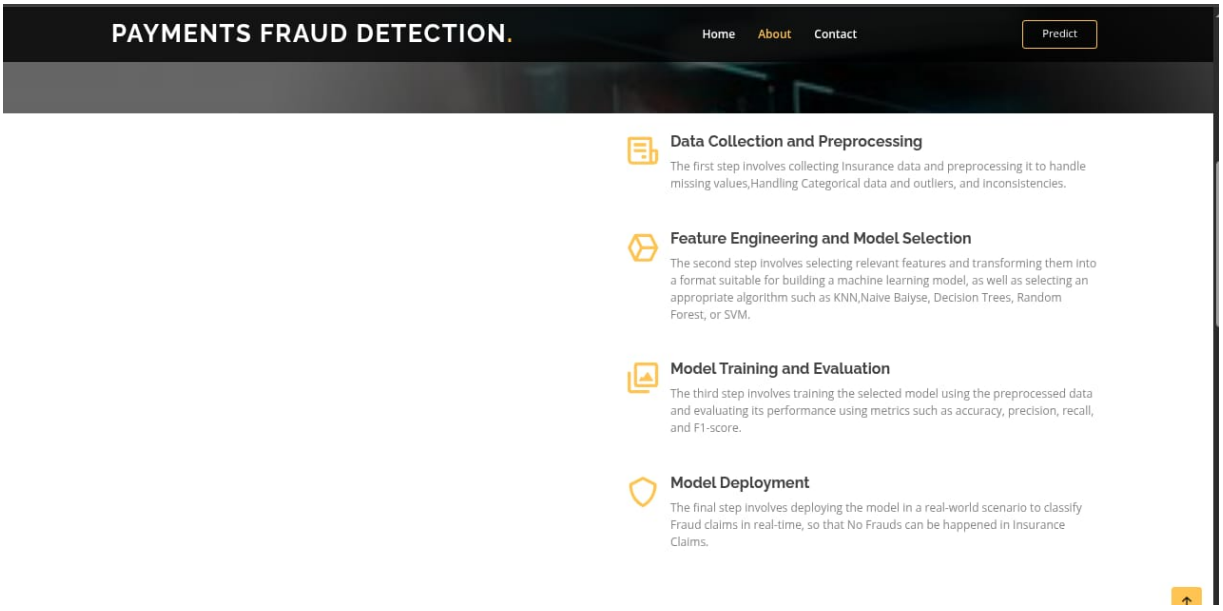
```
    features = [[
        step, type_encoded, amount, nameOrig_encoded, oldbalanceOrg,
        newbalanceOrig, nameDest_encoded, oldbalanceDest, newbalanceDest, amount_cleaned
    ]]

    prediction = int(model.predict(features)[0])
    result = "Fraudulent Transaction" if prediction == 1 else "Legitimate Transaction"
    return render_template('index.html', predict=result)

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

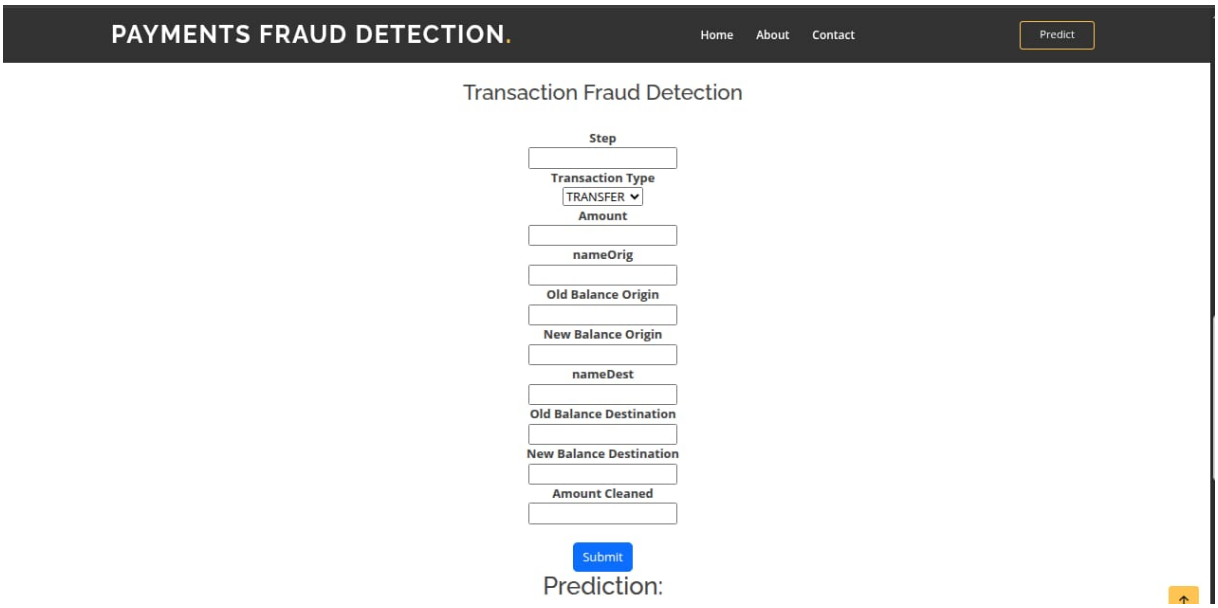
Activity 2.3: Running the Application

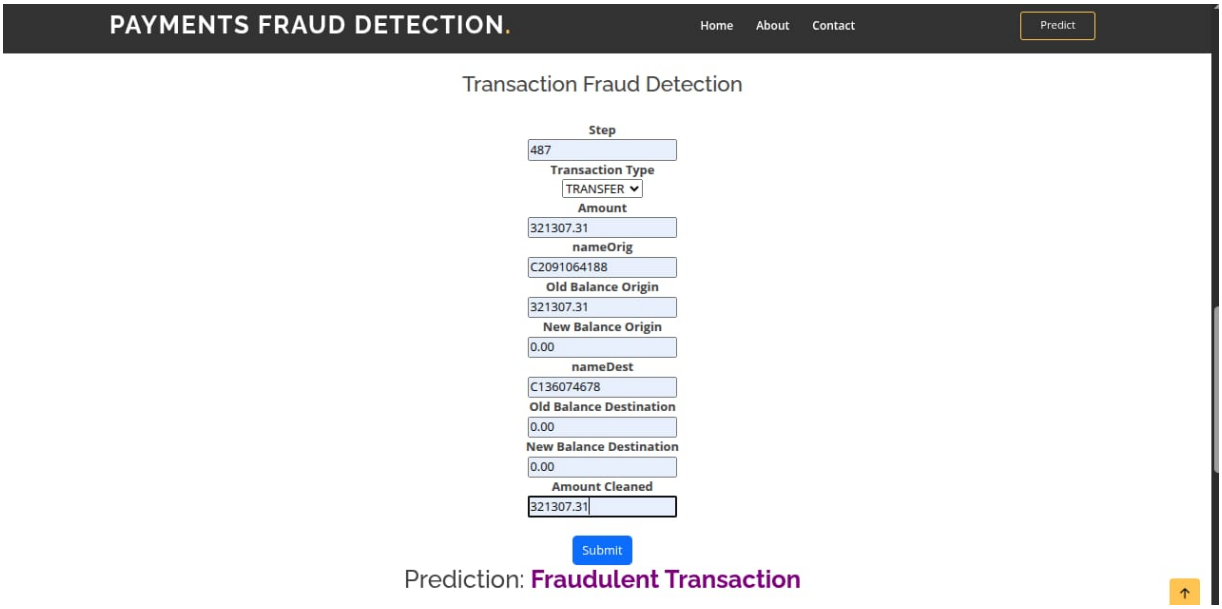
```
^C(venv) souparnika@souparnika-ThinkPad-L450:~/Downloads/ML_Auto_Insurance_Fraud_Detection_using_Machine_Learning/Insurance_fraud/
Flask$ python3 app.py
/home/souparnika/Downloads/ML_Auto_Insurance_Fraud_Detection_using_Machine_Learning/Insurance_fraud/Flask/app.py:4: UserWarning: [
14:47:36] WARNING: /workspace/src/collective/../data/../common/error_msg.h:82: If you are loading a serialized model (like pickle
in Python, RDS in R) or
configuration generated by an older version of XGBoost, please export the model by calling
`Booster.save_model` from that version first, then load it back in current version. See:

    https://xgboost.readthedocs.io/en/stable/tutorials/saving_model.html

for more details about differences between saving model and serializing.

  model = pickle.load(open("xgboost_fraud_model.pkl", "rb"))
 * Serving Flask app 'app'
 * 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
```

Activity 2.4: Output screenshots






PAYMENTS FRAUD DETECTION.
Home
About
Contact
Predict
Transaction Fraud Detection
Step
Transaction Type
TRANSFER
Amount
nameOrig
Old Balance Origin
New Balance Origin
nameDest
Old Balance Destination
New Balance Destination
Amount Cleaned
Submit
Prediction:


PAYMENTS FRAUD DETECTION.
Home
About
Contact
Predict
Transaction Fraud Detection
Step
487
Transaction Type
TRANSFER
Amount
321307.31
nameOrig
C2091064188
Old Balance Origin
321307.31
New Balance Origin
0.00
nameDest
C136074678
Old Balance Destination
0.00
New Balance Destination
0.00
Amount Cleaned
321307.31
Submit
Prediction: Fraudulent Transaction