ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

PROJECT DOCUMENTATION

TITLE : Online Payments Fraud Detection Using Machine Learning

TEAM ID : LTVIP2026TMIDS45374

Team Leader : Gayathri Katakamsetti

Team member : Dhavaleswarapu Keerthi

Team member : Garlapati Anand Babu

Team member : Alla Adi Keshava Reddy

Team member : Inturi Haranadha Reddy

Project Description:

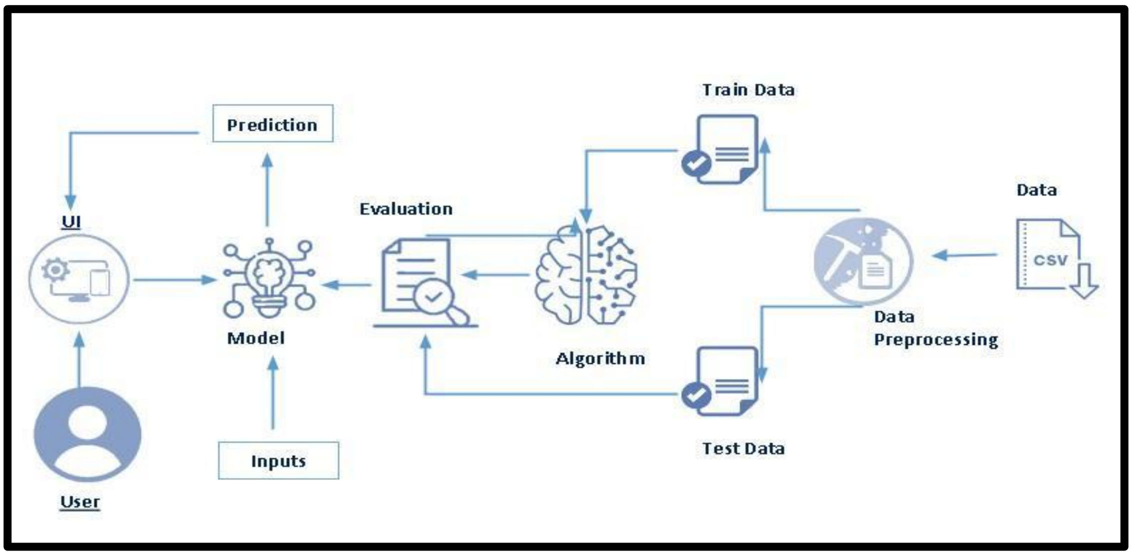
Online payment systems have become a major part of daily life, but they also face a serious problem—**fraudulent transactions**. Fraud detection is important because fraud causes financial loss to both customers and banks, and it reduces trust in digital payment systems.

This project focuses on building a **Machine Learning-based fraud detection system** that can automatically identify whether a transaction is **fraudulent or legitimate**. The model is trained using historical transaction data that contains features such as transaction type, amount, balance details, and other payment-related attributes.

In this project, the dataset is pre processed by handling missing values, encoding categorical variables, and splitting the data into training and testing sets. Then, Machine Learning algorithms such as **Logistic Regression** and **Random Forest Classifier** are used to train the model. After training, the model performance is evaluated using metrics like **accuracy score, confusion matrix, precision, recall, and F1-score**.

The final system helps in detecting fraud transactions quickly and efficiently, reducing financial loss and improving the security of online payment platforms..

Technical Architecture:



Pre requisites:

To complete this project, you must required 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
  - Regression and classification
  - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
  - Random forest: https://www.javatpoint.com/machine-learning-random-forest-

        algorithm
    - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **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 analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

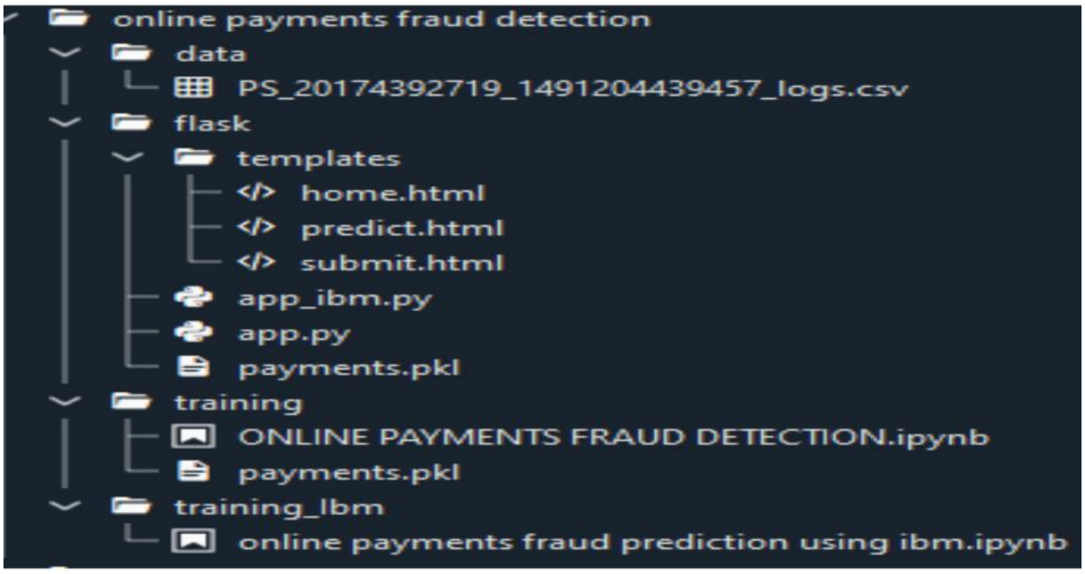
To accomplish this, we have to complete all the activities listed below,

- Data collection
    - Collect the dataset or create the dataset
- Visualizing and analyzing data
    - Univariate analysis
    - Bivariate analysis
    - Multivariate analysis
    - Descriptive analysis
- Data pre-processing
    - Checking for null values
    - Handling outlier
    - Handling categorical data
    - 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
    - Save the model
- Application Building

- Create an HTML file
- Build python code

Project Structure:

Create the Project folder which contains files as shown below



```
online payments fraud detection
├── data
│   └── PS_20174392719_1491204439457_logs.csv
├── flask
│   ├── templates
│   │   ├── home.html
│   │   ├── predict.html
│   │   └── submit.html
│   ├── app_ibm.py
│   ├── app.py
│   └── payments.pkl
├── training
│   ├── ONLINE PAYMENTS FRAUD DETECTION.ipynb
│   └── payments.pkl
└── training_ibm
    └── online payments fraud prediction using ibm.ipynb
```

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

Milestone 1: Data Collection

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. Eg: kaggle.com, UCI repository, etc.

In this project we have used final data.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

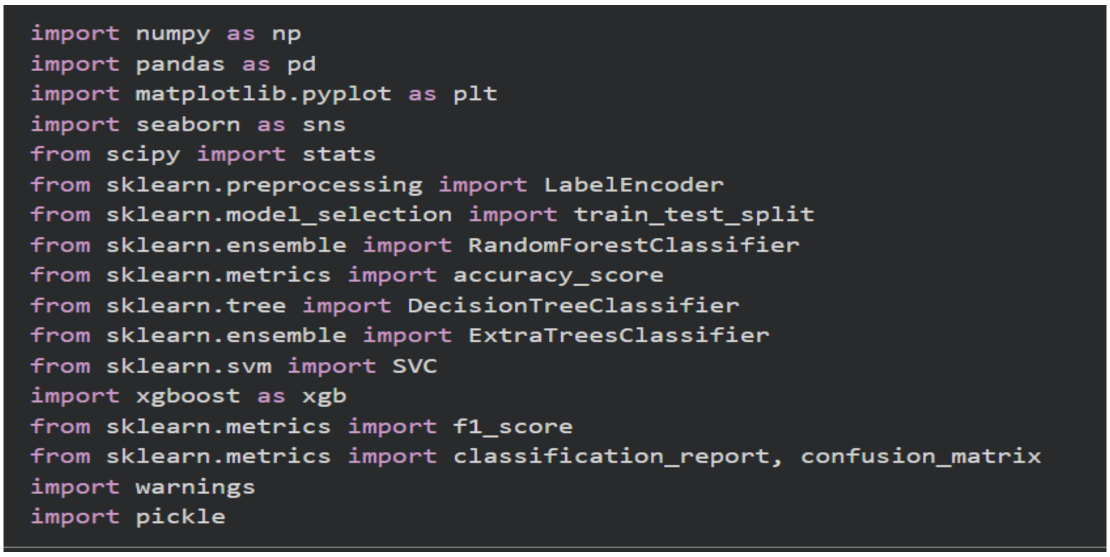
Milestone 2: Visualizing and analysing the data

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 1: Importing the libraries

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

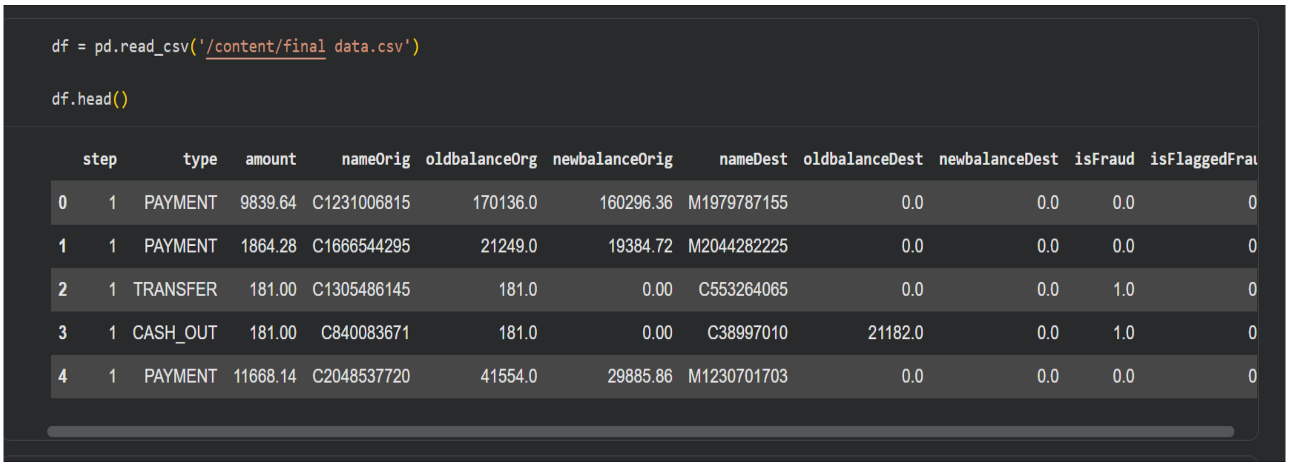


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import 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
```

Activity 2: 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.



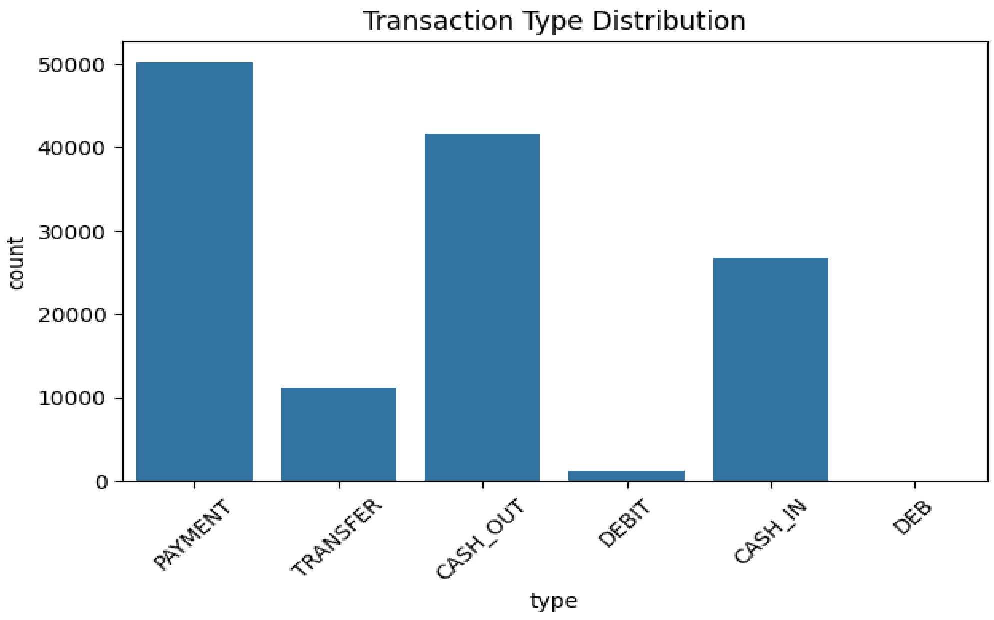
```
df = pd.read_csv('/content/final data.csv')

df.head()
```

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

Activity 3: Univariate analysis

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

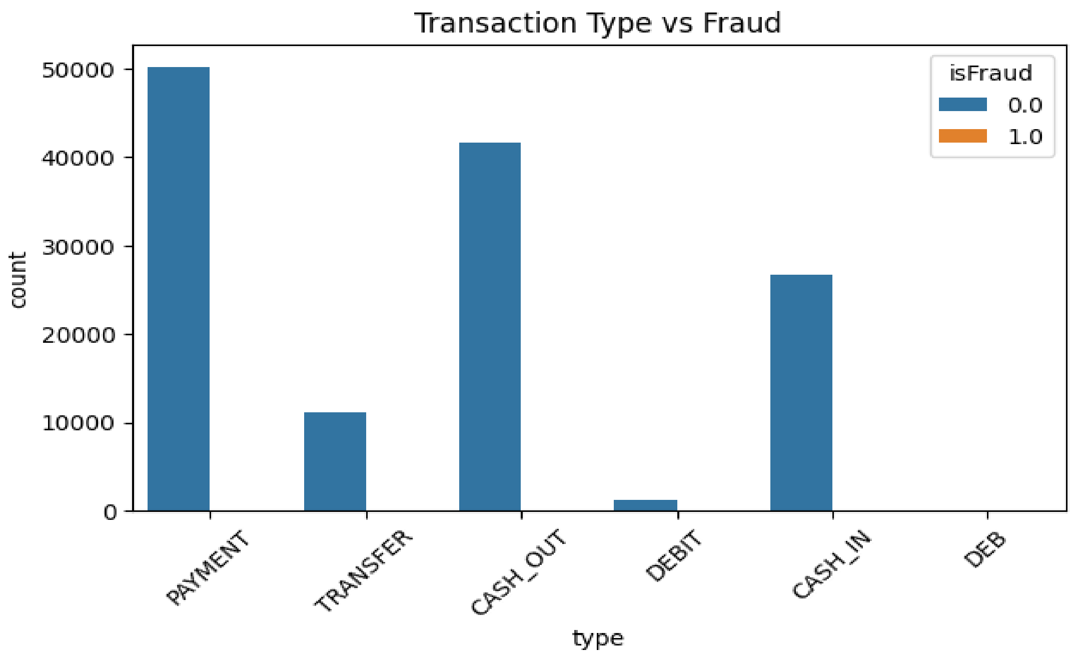
- Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



- In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
- From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.

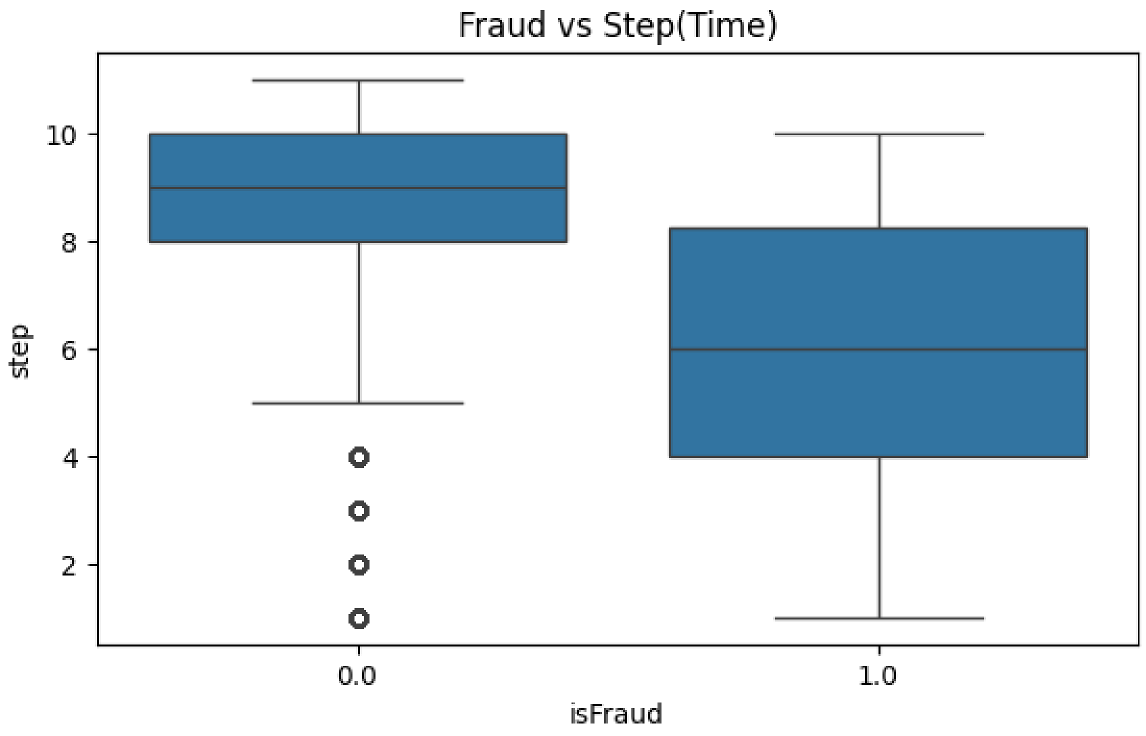
Activity 4: Bivariate analysis

- Bivariate analysis examines the relationship between two variables. In this study, the relationship between transaction type and fraud status (isFraud) was analyzed to determine whether certain transaction types are more prone to fraudulent activity.
- The bar chart comparing transaction types with fraud status shows that fraudulent transactions are not evenly distributed aeross all categories.
- Fraud appears to occur primarily in TRANSFER and CASH_OUT transactions, while very few or no fraudulent cases are observed in PAYMENT, CASH_IN, and DEBIT transactions.



Countplot:-

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

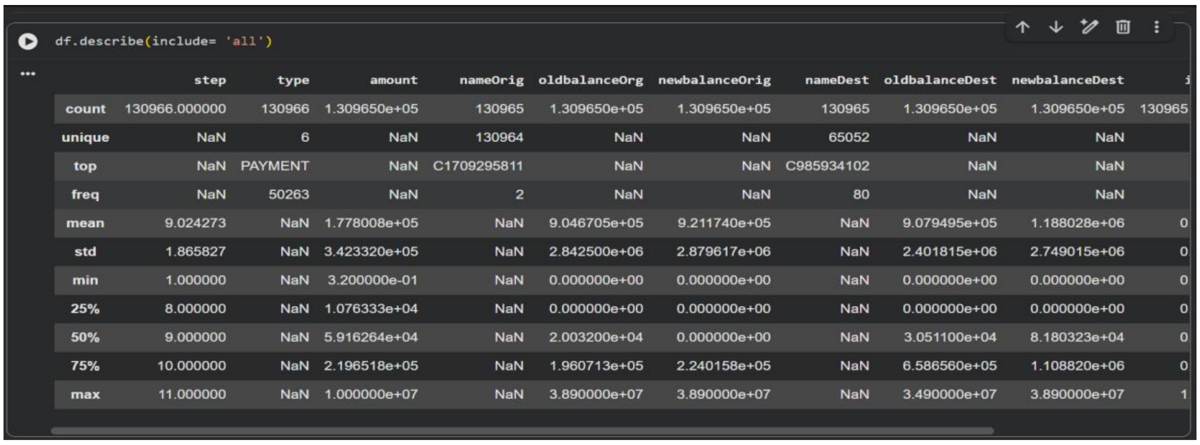


From the above graph we can infer the analysis such as

- Segmenting the gender column and married column based on bar graphs
- Segmenting the Education and Self-employed based on bar graphs ,for drawing insights such as educated people are employed.
- Loan amount term based on the property area of a person holding

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.



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

```
                 step     type        amount     nameOrig  oldbalanceOrg  newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest       i
count   130966.000000   130966  1.309650e+05       130965   1.309650e+05    1.309650e+05       130965    1.309650e+05    1.309650e+05  130965
unique            NaN        6           NaN       130964            NaN             NaN        65052             NaN             NaN
top               NaN  PAYMENT           NaN  C1709295811            NaN             NaN   C985934102             NaN             NaN
freq              NaN    50263           NaN            2            NaN             NaN           80             NaN             NaN
mean         9.024273      NaN  1.778008e+05          NaN   9.046705e+05    9.211740e+05          NaN    9.079495e+05    1.188028e+06       0
std          1.865827      NaN  3.423320e+05          NaN   2.842500e+06    2.879617e+06          NaN    2.401815e+06    2.749015e+06       0
min          1.000000      NaN  3.200000e-01          NaN   0.000000e+00    0.000000e+00          NaN    0.000000e+00    0.000000e+00       0
25%          8.000000      NaN  1.076333e+04          NaN   0.000000e+00    0.000000e+00          NaN    0.000000e+00    0.000000e+00       0
50%          9.000000      NaN  5.916264e+04          NaN   2.003200e+04    0.000000e+00          NaN    3.051100e+04    8.180323e+04       0
75%         10.000000      NaN  2.196518e+05          NaN   1.960713e+05    2.240158e+05          NaN    6.586560e+05    1.108820e+06       0
max         11.000000      NaN  1.000000e+07          NaN   3.890000e+07    3.890000e+07          NaN    3.490000e+07    3.890000e+07       1
```

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.

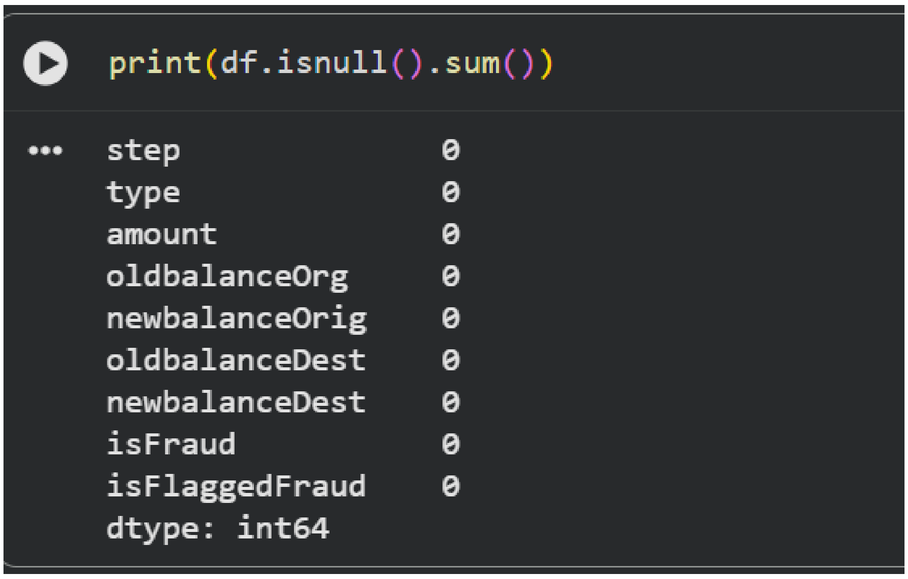
- Checking for null values
- Handling outliers
- Object Data label encoding
- Splitting dataset into training and test set

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.

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.

- 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.

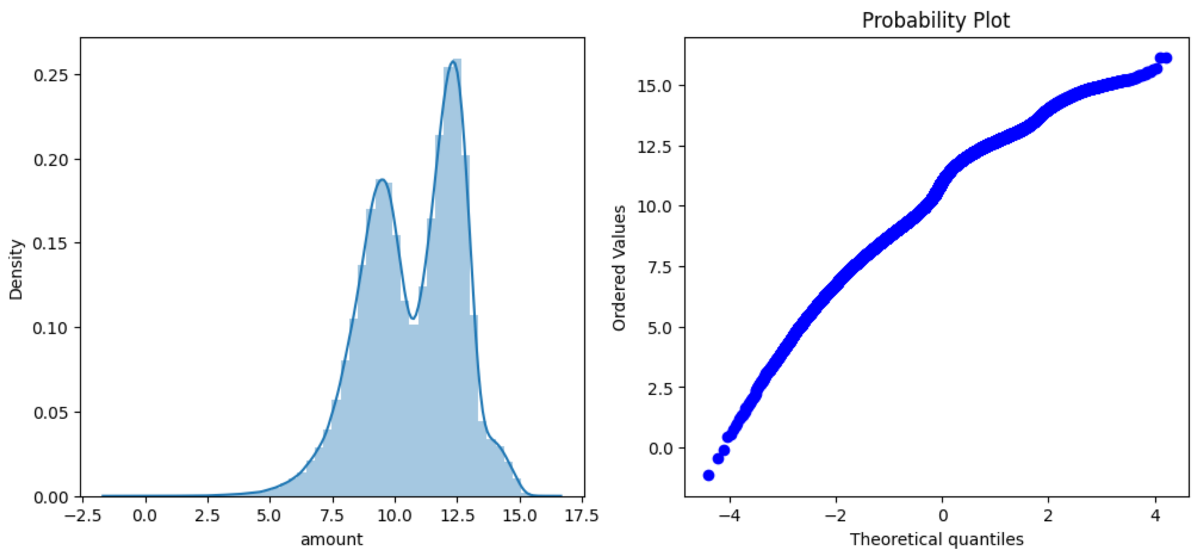


```
print(df.isnull().sum())
```

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

Activity 2: Handling Outliers

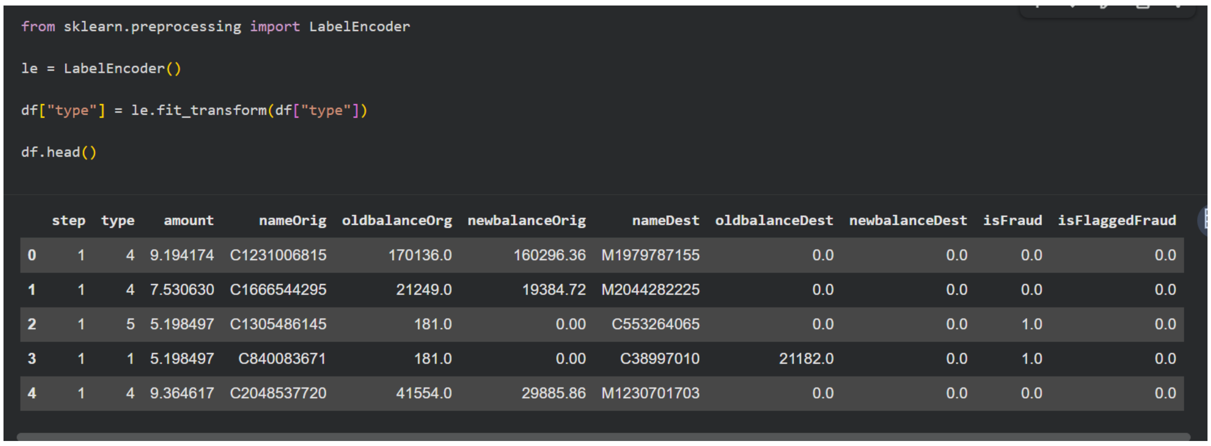
- In the context of online payment fraud detection, high transaction amounts may represent genuine fraud cases rather than errors or noise. Therefore, removing these outliers could lead to the loss of critical information and negatively impact the predictive performance of the model.



Activity 3: Object Data Label Encoding

In fraud detection datasets, many features are categorical (text-based), such as transaction type, device type, or payment method. Machine learning algorithms require numerical inputs, so these text labels need to be converted to numbers.

Label Encoding is a technique that converts each unique category in a variable into an integer value. For example, for a Transaction Type feature:

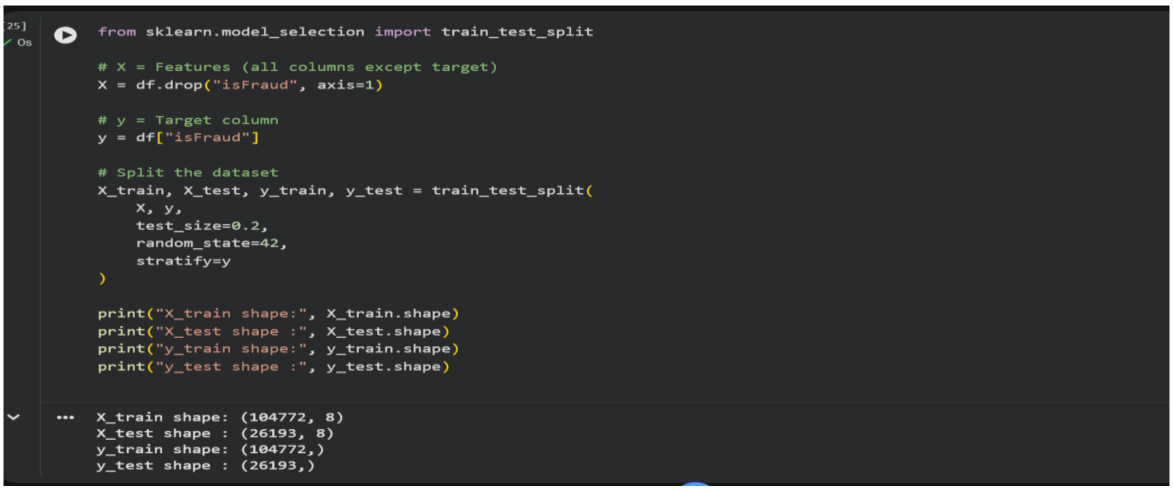


```python
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df["type"] = le.fit_transform(df["type"])

df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	4	9.194174	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0	0.0
1	1	4	7.530630	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0	0.0
2	1	5	5.198497	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0	0.0
3	1	1	5.198497	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0	0.0
4	1	4	9.364617	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0	0.0

Activity 4: Splitting data into train and test

To build and evaluate a reliable fraud detection model, the dataset was split into training and testing sets. The training set is used to train the machine learning model, while the testing set is used to assess the model's performance on unseen data, ensuring it can generalize well and avoid overfitting.



```python
from sklearn.model_selection import train_test_split

# X = Features (all columns except target)
X = df.drop("isFraud", axis=1)

# y = Target column
y = df["isFraud"]

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("X_train shape:", X_train.shape)
print("X_test shape :", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape :", y_test.shape)
```

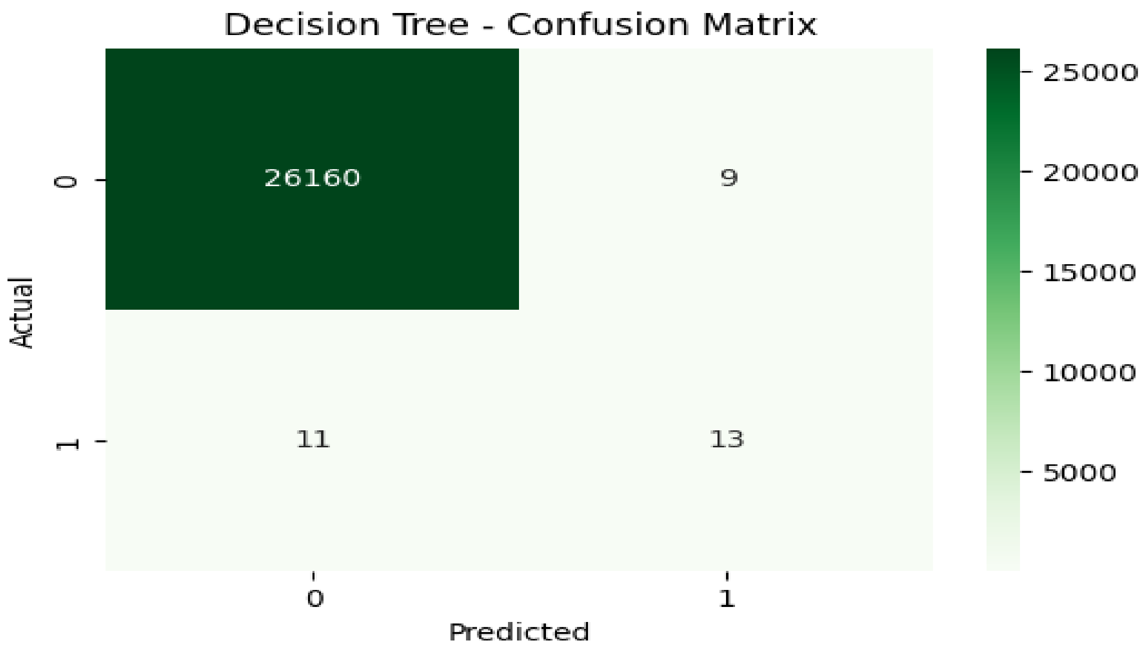
```
X_train shape: (104772, 8)
X_test shape : (26193, 8)
y_train shape: (104772,)
y_test shape : (26193,)
```

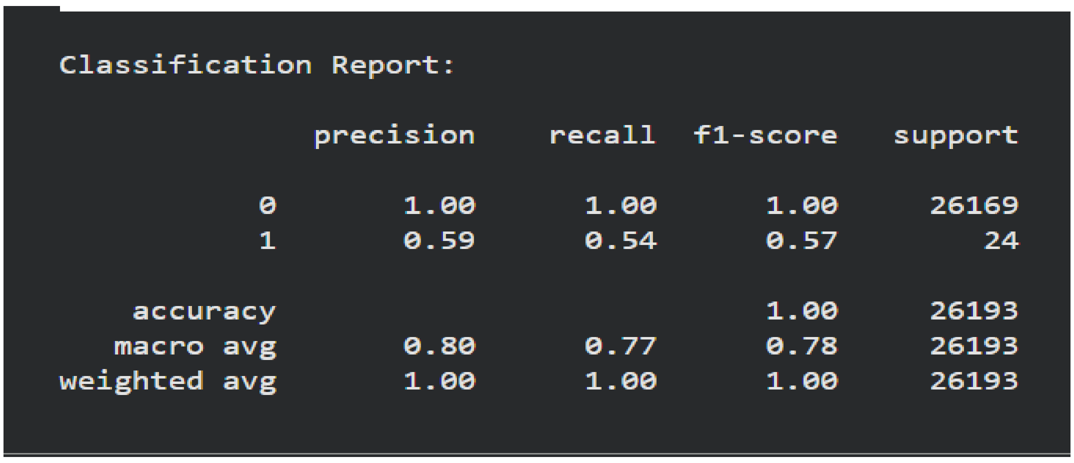
Milestone 4: Model Building

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 model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier 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 done.

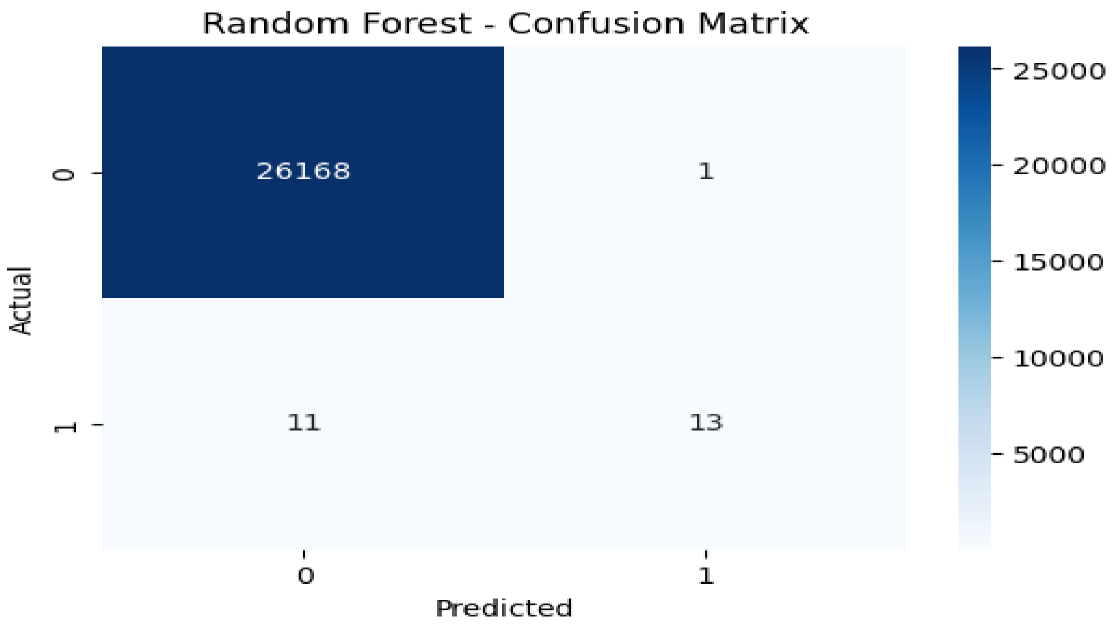


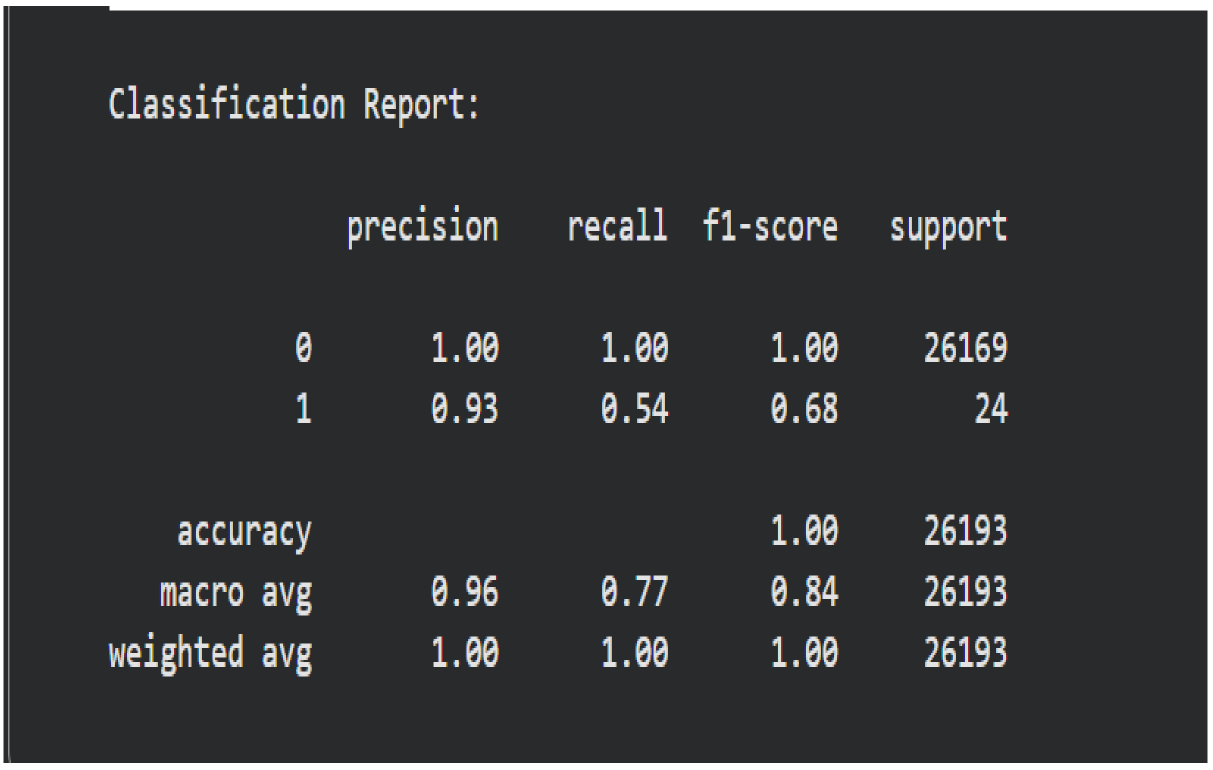


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     26169
           1       0.59      0.54      0.57        24

    accuracy                           1.00     26193
   macro avg       0.80      0.77      0.78     26193
weighted avg       1.00      1.00      1.00     26193
```

Activity 2: Random forest model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier 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 done.



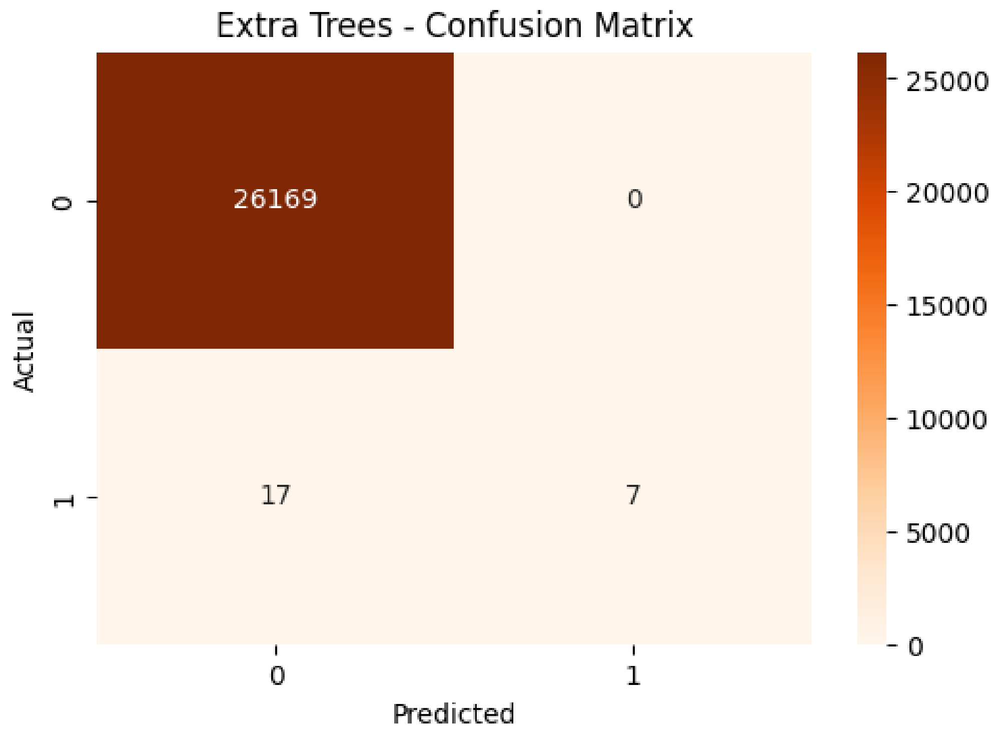


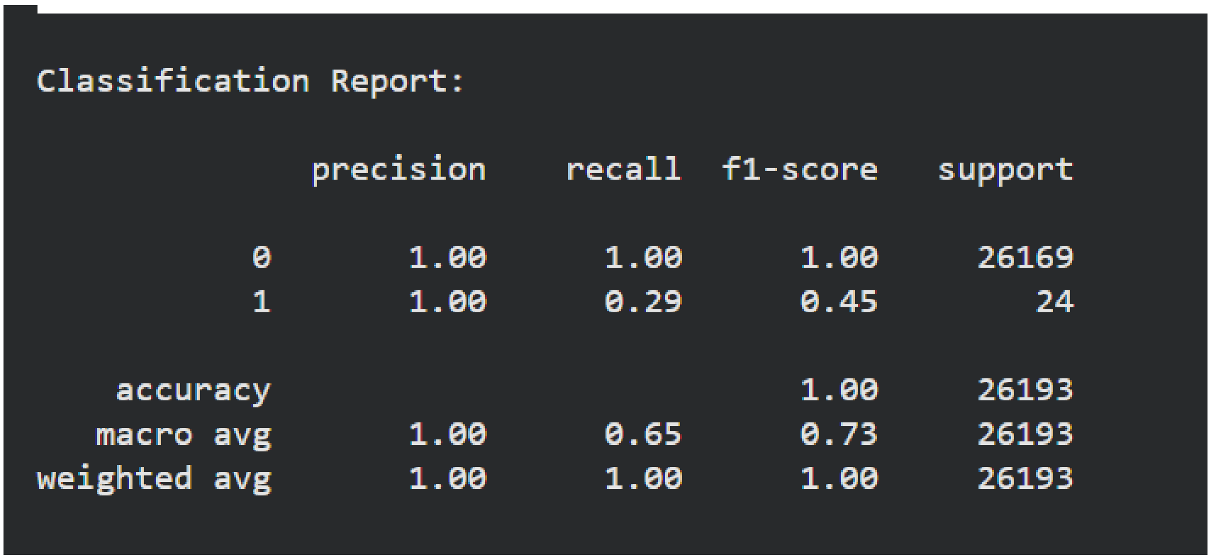
```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     26169
           1       0.93      0.54      0.68        24

    accuracy                           1.00     26193
   macro avg       0.96      0.77      0.84     26193
weighted avg       1.00      1.00      1.00     26193
```

Activity 3: Extra Trees Classifier

The **Extra Trees Classifier** is an ensemble machine learning model that builds many decision trees for classification. Unlike Random Forests, it chooses **split points randomly** and usually uses the **whole dataset** for each tree, making it **faster** and reducing overfitting.

Predictions are made by **majority vote** across all trees, making it robust and effective for high-dimensional or noisy data.

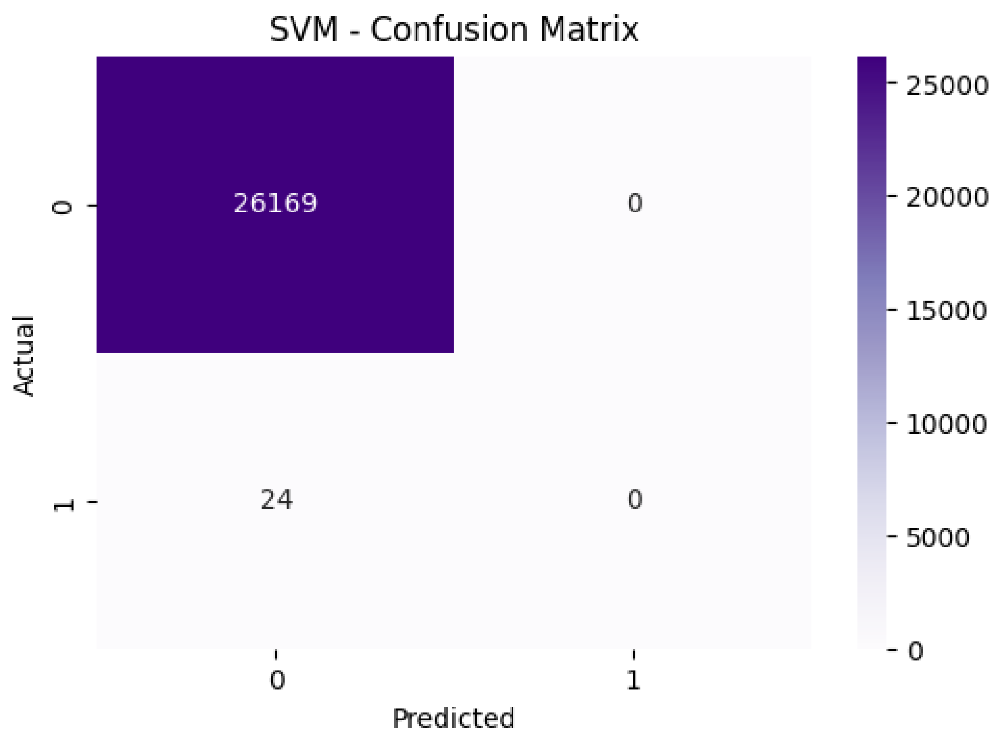


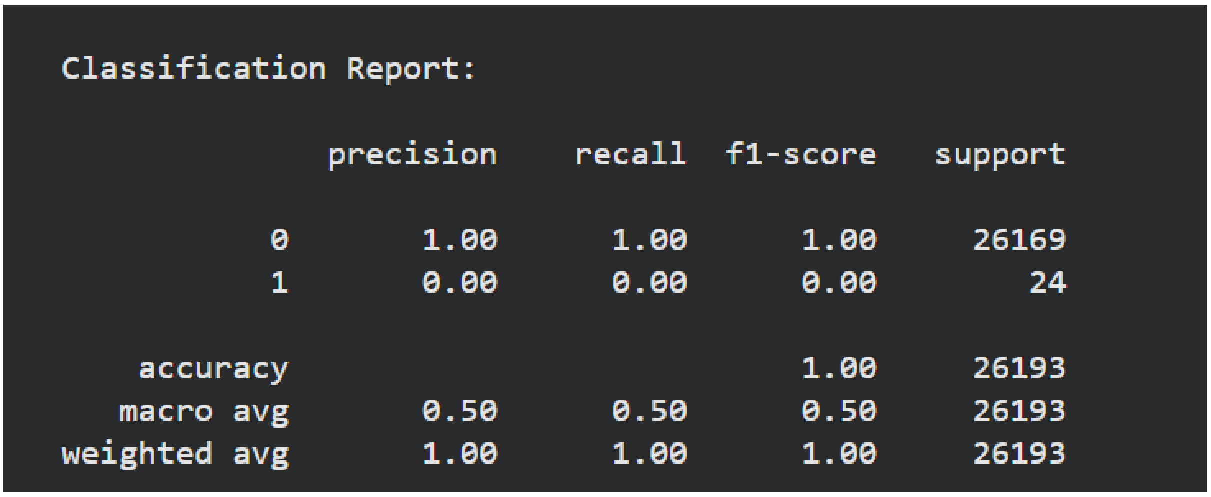


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     26169
           1       1.00      0.29      0.45        24

    accuracy                           1.00     26193
   macro avg       1.00      0.65      0.73     26193
weighted avg       1.00      1.00      1.00     26193
```

Activity 5:Support Vector Machine Classifier

The SVM classifier is a supervised learning algorithm that separates data into classes by finding the hyperplane with the maximum margin between them. It can handle linear and non-linear data using kernel functions and focuses on the support vectors closest to the boundary for making predictions. And the confusion matrix and classification report are:

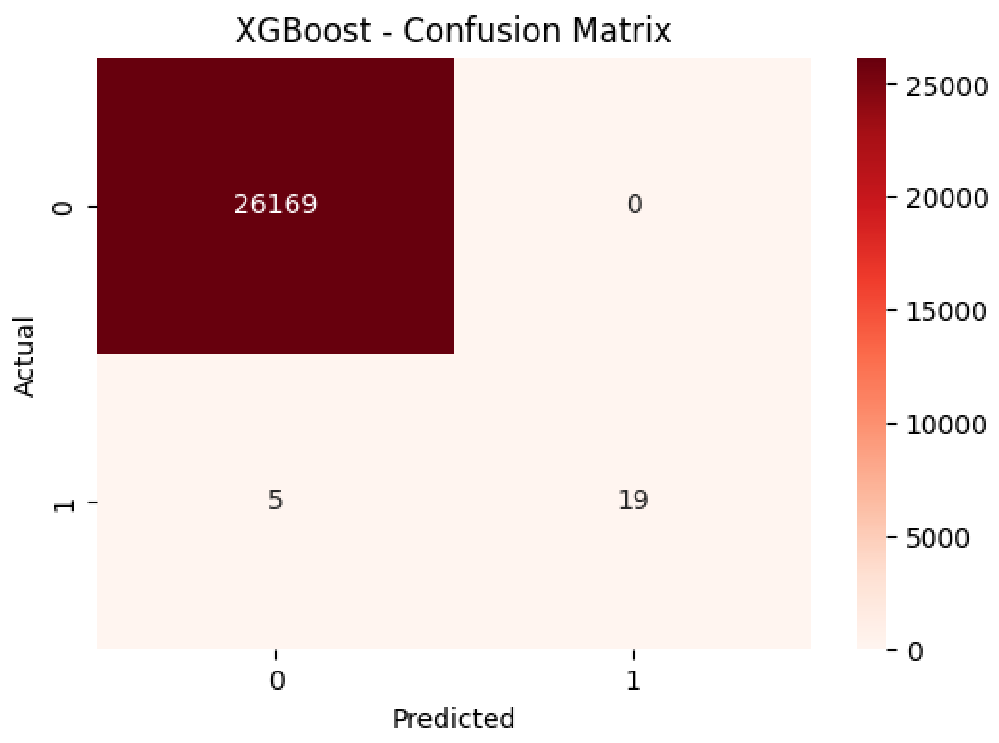


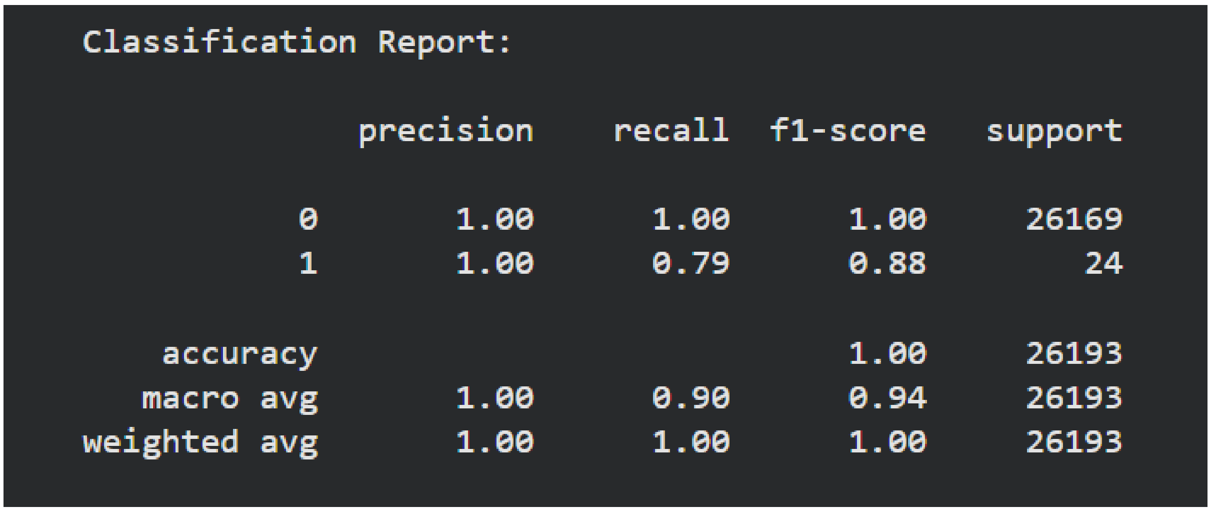


```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     26169
           1       0.00      0.00      0.00        24

    accuracy                           1.00     26193
   macro avg       0.50      0.50      0.50     26193
weighted avg       1.00      1.00      1.00     26193
```

Activity 4: Xgboost model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifier 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 done.

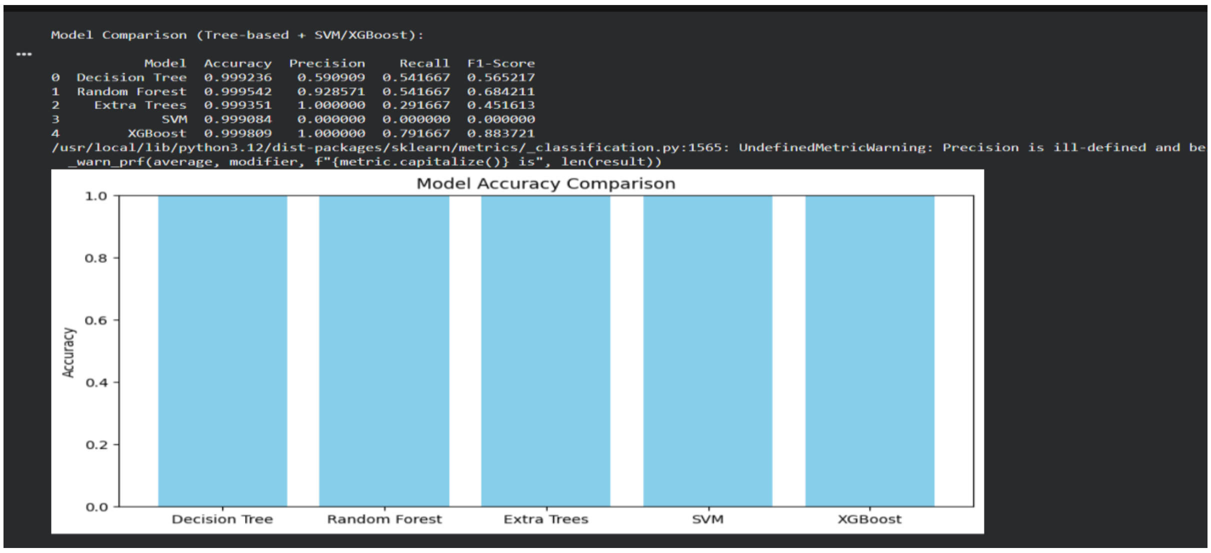



```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     26169
           1       1.00      0.79      0.88        24

    accuracy                           1.00     26193
   macro avg       1.00      0.90      0.94     26193
weighted avg       1.00      1.00      1.00     26193
```

Now let’s see the performance of all the models and save the best model

Activity 5: Compare the model

For comparing the above four models compareModel function is defined.

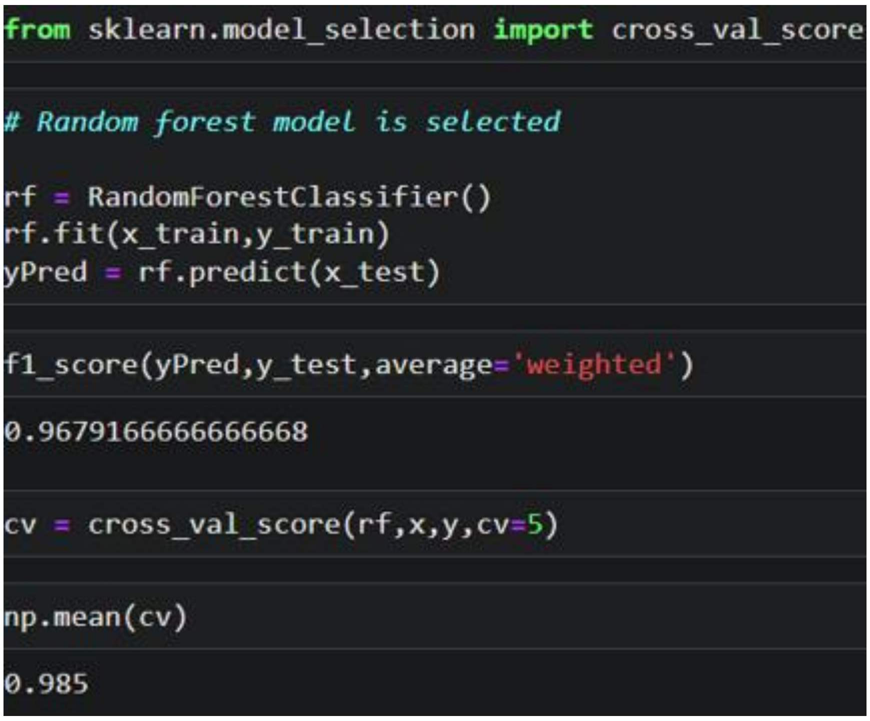


```
Model Comparison (Tree-based + SVM/XGBoost):
...
            Model  Accuracy  Precision    Recall  F1-Score
0   Decision Tree  0.999236   0.590909  0.541667  0.565217
1   Random Forest  0.999542   0.928571  0.541667  0.684211
2     Extra Trees  0.999351   1.000000  0.291667  0.451613
3             SVM  0.999084   0.000000  0.000000  0.000000
4         XGBoost  0.999809   1.000000  0.791667  0.883721
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

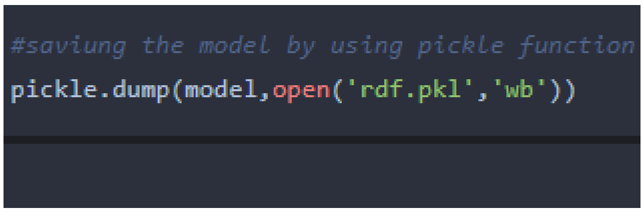
After calling the function, the results of models are displayed as output. From the four model Xgboost is performing well. From the below image, We can see the accuracy of the model. Xgboost is giving the accuracy of 94.7% with training data , 81.1% accuracy for the testing data.so we considering xgboost and deploying this model.

Activity 6: Evaluating performance of the model and saving the model

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



```
from sklearn.model_selection import cross_val_score

# Random forest model is selected

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred = rf.predict(x_test)

f1_score(yPred,y_test,average='weighted')

0.9679166666666668

cv = cross_val_score(rf,x,y,cv=5)

np.mean(cv)

0.985
```



```python
#saviung the model by using pickle function
pickle.dump(model,open('rdf.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 serverside script

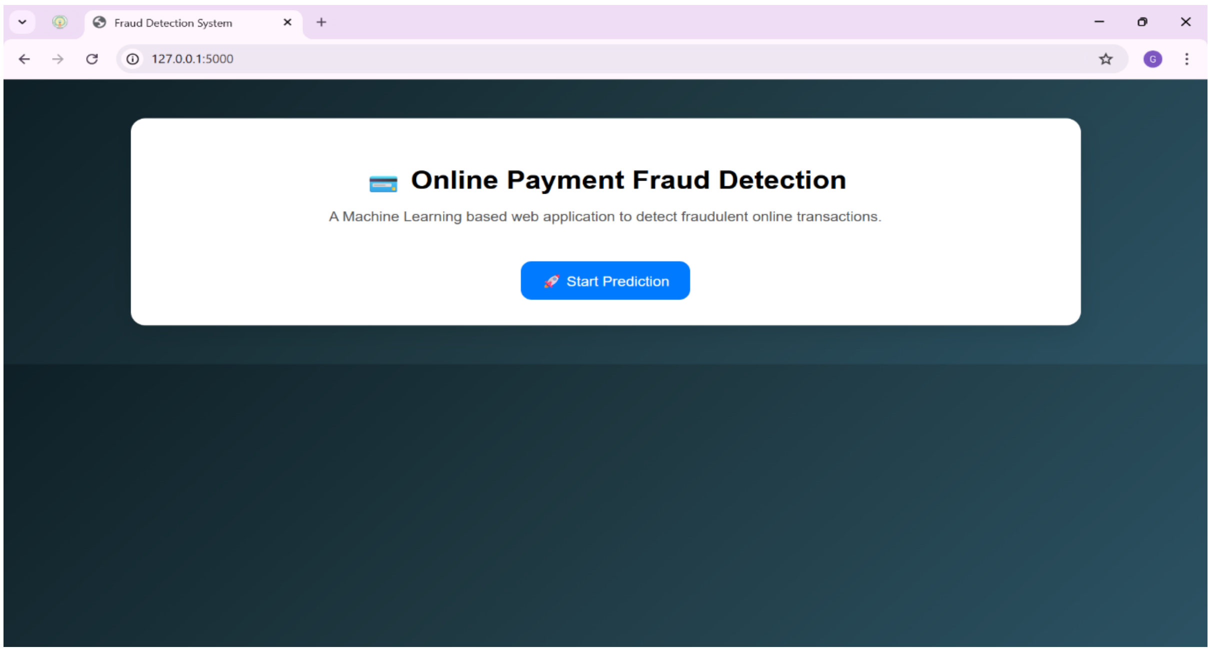
Activity1: Building Html Pages:

For this project create three HTML files namely

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

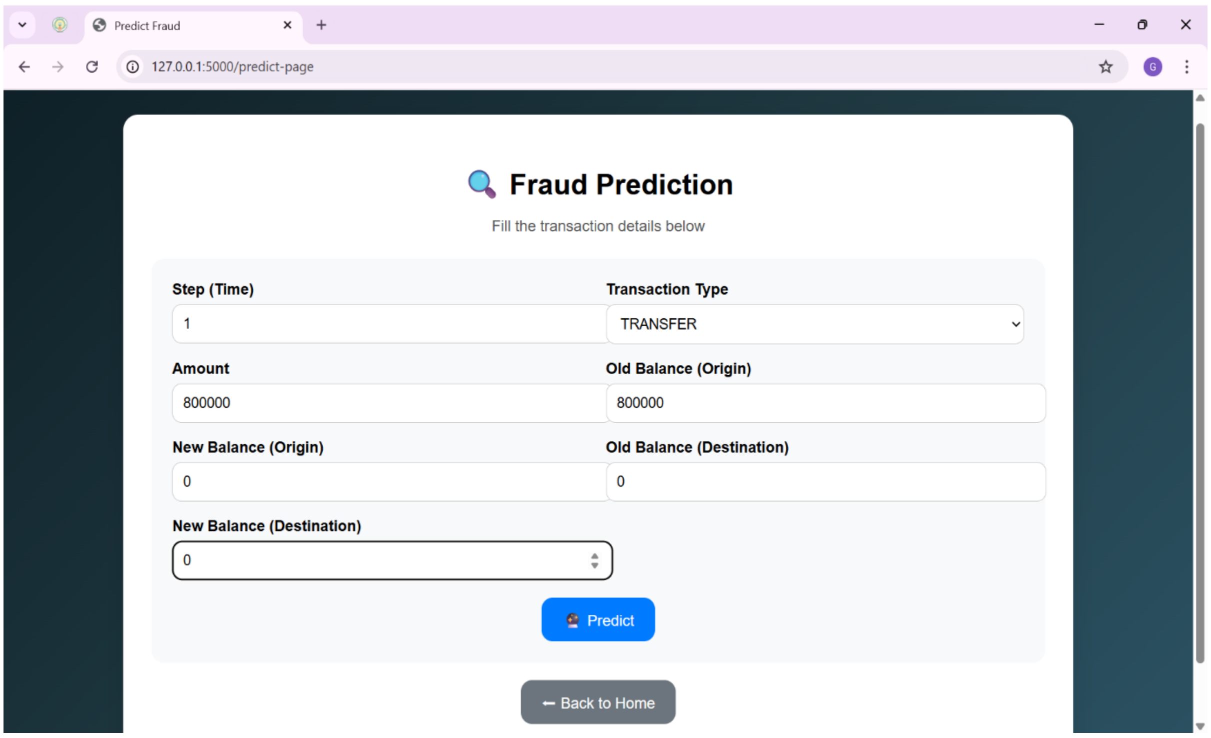
and save them in templates folder.

Let’s see how our home.html page looks like:


Fraud Detection System
127.0.0.1:5000
Online Payment Fraud Detection
A Machine Learning based web application to detect fraudulent online transactions.
Start Prediction

Now when you click on start prediction button you will get redirected to predict.html

Lets look how our predict.html file looks like:



Predict Fraud

127.0.0.1:5000/predict-page

Fraud Prediction

Fill the transaction details below

Step (Time): 1

Transaction Type: TRANSFER

Amount: 800000

Old Balance (Origin): 800000

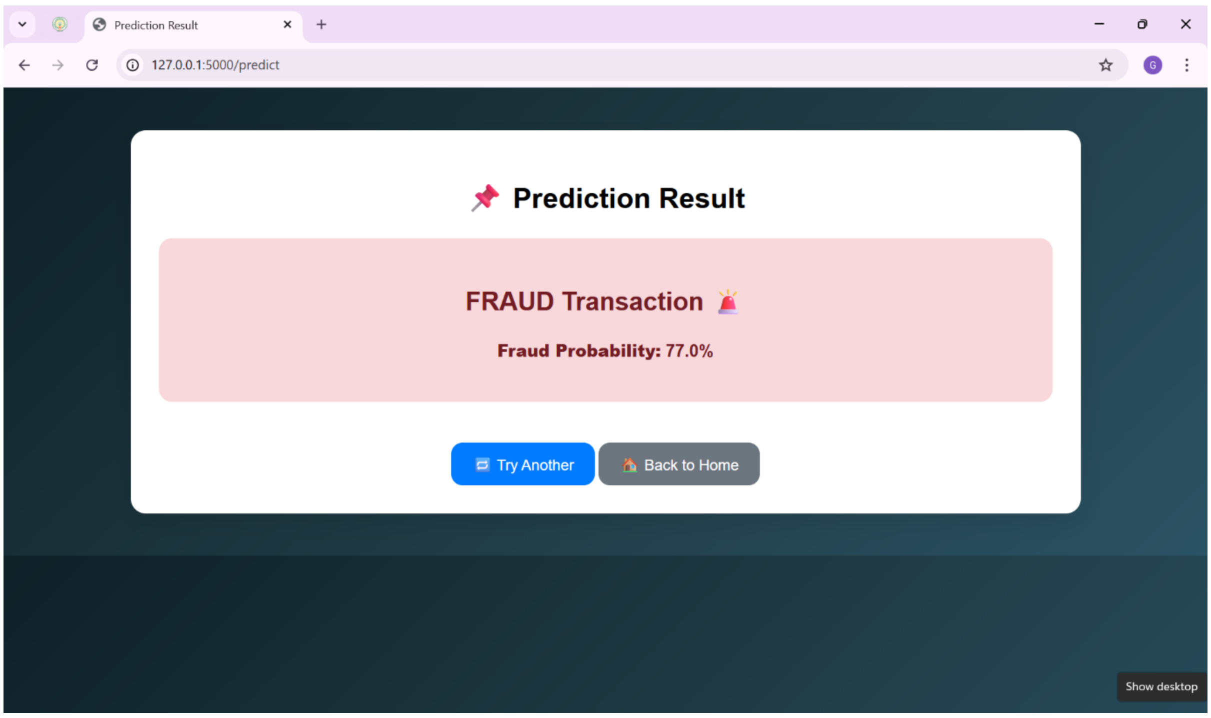
New Balance (Origin): 0

Old Balance (Destination): 0

New Balance (Destination): 0

Now when you click on predict button you will get redirected to submit.html

Lets look how our submit.html file looks like:



Prediction Result

127.0.0.1:5000/predict

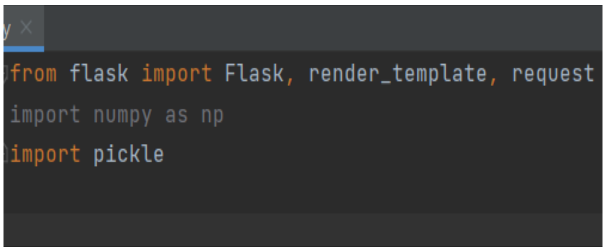
Prediction Result

FRAUD Transaction

Fraud Probability: 77.0%

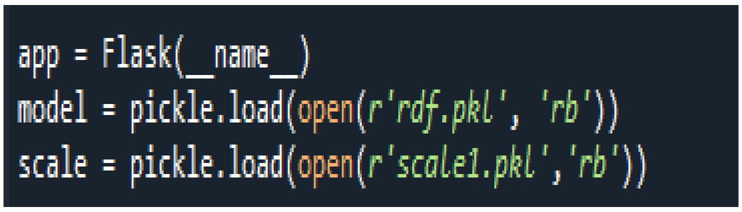
Activity 2: Build Python code:

Import the libraries



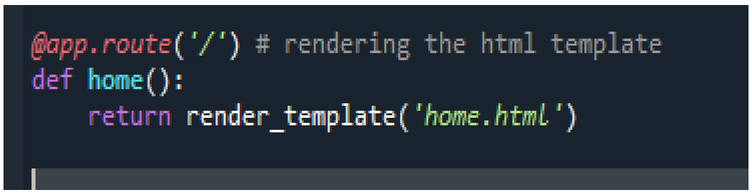
```python
from flask import Flask, render_template, request
import numpy as np
import pickle
```

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.



```python
app = Flask(__name__)
model = pickle.load(open(r'rdf.pkl', 'rb'))
scale = pickle.load(open(r'scale1.pkl','rb'))
```

Render HTML page:

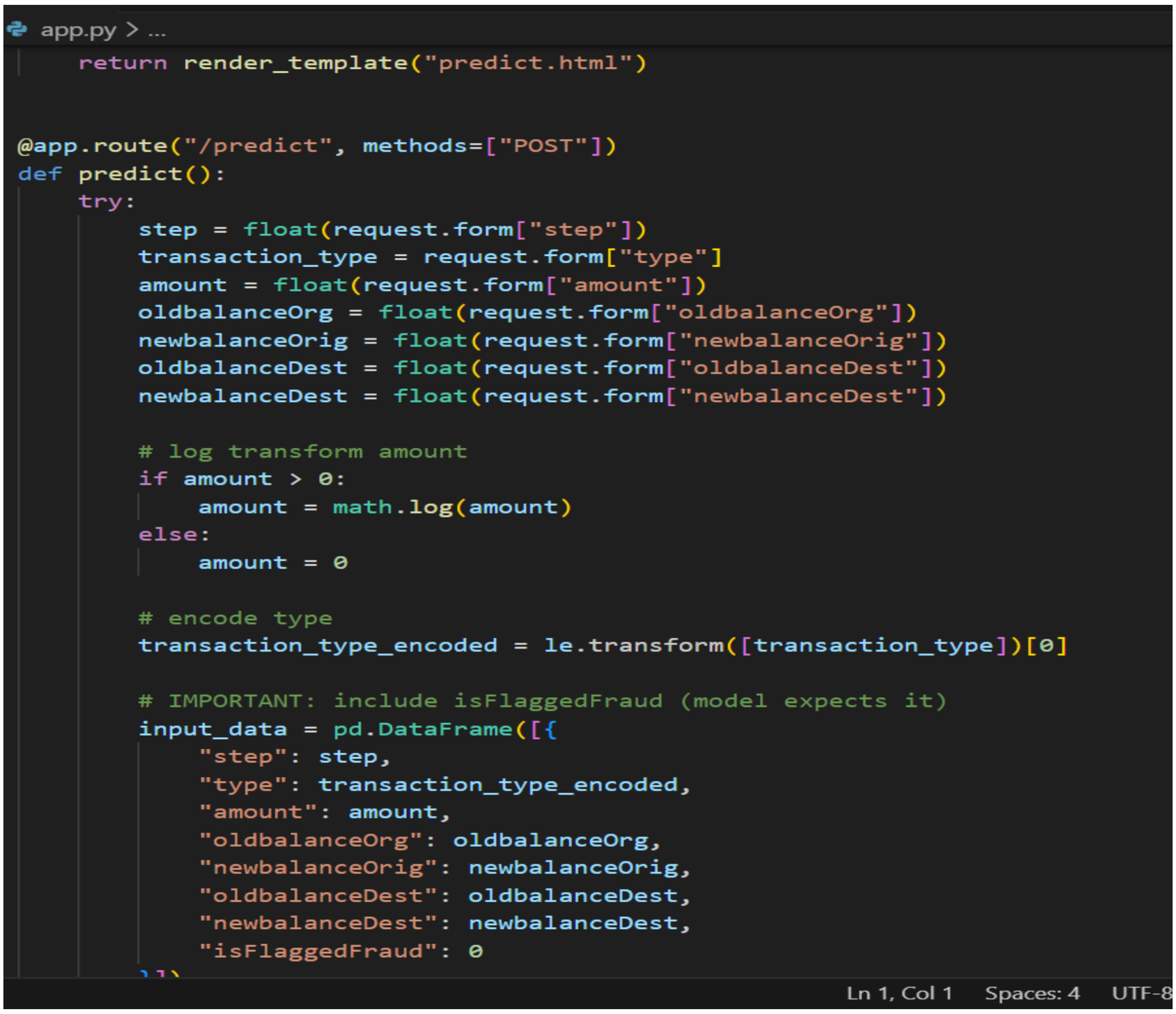


```python
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

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.

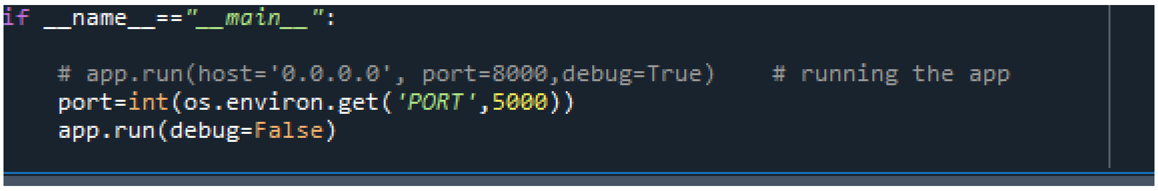
Retrieves the value from UI:



```
app.py > ...
    return render_template("predict.html")


@app.route("/predict", methods=["POST"])
def predict():
    try:
        step = float(request.form["step"])
        transaction_type = request.form["type"]
        amount = float(request.form["amount"])
        oldbalanceOrg = float(request.form["oldbalanceOrg"])
        newbalanceOrig = float(request.form["newbalanceOrig"])
        oldbalanceDest = float(request.form["oldbalanceDest"])
        newbalanceDest = float(request.form["newbalanceDest"])

        # log transform amount
        if amount > 0:
            amount = math.log(amount)
        else:
            amount = 0

        # encode type
        transaction_type_encoded = le.transform([transaction_type])[0]

        # IMPORTANT: include isFlaggedFraud (model expects it)
        input_data = pd.DataFrame([{
            "step": step,
            "type": transaction_type_encoded,
            "amount": amount,
            "oldbalanceOrg": oldbalanceOrg,
            "newbalanceOrig": newbalanceOrig,
            "oldbalanceDest": oldbalanceDest,
            "newbalanceDest": newbalanceDest,
            "isFlaggedFraud": 0
```

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 submit.html page earlier.

Main Function:

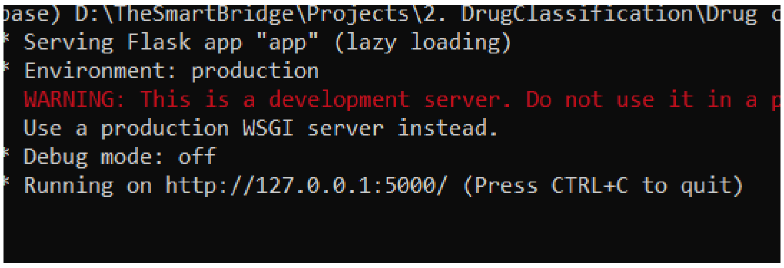


```
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)
```

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.

- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```