Project Manual

Date	03 November 2022
Team ID	Team -591837
Project Name	Online Payments Fraud Detection Using MI
Maximum Marks	4 Marks

Online Payments Fraud Detection using ML

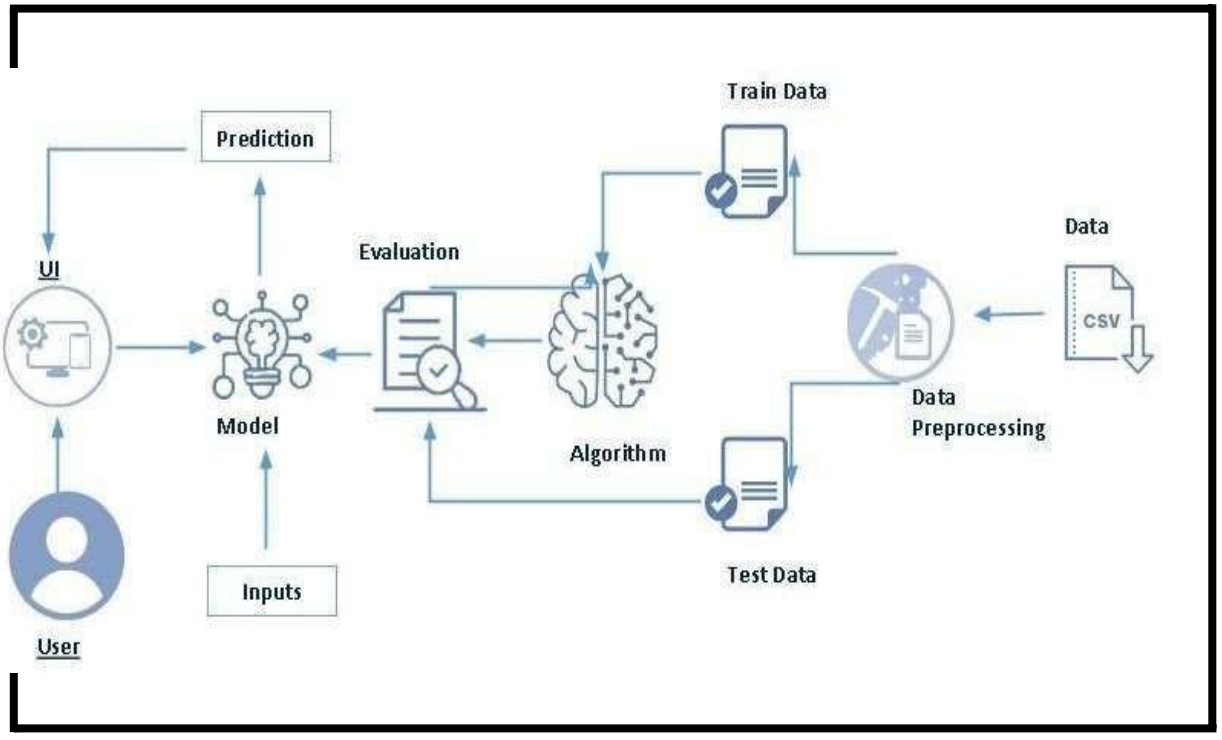
Project Description:

The Online Payments Fraud Detection project aims to enhance the security of online payment transactions by leveraging Machine Learning (ML) techniques to detect and prevent fraudulent activities. As the use of online payment systems continues to grow, so does the risk of fraudulent transactions. This project seeks to build a robust system that can identify and flag potentially fraudulent transactions in real-time, thereby minimizing financial losses and ensuring a secure online payment environment.

By developing a robust Online Payments Fraud Detection system, this project aims to contribute to the security and integrity of online payment platforms, ensuring a safer experience for users and minimizing financial risks associated with fraudulent activities.

We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier, xgboost Classifier**.**We will train and test the data with these algorithms. From this the best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

Technical Architecture:



Pre requisites:

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

- **Anaconda navigator and pycharm:**

Anaconda Navigator is used to manage Python environments and packages. Create a virtual environment named fraud_detecti on for the project. Install essential libraries like scikit-learn, pandas, matplotlib, seaborn, and Jupyter.

PyCharm serves as the Integrated Development Environment (IDE) for project development. Create a new project, configuring it to use the fraud_detecti on environment. Develop Python scripts for data preprocessing, feature engineering, model training, and real-time monitoring.

- **Python packages:**
  - Open anaconda prompt as administrator
  - Type"pip install numpy"and click enter.
  - Type"pip install pandas"andclickenter.
  - Type"pip install scikit-learn"andclickenter.
  - Type"pip install matplotlib"andclickenter.
  - Type"pip install scipy"andclickenter.
  - Type"pip install pickle-mixin"andclickenter.
  - Type"pip install seaborn"andclickenter.
  - Type"pipinstallFlask"and click enter.

Prior Knowledge:

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

ML Concepts

Supervised learning:

In the context of "Online Payments Fraud Detection using ML," supervised learning plays a crucial role. Supervised learning is a type of machine learning where the algorithm learns from labeled training data, and its goal is to make predictions or decisions based on that training data. In the case of fraud detection, supervised learning involves training a model using historical data where transactions are labeled as either legitimate or fraudulent.

Unsupervised learning:

Another important aspect of Online Payments Fraud Detection using ML, especially when dealing with scenarios where labeled fraud data is scarce or when fraud patterns are evolving and not well-defined. Unlike supervised learning, unsupervised learning does not rely on labeled training data but aims to discover patterns and structures within the data itself.

Regression and classification:

Classification is a supervised learning technique used when the goal is to predict the category or class of a given input. In the case of fraud detection, the two main classes are typically "legitimate" and "fraudulent" transactions.

Regression is used when the goal is to predict a continuous numerical output. In the context of fraud detection, regression can be applied to predict a risk score or probability of a transaction being fraudulent.

Decisiontree:

Decision trees are a popular machine learning algorithm used in various applications, including Online Payments Fraud Detection. Decision trees are especially useful for classification tasks, where the goal is to categorize data points into different classes.

X g boost Classifier:

XGBoost (eXtreme Gradient Boosting) is a popular machine learning algorithm that is widely used for various tasks, including classification. It is particularly effective in dealing with structured/tabular data and has been successfully applied to fraud detection problems.

Svm:

SVM is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates data points into different classes.

In the context of fraud detection, SVM can be trained to distinguish between legitimate and fraudulent transactions based on features like transaction amount, timestamp, and user information.

Extra tree classifier:

The Extra Trees Classifier is an ensemble learning method that builds multiple decision trees and merges their predictions.

It is similar to a Random Forest but uses random splitting points for each tree, making it computationally more efficient.

It can handle both classification and regression tasks, making it suitable for fraud detection.

Evaluation metrics:

Evaluation metrics help assess the performance of machine learning models.For classification tasks like fraud detection, common metrics include accuracy, precision, recall, F1-score, and AUC-ROC. The `classification_report` function in scikit-learn provides a comprehensive summary of these metrics.

Flask Basics :

Flask is a lightweight web application framework for Python. It simplifies the process of building web applications by providing tools and libraries for handling routes, templates, and user requests.In the context of fraud detection, Flask can be used to deploy a simple web interface for users to interact with the trained models.

Project Objectives:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some visualisation concepts.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

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

- Data collection
    - Collect the dataset or create the dataset
- Visualising and analysing data

  Importing the libraries
    - Read the Dataset
    - 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:

the Project folder which contains files as shown below

- We are building a flask application which needs HTML pages stored in thetemplates folder and a python script app.py for scripting.
- Model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains model training files and the training_ibm folder contains IBM deployment files.

Milestone 1: Data Collection

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

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

Kaggle Credit Card Fraud Detection Dataset This dataset contains transactions made by credit cards in September 2013 and is often used for fraud detection projects. IEEE-CIS Fraud Detection Dataset: A dataset from a Kaggle competition focused on fraud detection in financial transactions.

Ensure that the dataset is representative of the problem you're trying to solve. It should include a mix of legitimate and fraudulent transactions.

Verify the integrity and reliability of the dataset, and be aware of any biases that may be present.
Respect data privacy and comply with relevant regulations when using real-world datasets.

Milestone 2: Visualizing and analyzing data Activity

1: Importing the libraries

Visualizing and analyzing data is a crucial step in understanding the patterns and characteristics of the dataset, especially in the context of online payments fraud detection using machine learning

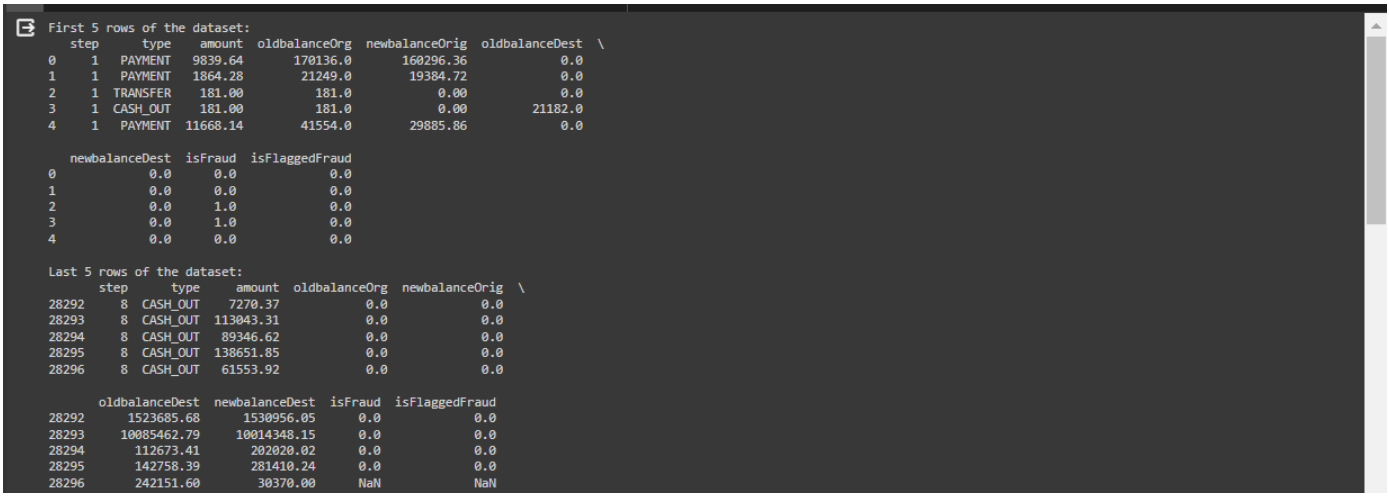
Import the necessary libraries as shown in the image. (optional) Here we have used visualization style as five thirty eight.

```
import pickle
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.ensemble import ExtraTreesClassifier, RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Activity 1: Importing the libraries
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the datasetwith 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.



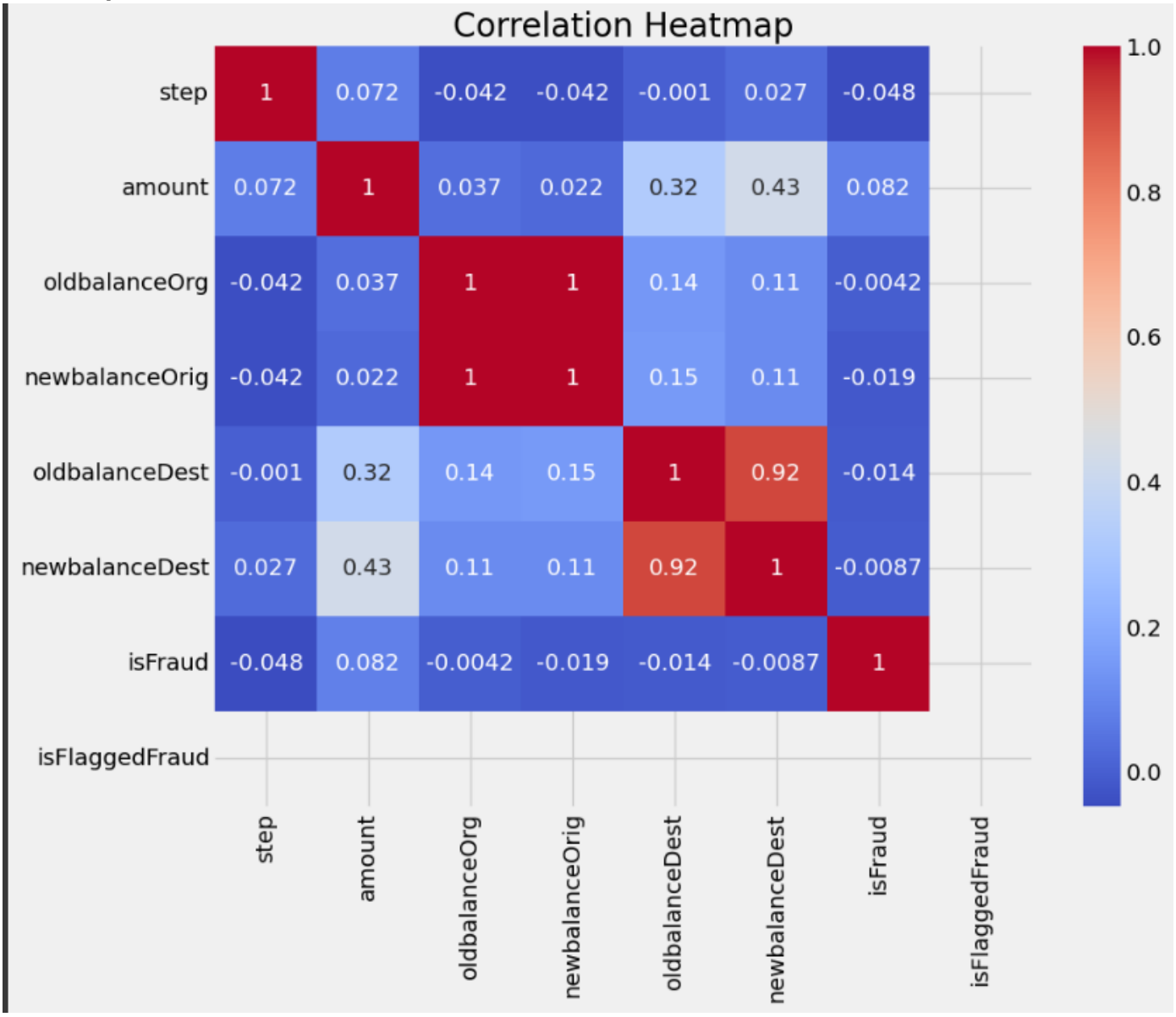
```
First 5 rows of the dataset:
   step      type    amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
0     1   PAYMENT   9839.64       170136.0       160296.36             0.0
1     1   PAYMENT   1864.28        21249.0        19384.72             0.0
2     1  TRANSFER    181.00          181.0            0.00             0.0
3     1  CASH_OUT    181.00          181.0            0.00         21182.0
4     1   PAYMENT  11668.14        41554.0        29885.86             0.0

   newbalanceDest  isFraud  isFlaggedFraud
0             0.0      0.0             0.0
1             0.0      0.0             0.0
2             0.0      1.0             0.0
3             0.0      1.0             0.0
4             0.0      0.0             0.0

Last 5 rows of the dataset:
       step      type     amount  oldbalanceOrg  newbalanceOrig  \
28292     8  CASH_OUT    7270.37            0.0             0.0
28293     8  CASH_OUT  113043.31            0.0             0.0
28294     8  CASH_OUT   89346.62            0.0             0.0
28295     8  CASH_OUT  138651.85            0.0             0.0
28296     8  CASH_OUT   61553.92            0.0             0.0

       oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
28292      1523685.68      1530956.05      0.0             0.0
28293     10085462.79     10014348.15      0.0             0.0
28294       112673.41       202020.02      0.0             0.0
28295       142758.39       281410.24      0.0             0.0
28296       242151.60        30370.00      NaN             NaN
```

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

Heat map:



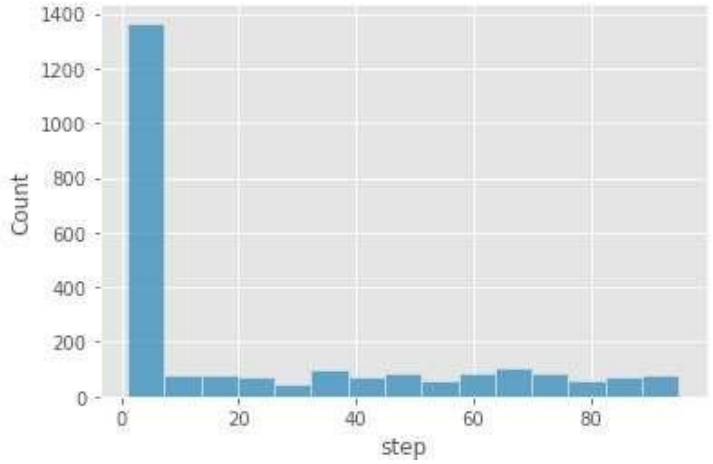
Here, a heatmap is used to understand the relationship between the input attributes andthe anticipated goal value.

Activity 3: Univariate analysis

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

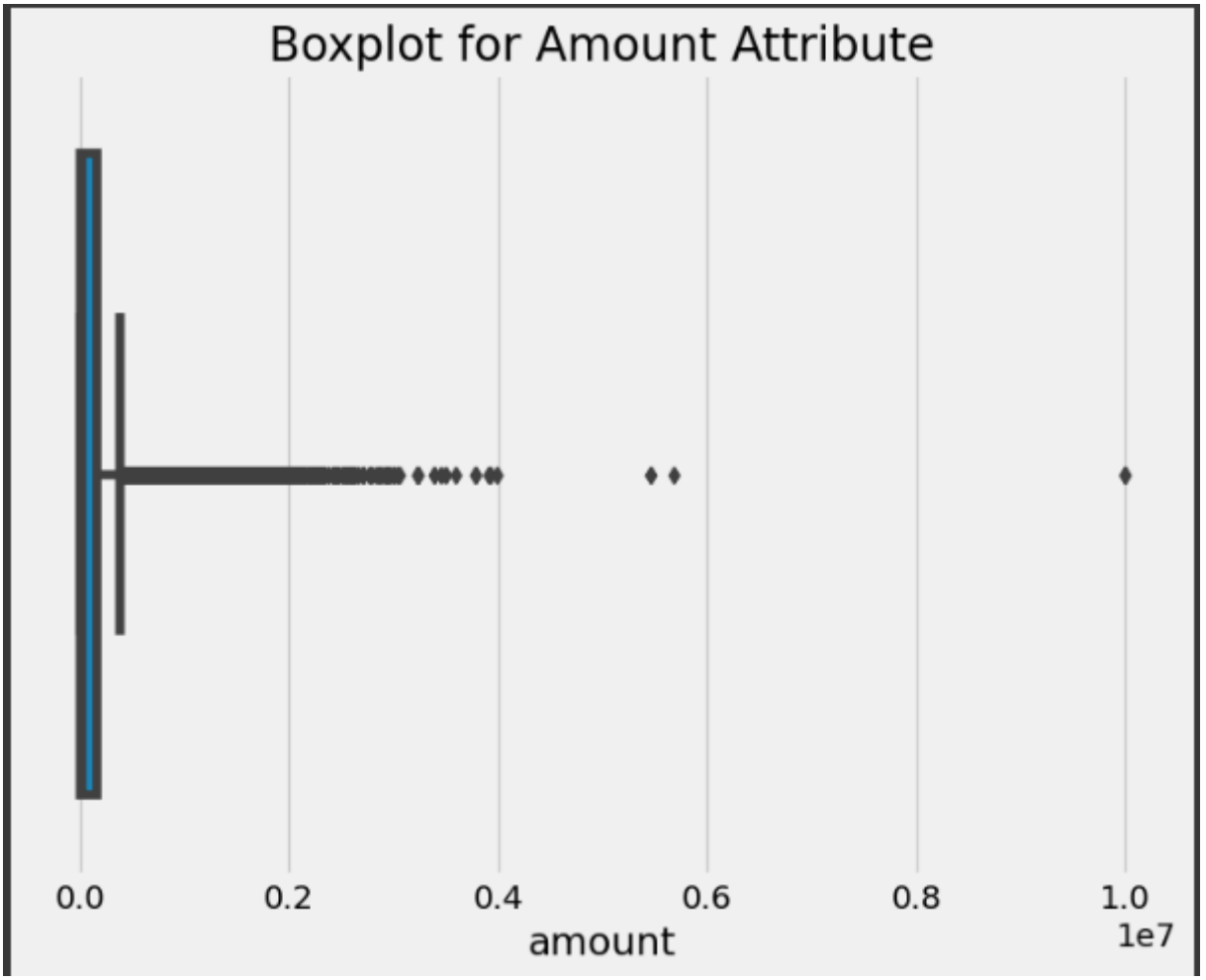
```
#step
sns.histplot(data=df,x='step')
```

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



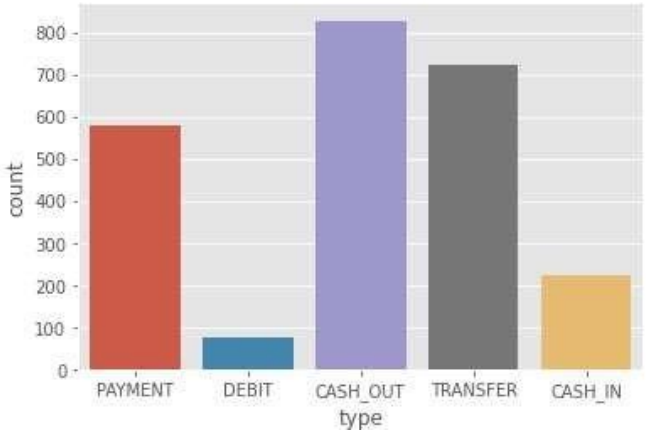
The distribution of one or more variables is represented by a histogram, a traditional visualisation tool, by counting the number of observations that fall within.

Here, the relationship between the step attribute and the boxplot is visualised.


Boxplot for Amount Attribute
0.0
0.2
0.4
0.6
0.8
1.0
amount
1e7

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

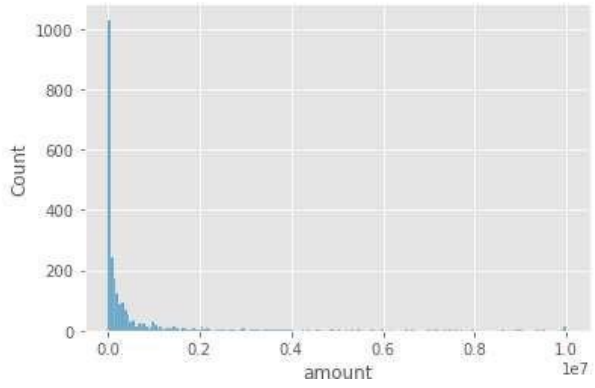
```
<AxesSubplot:xlabel='type', ylabel='count'>
```



Here, the counts of observations in the type attribute of the dataset will be displayed using a countplot.

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

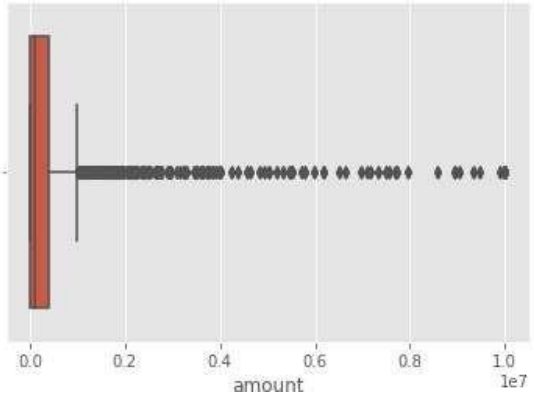
```
<AxesSubplot:xlabel='amount', ylabel='Count'>
```



By creating bins along the data's range and then drawing bars to reflect the number ofobservations that fall within the amount attribute in the dataset.

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

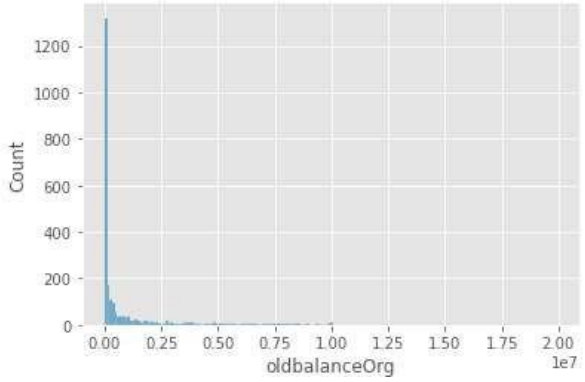
```
<AxesSubplot:xlabel='amount'>
```



Here, the relationship between the amount attribute and the boxplot is visualised.

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

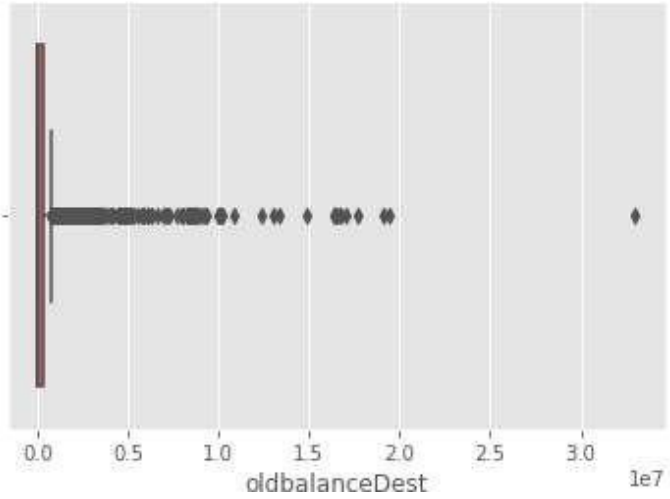
```
<AxesSubplot:xlabel='oldbalanceOrg', ylabel='Count'>
```



utilising the value counts() function here to determine how many times the nameDest columnappears.

```
: #oldbalanceDest
  sns.boxplot(data=df,x='oldbalanceDest')
```

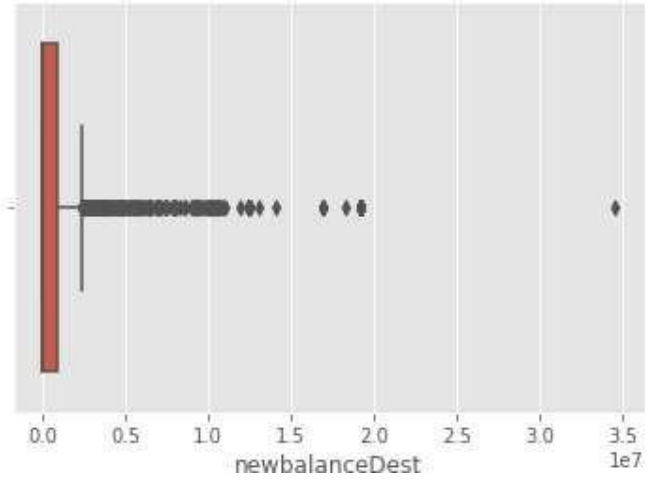
```
: <AxesSubplot:xlabel='oldbalanceDest'>
```



Here, the relationship between the oldbalanceDest attribute and the boxplot is visualised.

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

```
<AxesSubplot:xlabel='newbalanceDest'>
```



Here, the relationship between the newbalanceDest attribute and the boxplot is visualised.

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

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

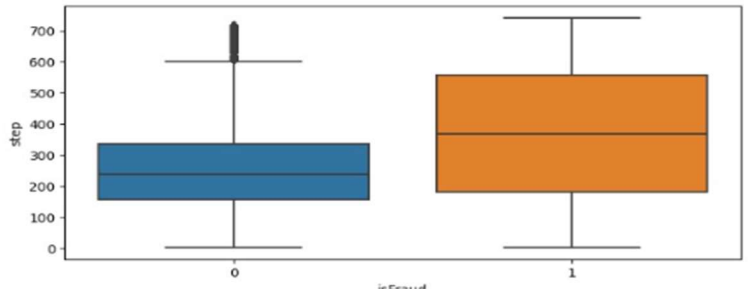
Here, we're using the value counts method to figure out how many classes there are in the dataset's target isFraud column.

```
PS C:\Users\rajab\OneDrive\Desktop\Python> & C:/Users/rajab/AppData/Local/Programs/Python/Python39/python.exe c:/Users/raja
b/OneDrive/Desktop/Python/FakePayment.py
First 5 rows of the dataset:
   step      type    amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0     1   PAYMENT   9839.64       170136.0       160296.36             0.0             0.0        0               0
1     1   PAYMENT   1864.28        21249.0        19384.72             0.0             0.0        0               0
2     1  TRANSFER    181.00          181.0            0.00             0.0             0.0        1               0
3     1  CASH_OUT    181.00          181.0            0.00         21182.0             0.0        1               0
4     1   PAYMENT  11668.14        41554.0        29885.86             0.0             0.0        0               0

Last 5 rows of the dataset:
         step      type      amount  oldbalanceOrg  ...  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
6362615   743  CASH_OUT   339682.13      339682.13  ...            0.00       339682.13        1               0
6362616   743  TRANSFER  6311409.28     6311409.28  ...            0.00            0.00        1               0
6362617   743  CASH_OUT  6311409.28     6311409.28  ...        68488.84      6379898.11        1               0
6362618   743  TRANSFER   850002.52      850002.52  ...            0.00            0.00        1               0
6362619   743  CASH_OUT   850002.52      850002.52  ...      6510099.11      7360101.63        1               0

[5 rows x 9 columns]
c:\Users\rajab\OneDrive\Desktop\Python\FakePayment.py:33: FutureWarning: The default value of numeric_only in DataFrame.cor
r is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_o
nly to silence this warning.
  sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
Null values in the dataset:
step              0
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
type_encoded      0
dtype: int64
Random Forest Classifier Evaluation:
[[1270874      30]
```

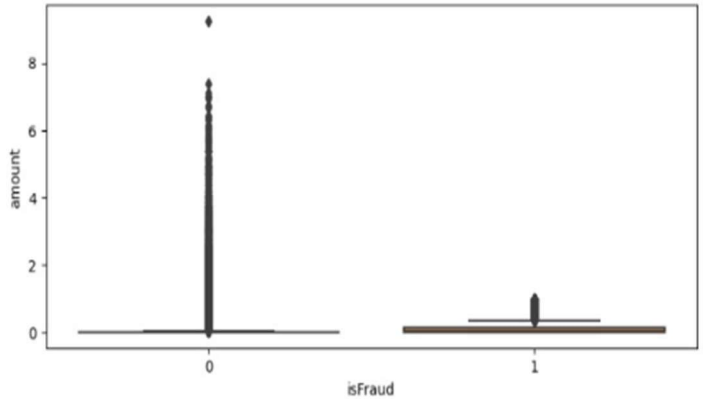
converting 0-means: is not fraud and 1-means: is fraud using the loc technique here

Activity 4: Bivariate analysis

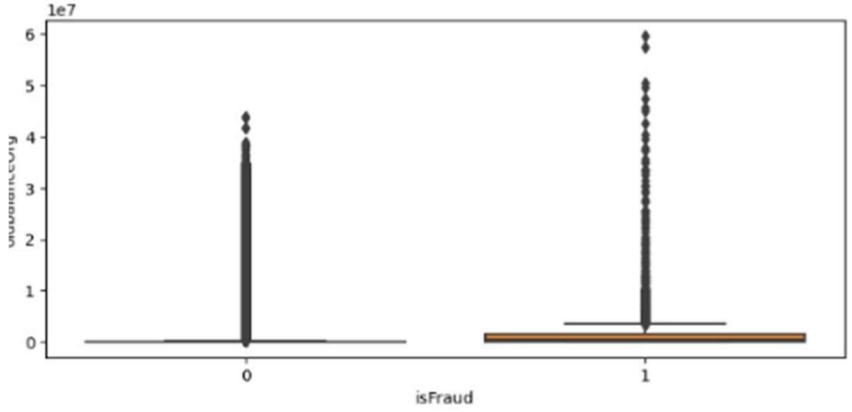
Here we are visualising the relationship between isFraud and step.boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.



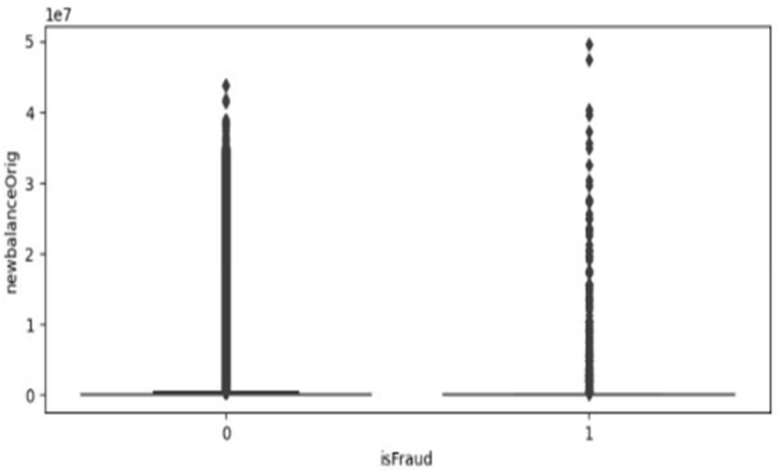
Here we are visualising the relationship between isFraud and amount.boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.



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



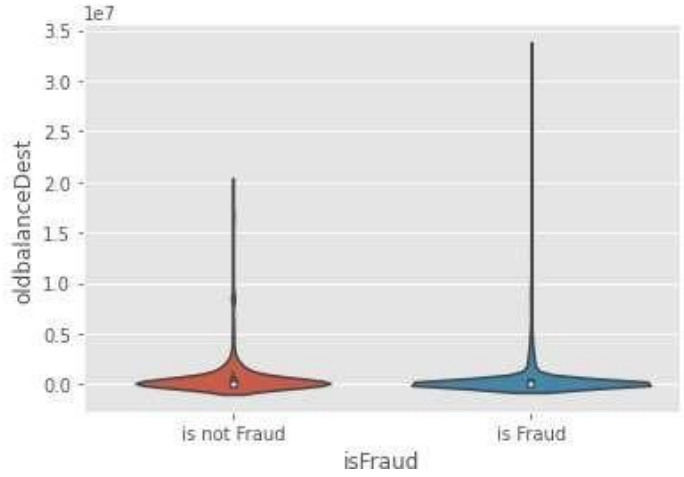
Here we are visualising the relationship between isFraud and newbalanceOrig. boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.


1e7
5
4
3
2
1
0
newbalanceOrig
0
1
isFraud

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

```
sns.violinplot(data=df,x='isFraud',y='oldbalanceDest')
```

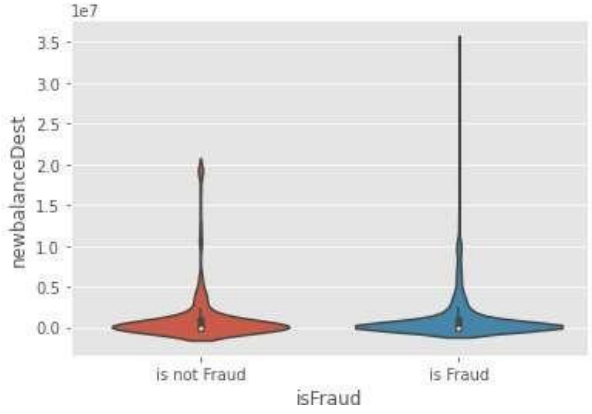
```
<AxesSubplot:xlabel='isFraud', ylabel='oldbalanceDest'>
```



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

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

```
<AxesSubplot:xlabel='isFraud', ylabel='newbalanceDest'>
```



Activity 5: 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.

```
PS C:\Users\rajab\OneDrive\Desktop\Python> & C:/Users/rajab/AppData/Local/Programs/Python/Python39/python.exe c:/Users/rajab/OneDrive/Desktop/Python/fp.py
              step     type      amount     nameOrig  oldbalanceOrg  ...     nameDest  oldbalanceDest  newbalanceDest      isFraud  isFlaggedFraud
count   6.362620e+06  6362620  6.362620e+06      6362620   6.362620e+06  ...      6362620    6.362620e+06    6.362620e+06  6.362620e+06    6.362620e+06
unique           NaN        5           NaN      6353307            NaN  ...      2722362             NaN             NaN           NaN             NaN
top              NaN  CASH_OUT           NaN  C1902386530            NaN  ...  C1286084959             NaN             NaN           NaN             NaN
freq             NaN  2237500           NaN            3            NaN  ...          113             NaN             NaN           NaN             NaN
mean    2.433972e+02       NaN  1.798619e+05          NaN   8.338831e+05  ...          NaN    1.100702e+06    1.224996e+06  1.290820e-03    2.514687e-06
std     1.423320e+02       NaN  6.038582e+05          NaN   2.888243e+06  ...          NaN    3.399180e+06    3.674129e+06  3.590480e-02    1.585775e-03
min     1.000000e+00       NaN  0.000000e+00          NaN   0.000000e+00  ...          NaN    0.000000e+00    0.000000e+00  0.000000e+00    0.000000e+00
25%     1.560000e+02       NaN  1.338957e+04          NaN   0.000000e+00  ...          NaN    0.000000e+00    0.000000e+00  0.000000e+00    0.000000e+00
50%     2.390000e+02       NaN  7.487194e+04          NaN   1.420800e+04  ...          NaN    1.327057e+05    2.146614e+05  0.000000e+00    0.000000e+00
75%     3.350000e+02       NaN  2.087215e+05          NaN   1.073152e+05  ...          NaN    9.430367e+05    1.111909e+06  0.000000e+00    0.000000e+00
max     7.430000e+02       NaN  9.244552e+07          NaN   5.958504e+07  ...          NaN    3.560159e+08    3.561793e+08  1.000000e+00    1.000000e+00

[11 rows x 11 columns]
```

Milestone 3: 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

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.

```
# Shape of csv data
df.shape
```

```
(2430, 10)
```

Activity 1: Checking for null values

Isnull is used (). sum() to check your database for null values. Using the df.info() function, the data type can be determined.

For checking the null values, data.isnull() function is used. To sum those null values we use the .sum() function to it. From the above image we found that there are no null values present in our dataset.So we can skip handling of missing values step.determining the types of each attribute in the dataset using the info() function.

```
# Activity 4: Data Preprocessing

# Activity 4.1: Checking for Null Values
print("Null values in the dataset:")
print(data.isnull().sum())

# Activity 4.2: Handling Outliers (Example using Boxplot for 'amount')
plt.figure(figsize=(8, 6))
sns.boxplot(x='amount', data=data)
plt.title('Boxplot for Amount Attribute')
plt.show()

# Activity 4.3: Handling Missing Values (if present)
# Example: Filling missing values with mean
data.fillna(data.mean(), inplace=True)
```

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: 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

```
Random Forest Classifier Evaluation:
[[5641    0]
 [  10    9]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5641
           2       1.00      0.47      0.64        19

    accuracy                           1.00      5660
   macro avg       1.00      0.74      0.82      5660
weighted avg       1.00      1.00      1.00      5660
```

.

Activity 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 Evaluation:
[[5635    6]
 [   8   11]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5641
           2       0.65      0.58      0.61        19

    accuracy                           1.00      5660
   macro avg       0.82      0.79      0.80      5660
weighted avg       1.00      1.00      1.00      5660
```

Activity 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.

```
Extra Trees Classifier Evaluation:
[[5640    1]
 [  11    8]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5641
           2       0.89      0.42      0.57        19

    accuracy                           1.00      5660
   macro avg       0.94      0.71      0.79      5660
weighted avg       1.00      1.00      1.00      5660
```

Activity 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

```
Evaluating Support Vector Machine:
[[5641    0]
 [  19    0]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5641
           2       0.00      0.00      0.00        19

    accuracy                           1.00      5660
   macro avg       0.50      0.50      0.50      5660
weighted avg       0.99      1.00      0.99      5660
```

Activity 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.

```
XGBoost Classifier Evaluation:
[[5640    1]
 [   5   14]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5641
           2       0.93      0.74      0.82        19

    accuracy                           1.00      5660
   macro avg       0.97      0.87      0.91      5660
weighted avg       1.00      1.00      1.00      5660
```

Activity 6: Compare the model

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 svc is performing well. From the below image, We can see the accuracy of the model is 79% accuracy. .

```
# Activity 6: Compare the Models
def compareModels(models, X_train, X_test, y_train, y_test):
    for name, model in models.items():
        print(f"Evaluating {name}:")
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)

        # Evaluation
        print(confusion_matrix(y_test, predictions))
        print(classification_report(y_test, predictions))
```

Activity 7: Evaluating performance of the model and saving the model

From sklearn, accuracy_score is used to evaluate the score of the model. On the parameters, we have given svc (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model is svc by pickle.dump().

```
Evaluating performance of the model and saving the model
def saveModel(model, X, y, filename):
    model.fit(X, y)
    pickle.dump(model, open(filename, 'wb'))
# Assuming X_train, X_test, y_train, y_test are available after preprocessing
rf_model = RandomForest(X_train, X_test, y_train, y_test)
dt_model = DecisionTree(X_train, X_test, y_train, y_test)
et_model = ExtraTree(X_train, X_test, y_train, y_test)
svc_model = SupportVector(X_train, X_test, y_train, y_test)
xg_model = xgboost(X_train, X_test, y_train, y_test)
models = {
    'Random Forest': RandomForestClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Extra Trees': ExtraTreesClassifier(),
    'Support Vector Machine': SVC(),
    'XGBoost': XGBClassifier()
}

compareModels(models, X_train, X_test, y_train, y_test)

# Save the best performing model
best_model = SVC()  # Assuming SVC performed the best
saveModel(best_model, X, y, "C:\\Users\\rajab\\Downloads\\dataset\\best_model2.pkl")
```

Milestone 5: Application Building

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

This section has the following tasks

Building HTML Pages Building

server side script

Activity1: Building Html Pages:

For this project create three HTML files namely

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

and save them in the templates folder.

Html home:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Fraud Detection System</title>
    <link rel="stylesheet" href="../static/styles.css">
</head>
<body class="background-image">
    <div class="container">
        <h1>Welcome to the Fraud Detection System</h1>
        <p>Explore the prediction page to make a prediction.</p>
        <a href="Predict.html" class="button">Go to Prediction Page</a>
    </div>
</body>
</html>
```

Html predict:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction Result</title>
    <link rel="stylesheet" href="../static/styles.css">
</head>
<body>
    <div class="container">
        <h1>Prediction Result</h1>
        <h3>The predicted result is: 1</h3>
        <p>0 means "NOT FRAUD" and 1 means "FRAUD"</p>
        <a href="Home.html" class="button">Go back to Home</a>
    </div>
</body>
</html>
```

Html submit:

ne wrap

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Fraud Prediction</title>
    <link rel="stylesheet" href="../static/styles.css">
</head>
<body>
    <div class="container">
        <h1>Fraud Prediction</h1>
        <form method="POST" action="{{ url_for('predict_up') }}">
            <label for="step">Step:</label>
            <input type="text" id="step" name="step"><br><br>

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

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

            <label for="oldbalanceOrg">Old Balance Origin:</label>
            <input type="text" id="oldbalanceOrg" name="oldbalanceOrg"><br><br>

            <label for="newbalanceOrig">New Balance Origin:</label>
            <input type="text" id="newbalanceOrig" name="newbalanceOrig"><br><br>

            <label for="oldbalanceDest">Old Balance Destination:</label>
            <input type="text" id="oldbalanceDest" name="oldbalanceDest"><br><br>

            <label for="newbalanceDest">New Balance Destination:</label>
            <input type="text" id="newbalanceDest" name="newbalanceDest"><br><br>

            <label for="balance_difference">Balance Difference:</label>
            <input type="text" id="balance_difference" name="balance_difference"><br><br>

            <input type="submit" value="Submit">
        </form>

        <a href="Home.html" class="button">Go back to Home</a>
    </div>
</body>
</html>
```

Html styles:

```css
/* Example styles */
body {
    font-family: 'Poppins', sans-serif;
    font-family: 'Vina Sans', sans-serif;
    background-color: #f0f0f0;
    color: #333;
    margin: 0;
    padding: 0;
    text-align: center;
    background-image: url('/static/bg.webp');
    background-size: cover;
    background-position: center;
    background-attachment: fixed;

}

h1 {
    color: #F24E08;
    font-size: 35px;
    font-weight: 1500;
    margin-top: 15%;
}
h3 {
    color:#DF971B ;
    font-size: 25px;
    margin-top: auto;
}
p{
    color: #C4C4D7;
    font-size: 16px;
    font-weight: bolder;
}
button{
    border: none;
    outline: none;
    cursor: pointer;
    width: 12%;
    height: 40px;
    background-color: #ff4141;
    color: white;
    display: inline-block;
```

```
    border-radius: 15px;
}
input{
    border: none;
    outline: none;
    box-sizing: border-box;
    width: 20%;
    height: 40px;
    padding: 8px 15px;
    border-bottom: 2px solid #ccc;
    font-size: 16px;
    cursor: pointer;

}
label{
    font-size: 16px;
    align-items:flex-start;
    font-weight: 100;
    line-height: 1.7em;
    color:#ff4141;
}
bg{
    background-attachment:fixed;
    background-position: center;
    background-size: cover;
}
background-image{

}
```

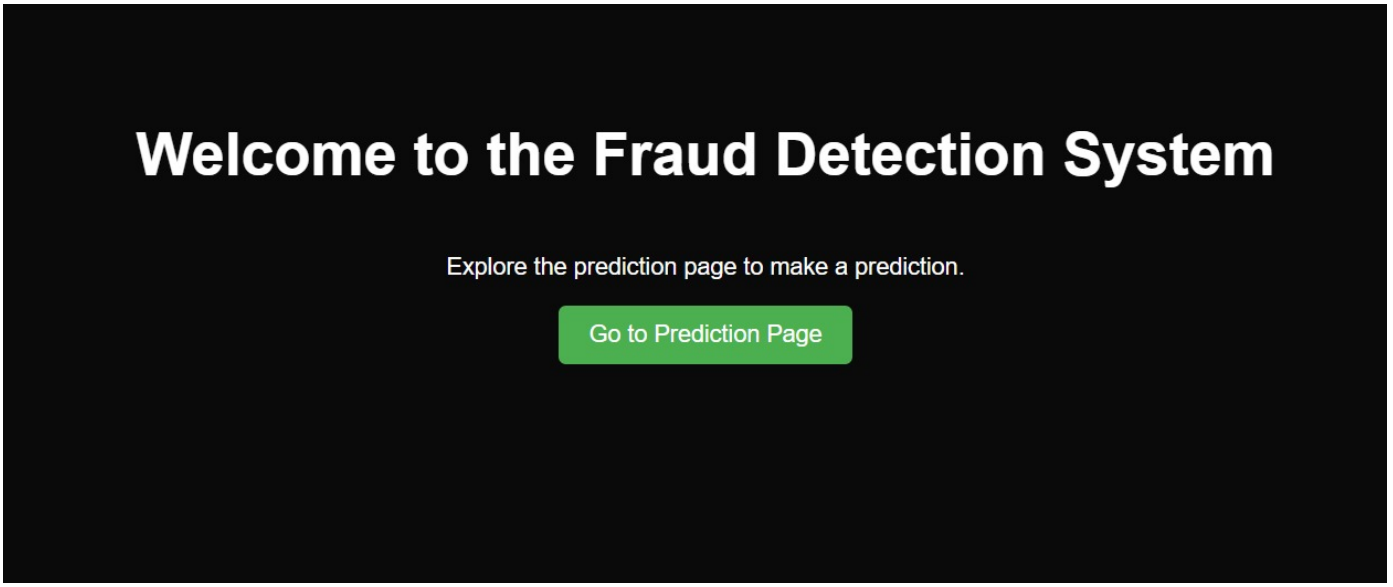
Html app:

C: > Users > manoj > Downloads > app.py > ...

```python
from flask import Flask, render_template, request
import pickle
import numpy as np
import warnings
from sklearn.exceptions import InconsistentVersionWarning

# Suppress scikit-learn version warning during development
warnings.filterwarnings("ignore", category=InconsistentVersionWarning)

# Load the trained model
model = pickle.load(open("best_model1.pkl", 'rb'))

app = Flask(__name__)

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

@app.route('/predict', methods=['GET', 'POST'])
def home1():
    if request.method == 'GET':
        return render_template('predict.html')

@app.route('/submit', methods=['POST', 'GET'])
def predict_up():
    if request.method == 'POST':
        form_values = [request.form[field] for field in request.form]
        print("Form values:", form_values)

        # Check if any form field is empty
        if any(value == '' for value in form_values):
            return render_template('prediction.html', prediction_text="Ple

        try:
            # Convert form values to float (excluding empty strings)
            form_values_float = [float(value) for value in form_values if
            print("Form values (float):", form_values_float)
```

```
        # Check if any form field is empty
        if any(value == '' for value in form_values):
            return render_template('prediction.html', prediction_text="Ple

        try:
            # Convert form values to float (excluding empty strings)
            form_values_float = [float(value) for value in form_values if
            print("Form values (float):", form_values_float)

            if len(form_values_float) > 0:
                x = np.array(form_values_float).reshape(1, -1)
                print("Input shape:", x.shape)

                pred = model.predict(x)
                print("Prediction:", pred[0])

                return render_template('prediction.html', prediction_text=
            else:
                return render_template('prediction.html', prediction_text=
        except ValueError:
            return render_template('prediction.html', prediction_text="Inv
    else:
        return render_template('prediction.html', prediction_text="No pred

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

Final output



Fraud Prediction

Step:

Type:

Amount:

Old Balance Origin:

New Balance Origin:

Old Balance Destination:

New Balance Destination:

Balance Difference:

Submit

Go back to Home

Fraud Prediction

Step: 94

Type: 4

Amount: 14.590090

Old Balance Origin: 2169679.910.0

New Balance Origin: 0.0

Old Balance Destination: 0.0

New Balance Destination: 0.0

Balance Difference:

Submit

Go back to Home

Prediction Result

The predicted result is: 1

0 means "NOT FRAUD" and 1 means "FRAUD"

Go back to Home

Prediction Result

The predicted result is: 0

0 means "NOT FRAUD" and 1 means "FRAUD"

Go back to Home