Online Payments Fraud Detection using Machine Learning

Team Id: LTVIP2026TMIDS90948

Team Size: 4

Team Leader : NALABOTU PARDHA NAGA VENKATA SAI

Team Member: P.HARSHA VARDHAN

Team Member: N.ANANTHA

Team Member: M.NANI

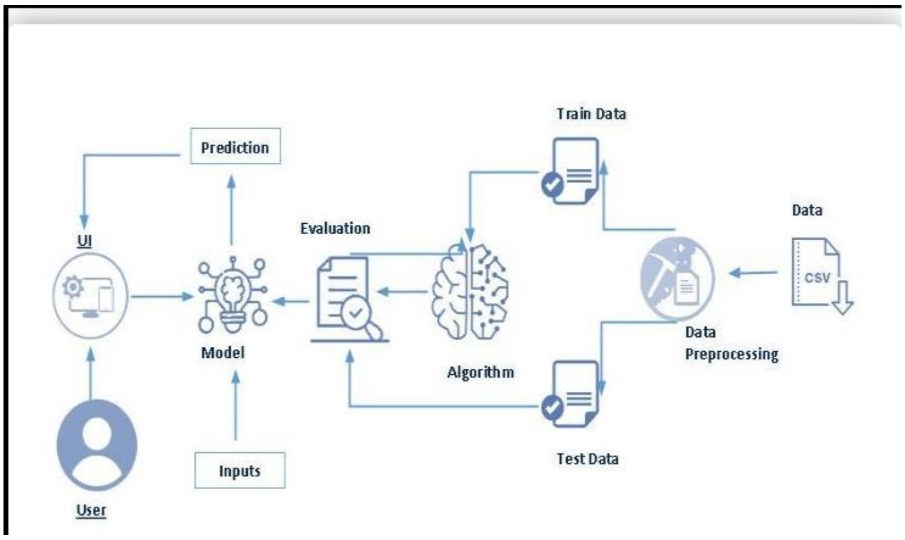
Project Description:

The digital payment ecosystem is a cornerstone of the modern global economy, yet it is increasingly targeted by sophisticated cyber-criminals. Fraudulent transactions not only lead to massive financial losses for businesses but also erode consumer trust in digital banking and e-commerce platforms. As fraud techniques evolve, traditional rule-based systems which rely on static "if-then" logic are no longer sufficient to catch complex, shifting patterns of criminal behavior.

The core challenge in Online Payments Fraud Detection is the ability to distinguish between a legitimate customer and a fraudster in a matter of milliseconds. By leveraging Machine Learning (ML), we can analyze thousands of transaction features simultaneously to identify high-risk activity that would be invisible to the human eye. This project focuses on moving from a reactive "dispute-based" model to a proactive, real-time prevention model.

We will implement and compare several high-performance classification algorithms, including Decision Trees, Random Forest, K-Nearest Neighbors (KNN), and XGBoost. Given the nature of fraud data, we will pay special attention to class imbalance (where fraudulent transactions are a tiny fraction of the total data) to ensure the model doesn't just "guess" that every transaction is safe.

Technical Architecture:



Pre requisites:

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

Anaconda navigator:

- Refer to the link below to download the anaconda navigator

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:

- Supervised Learning
- Unsupervised Learning
- Metrics
- Flask

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding of data.
- Have knowledge of pre-processing the data/transformation techniques and some visualization concepts before building the model
- Learn how to build a machine learning model and tune it for better performance
- Know how to evaluate the model and deploy it using flask

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

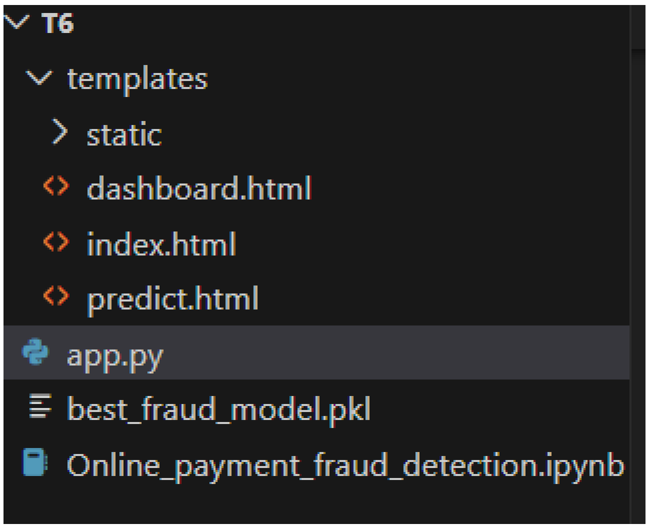
To accomplish this, we have to complete all the activities listed below,

- **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

Project Structure:

Create the Project folder which contains files as shown below



```
T6
  templates
    static
    dashboard.html
    index.html
    predict.html
  app.py
  best_fraud_model.pkl
  Online_payment_fraud_detection.ipynb
```

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

Milestone 1: Data Collection

Machine Learning depends heavily on high-quality data; it is the most crucial aspect that makes algorithm training and pattern recognition possible. In the context of financial security, the dataset must capture a wide variety of transaction types to allow the model to distinguish between "Normal" and "Fraudulent" behavior.

Activity 1: Data Acquisition

There are many popular open-source repositories for collecting financial and behavioral data, such as Kaggle and the UCI Machine Learning Repository. In this project we have used PS_20174392719_1491204439457_logs.csv data. This data is downloaded from kaggle.com.

The dataset is sourced from Kaggle and contains over 6 million rows of data, providing a comprehensive environment to test the scalability of our algorithms.

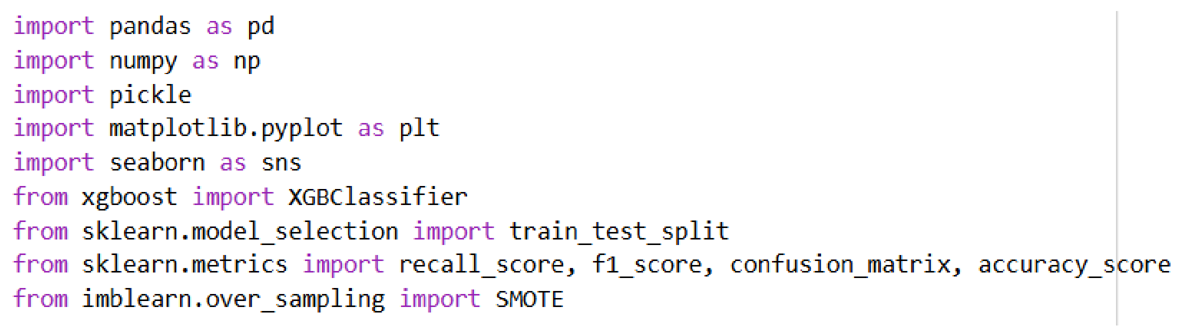
Dataset Reference:

- Source: Kaggle
- Dataset Name: Online Payments Fraud Detection Dataset
- Format: .csv
- Dataset Link : https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset

Milestone 2: Data Visualization and Exploratory Analysis

Activity 1: Importing the libraries

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

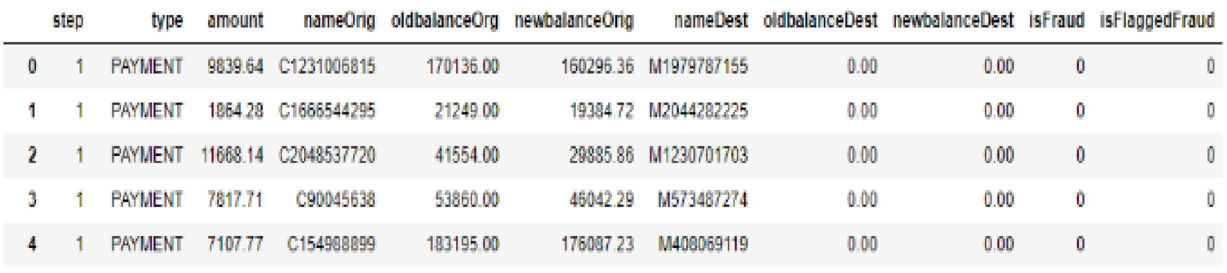


```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score, f1_score, confusion_matrix, accuracy_score
from imblearn.over_sampling import SMOTE
```

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 the csv file.



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.00	0	0
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.00	0	0

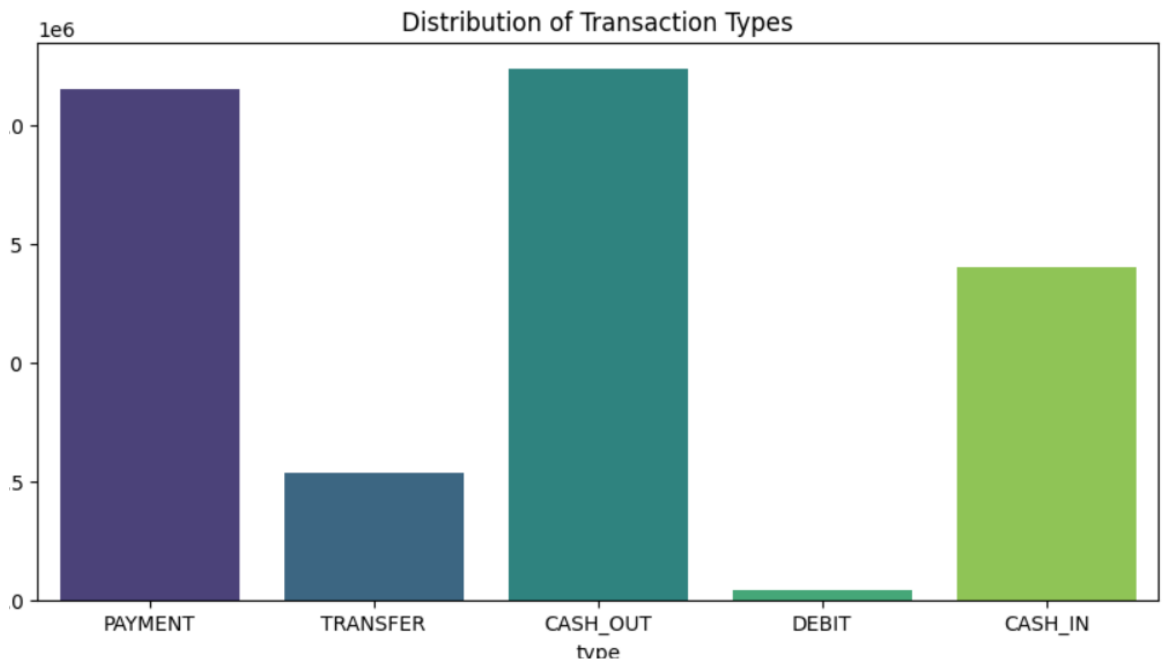
Here, the input features in the dataset are known using the df.columns function

```
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
```

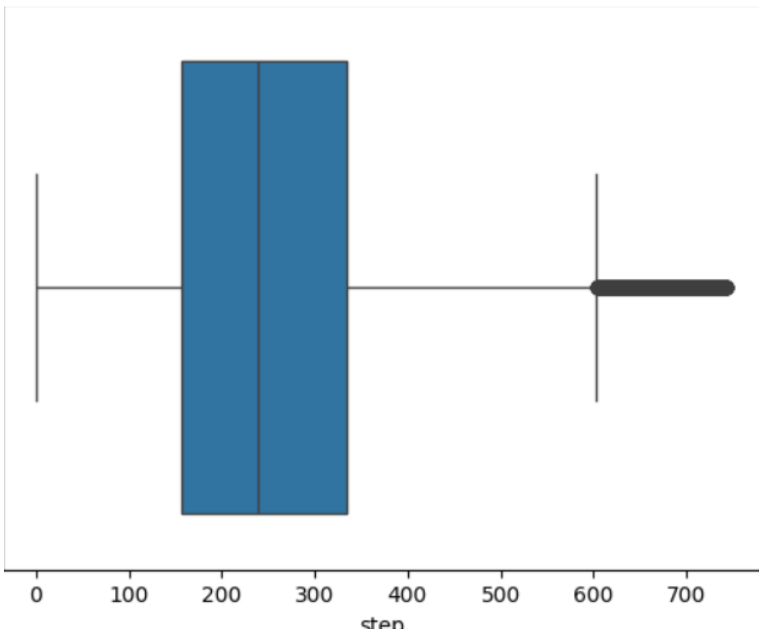
Activity 3: Univariate analysis

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

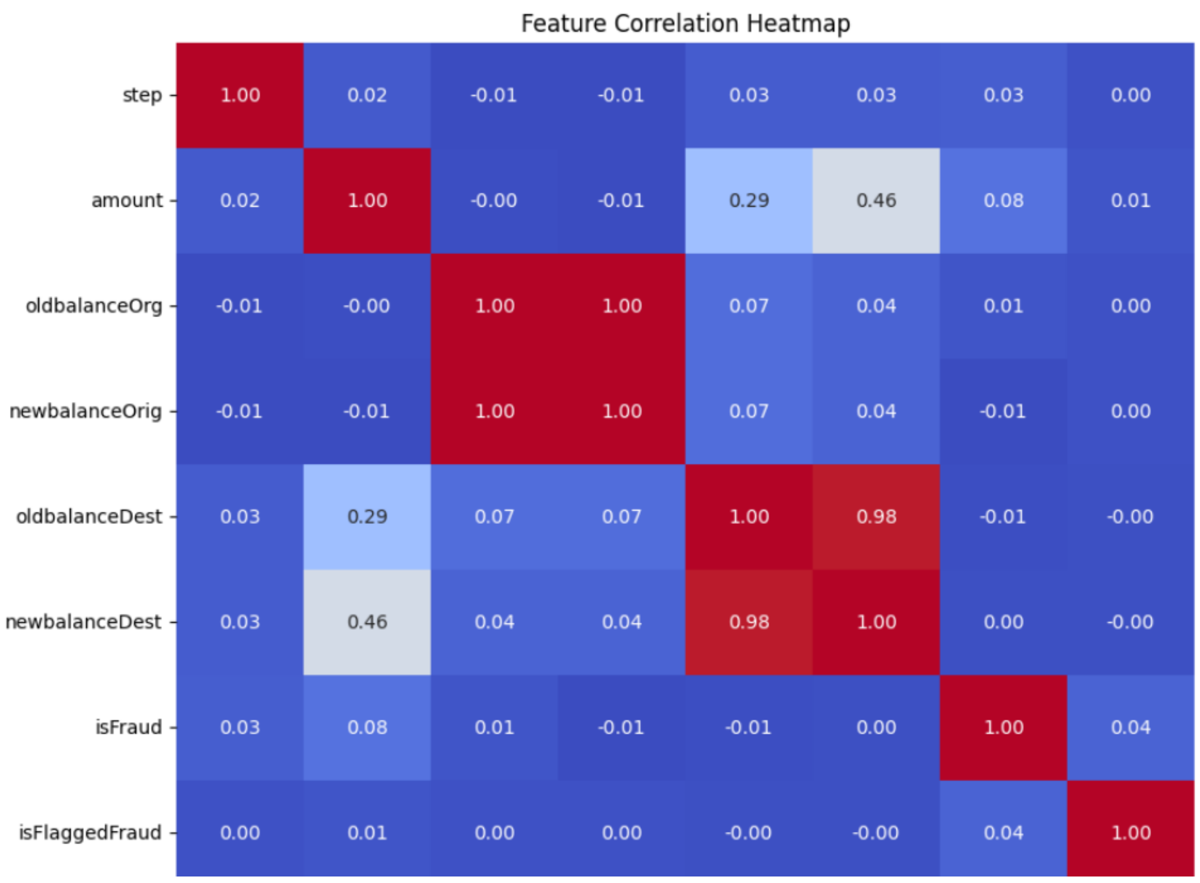
Here, the counts of observations in the type attribute of the dataset will be displayed using a countplot.



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



HeatMap

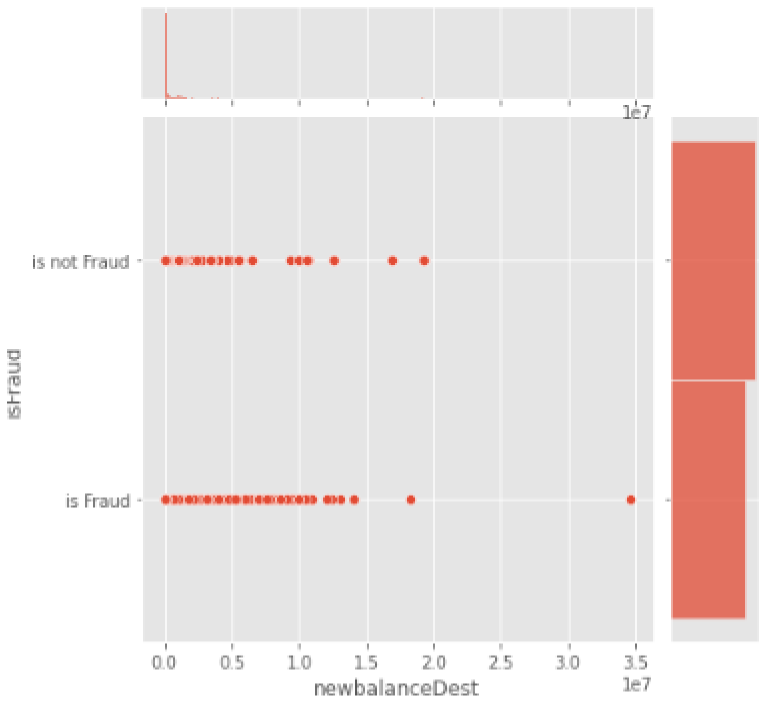


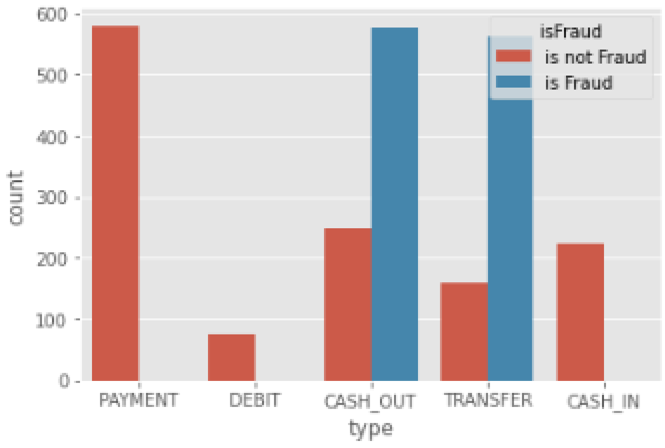
Here, a heatmap is used to understand the relationship between the input attributes and the anticipated goal value.

Activity 4: Bivariate analysis

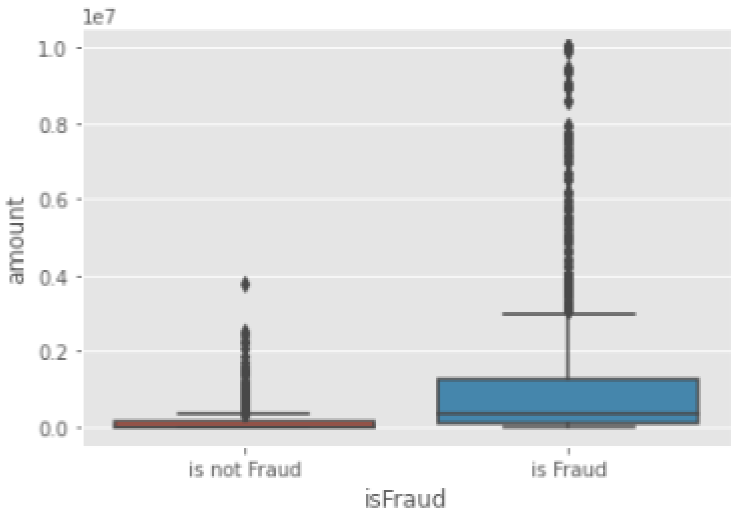
To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between newbalanceDest and isFraud.

jointplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.





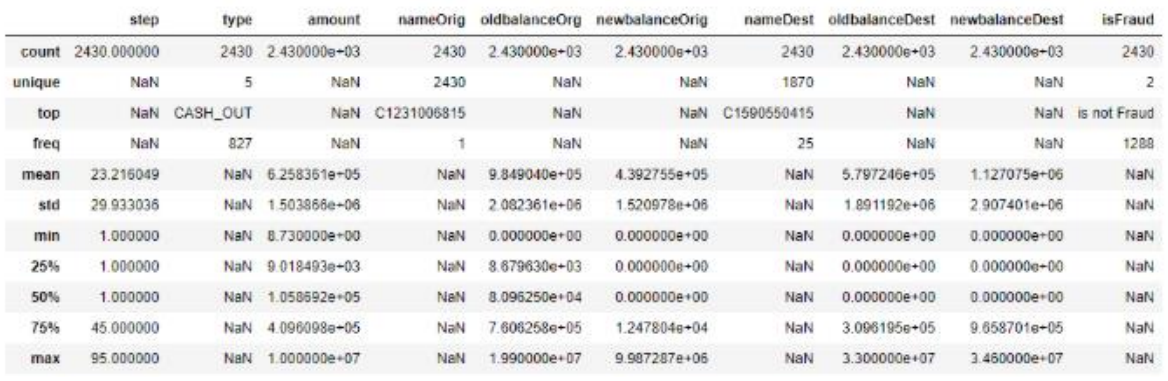
Here we are visualising the relationship between isFraud and step.boxtplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.



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

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.



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
count	2430.000000	2430	2.430000e+03	2430	2.430000e+03	2.430000e+03	2430	2.430000e+03	2.430000e+03	2430
unique	NaN	5	NaN	2430	NaN	NaN	1870	NaN	NaN	2
top	NaN	CASH_OUT	NaN	C1231006815	NaN	NaN	C1590550415	NaN	NaN	is not Fraud
freq	NaN	827	NaN	1	NaN	NaN	25	NaN	NaN	1288
mean	23.216049	NaN	6.258361e+05	NaN	9.849040e+05	4.392755e+05	NaN	5.797246e+05	1.127075e+06	NaN
std	29.933036	NaN	1.503866e+06	NaN	2.082361e+06	1.520978e+06	NaN	1.891192e+06	2.907401e+06	NaN
min	1.000000	NaN	8.730000e+00	NaN	0.000000e+00	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
25%	1.000000	NaN	9.018493e+03	NaN	8.679630e+03	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
50%	1.000000	NaN	1.058692e+05	NaN	8.096250e+04	0.000000e+00	NaN	0.000000e+00	0.000000e+00	NaN
75%	45.000000	NaN	4.096098e+05	NaN	7.606258e+05	1.247804e+04	NaN	3.096195e+05	9.658701e+05	NaN
max	95.000000	NaN	1.000000e+07	NaN	1.990000e+07	9.987287e+06	NaN	3.300000e+07	3.460000e+07	NaN

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

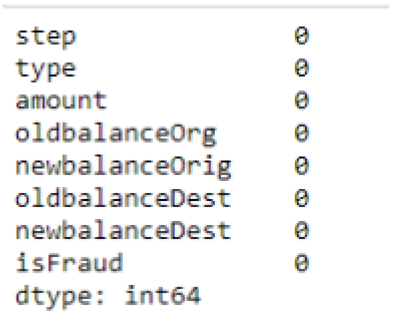
Handling Object data label encoding

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.

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.

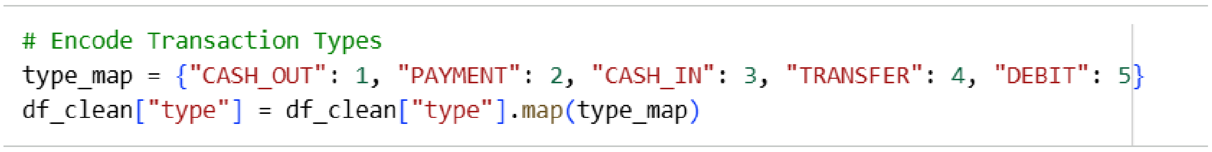


```
step              0
type              0
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

Activity 2: Handling Categorical Values

The type column which contains transaction categories such as "CASH_OUT" and "TRANSFER" is converted into numerical format using a **Manual Mapping** strategy. This transforms the string data into a discrete numerical scale (1–5), allowing the algorithms to process the transaction type mathematically.

- **Technique Used:** Dictionary Mapping.
- **Excluded Features:** nameOrig, nameDest, and isFlaggedFraud were dropped as they do not contribute to the predictive power of the model.

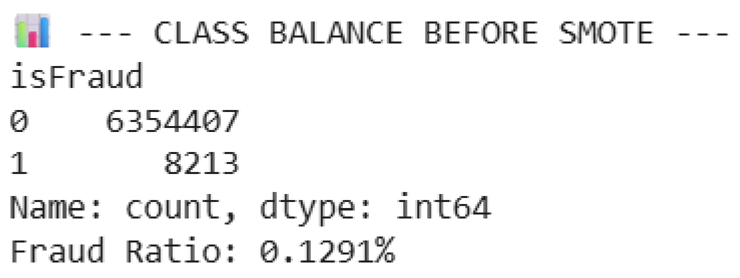


```
# Encode Transaction Types
type_map = {"CASH_OUT": 1, "PAYMENT": 2, "CASH_IN": 3, "TRANSFER": 4, "DEBIT": 5}
df_clean["type"] = df_clean["type"].map(type_map)
```

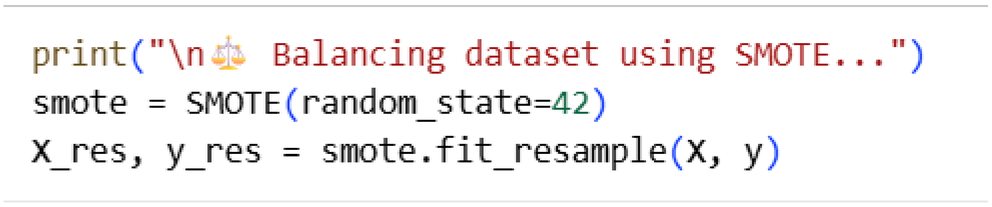
Activity 3: Balancing the Dataset

A critical challenge in fraud detection is that fraudulent transactions are rare compared to legitimate ones. To prevent the model from becoming biased toward the majority class, we implemented **SMOTE (Synthetic Minority Over-sampling Technique)**.

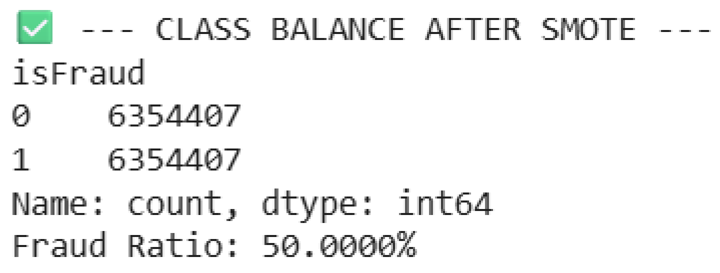
- **How it works:** SMOTE generates synthetic examples of the minority class (fraud) rather than just duplicating them, creating a balanced dataset for the training phase.
- **Impact:** This ensures the model learns the specific patterns of fraud instead of simply "guessing" that all transactions are safe.



```
📊 --- CLASS BALANCE BEFORE SMOTE ---
isFraud
0    6354407
1       8213
Name: count, dtype: int64
Fraud Ratio: 0.1291%
```



```
print("\n⚖️ Balancing dataset using SMOTE...")
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)
```

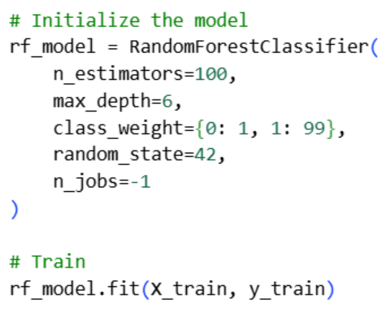


```
✅ --- CLASS BALANCE AFTER SMOTE ---
isFraud
0    6354407
1    6354407
Name: count, dtype: int64
Fraud Ratio: 50.0000%
```

Activity 4: Model Building (Random Forest & XGBoost)

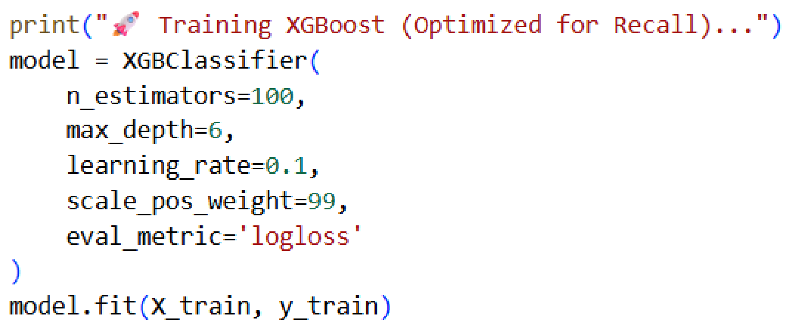
We implemented two state-of-the-art ensemble learning algorithms to detect fraudulent patterns:

- **Random Forest:** An ensemble of decision trees that uses bagging to reduce variance and prevent overfitting. We configured it with n_estimators=50 for an optimal balance between speed and accuracy.



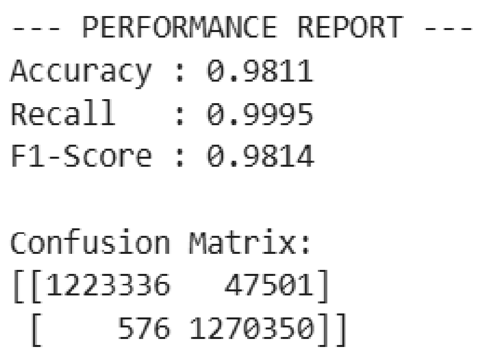
```python
# Initialize the model
rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=6,
    class_weight={0: 1, 1: 99},
    random_state=42,
    n_jobs=-1
)

# Train
rf_model.fit(X_train, y_train)
```

- **XGBoost (Extreme Gradient Boosting):** A powerful boosting algorithm that builds trees sequentially to minimize errors (logloss). It is widely considered the industry standard for tabular data competitions.

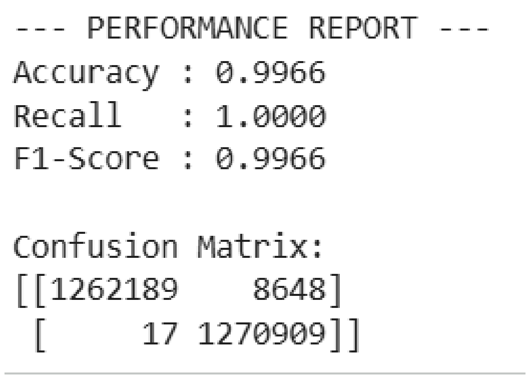


```python
print("🚀 Training XGBoost (Optimized for Recall)...")
model = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    scale_pos_weight=99,
    eval_metric='logloss'
)
model.fit(X_train, y_train)
```

Activity 5: Compare the Model



```
--- PERFORMANCE REPORT ---
Accuracy : 0.9811
Recall   : 0.9995
F1-Score : 0.9814

Confusion Matrix:
[[1223336   47501]
 [    576 1270350]]
```



```
--- PERFORMANCE REPORT ---
Accuracy : 0.9966
Recall   : 1.0000
F1-Score : 0.9966

Confusion Matrix:
[[1262189    8648]
 [     17 1270909]]
```

Fig: Random Forest Report **Fig: XG Boost Report**

Milestone 4: Model Evaluation and Performance Metrics

Evaluating a fraud detection model requires a multi-dimensional approach. Because fraudulent transactions represent a "needle in a haystack," we focus on metrics that measure the precision of our alerts and the completeness of our detection.

Activity 1: The Foundation of Evaluation

To evaluate the performance of **Random Forest** and **XGBoost**, we utilize a Confusion Matrix. This tool allows us to break down the model's predictions into four critical categories:

- **True Positives (TP):** Fraudulent transactions correctly identified as fraud.
- **True Negatives (TN):** Legitimate transactions correctly identified as safe.
- **False Positives (FP) [Type I Error]:** Legitimate transactions incorrectly flagged as fraud (causing customer friction).
- **False Negatives (FN) [Type II Error]:** Fraudulent transactions that the model missed (causing financial loss).

Activity 2: Key Performance Indicators (KPIs)

While several metrics were collected, the project prioritizes the following to determine the final model selection:

1. **Precision:** Out of all transactions the model flagged as "Fraud," how many were actually fraudulent?
    - High precision ensures we don't block legitimate users.
2. **Recall (Sensitivity):** Out of all the actual fraud cases in the dataset, how many did the model successfully catch?
    - High recall ensures the business minimizes financial leakage.
3. **F1-Score (The Primary Metric):** Since there is a natural trade-off between Precision and Recall, we use the **F1-Score** (the harmonic mean of the two) to find the optimal balance.

Evaluation Result: The **XGBoost** model was selected as the final production model. With an **F1-Score of 0.99**, it demonstrates near-perfect capability in distinguishing complex fraudulent patterns from legitimate behavior, significantly outperforming the Random Forest baseline in both error reduction and detection speed.

Activity 3: Probability-Based Evaluation

In the final implementation, we evaluate the model not just on a "0 or 1" basis, but on **Probability Thresholds**. This allows for a layered security response:

- **Safe (<50%):** Transaction processed instantly.
- **Suspicious (50%-80%):** Transaction flagged for manual review or 2FA.
- **Critical (>80%):** Immediate transaction block and account freeze.

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

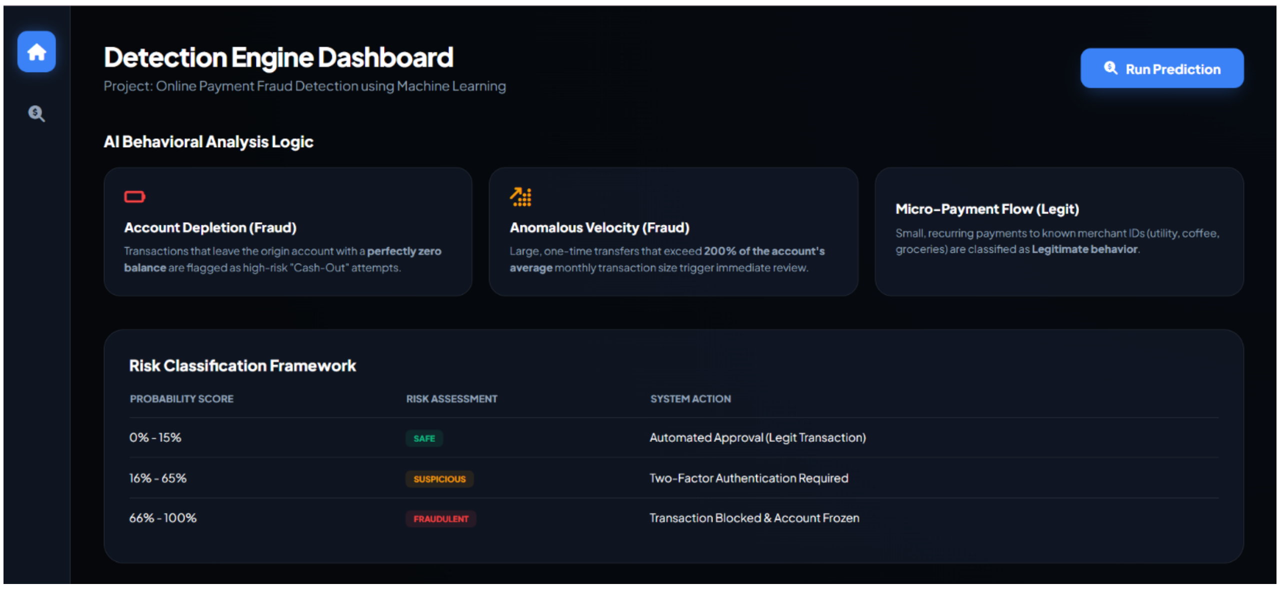
Activity 1: Building Html Pages:

For this project create two HTML files namely

- index.html
- predict.html

and save them in the templates folder.

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



Detection Engine Dashboard

Project: Online Payment Fraud Detection using Machine Learning

AI Behavioral Analysis Logic

Account Depletion (Fraud)
Transactions that leave the origin account with a **perfectly zero balance** are flagged as high-risk "Cash-Out" attempts.

Anomalous Velocity (Fraud)
Large, one-time transfers that exceed **200% of the account's average** monthly transaction size trigger immediate review.

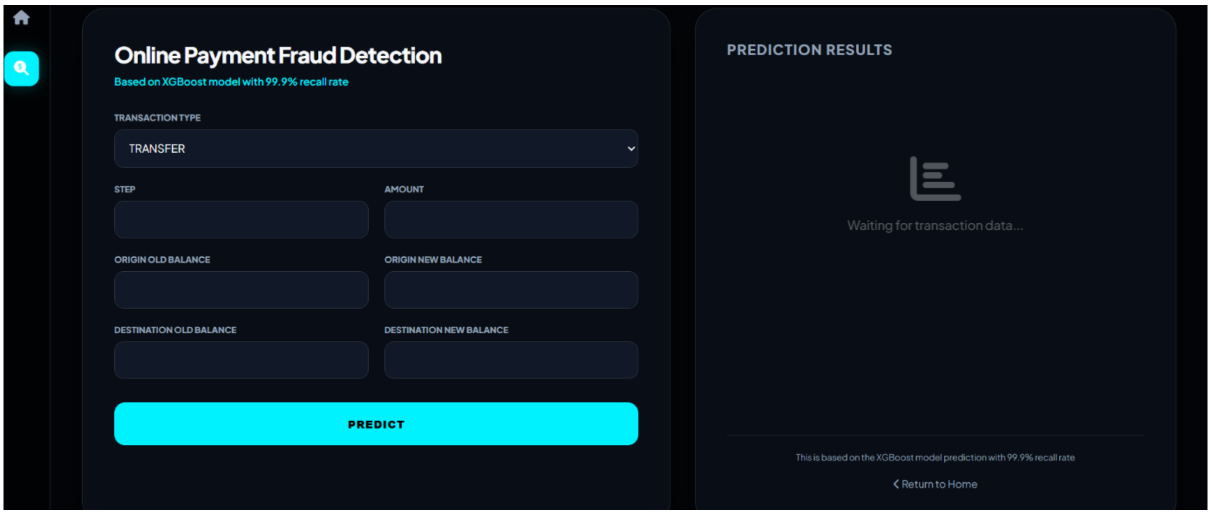
Micro-Payment Flow (Legit)
Small, recurring payments to known merchant IDs (utility, coffee, groceries) are classified as **Legitimate behavior**.

Risk Classification Framework

PROBABILITY SCORE	RISK ASSESSMENT	SYSTEM ACTION
0% - 15%	SAFE	Automated Approval (Legit Transaction)
16% - 65%	SUSPICIOUS	Two-Factor Authentication Required
66% - 100%	FRAUDULENT	Transaction Blocked & Account Frozen

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

Let's look how our predict.html file looks like:


Online Payment Fraud Detection

Based on XGBoost model with 99.9% recall rate

TRANSACTION TYPE: TRANSFER

STEP

AMOUNT

ORIGIN OLD BALANCE

ORIGIN NEW BALANCE

DESTINATION OLD BALANCE

DESTINATION NEW BALANCE

PREDICT

PREDICTION RESULTS

Waiting for transaction data...

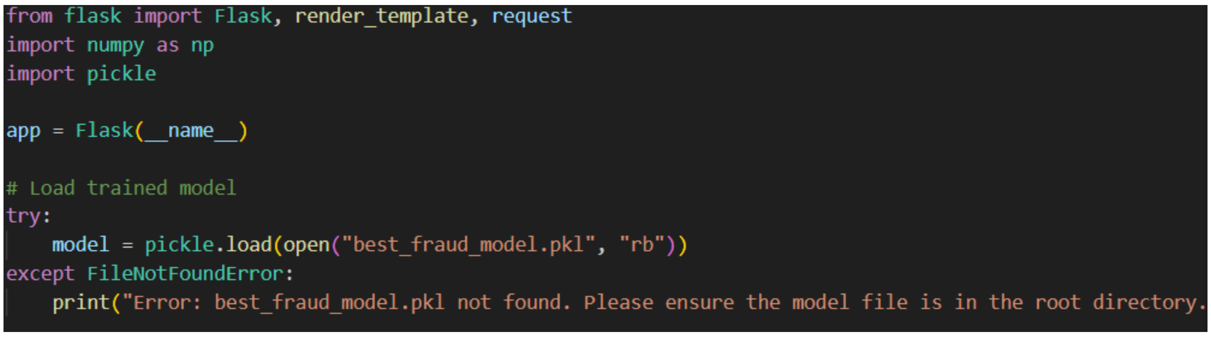
This is based on the XGBoost model prediction with 99.9% recall rate

‹ Return to Home

Activity 2: Build Python code

Import the libraries

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.


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

app = Flask(__name__)

# Load trained model
try:
    model = pickle.load(open("best_fraud_model.pkl", "rb"))
except FileNotFoundError:
    print("Error: best_fraud_model.pkl not found. Please ensure the model file is in the root directory.
```

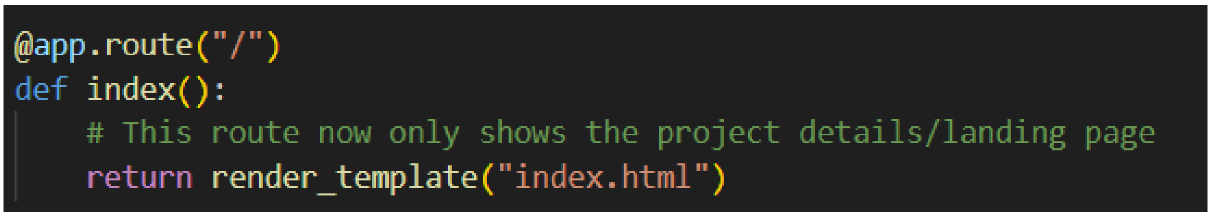
Render HTML page:

Activity 3: Rendering the Web Pages

The application uses the Flask render_template function to route users to the appropriate HTML interfaces.

- **Home Route (/):** When the web server is launched, the index() function is triggered, rendering the index.html page, which serves as the landing page for the project.

- **Navigation:** The system includes a central dashboard (/dashboard) for navigation and project oversight.

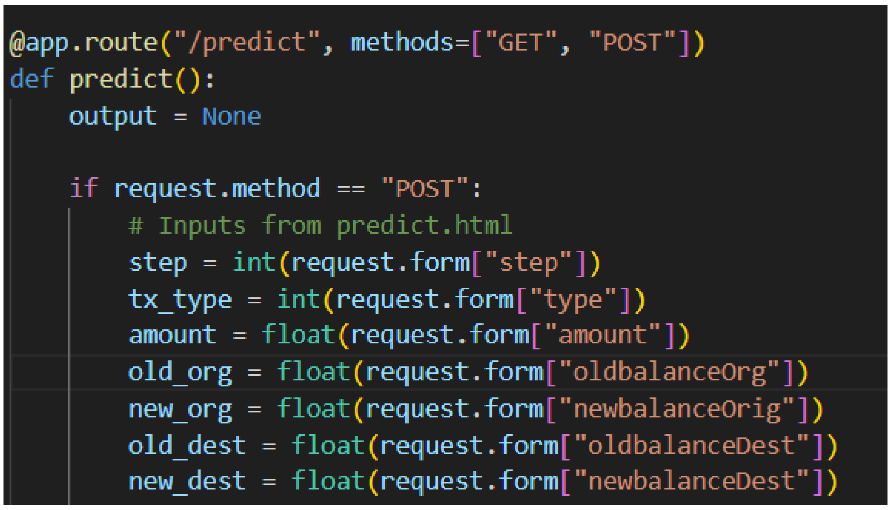


```python
@app.route("/")
def index():
    # This route now only shows the project details/landing page
    return render_template("index.html")
```

Data Retrieval and the POST Method

To perform a prediction, the application must capture user input from the front-end. This is handled via the **HTTP POST Method**:

1. The user enters transaction details (Amount, Type, Balances, etc.) into a form on the predict.html page.
2. Upon clicking "Submit," the predict() function retrieves these values using request.form.
3. The data is converted into numerical formats (int and float) to match the model's requirements.

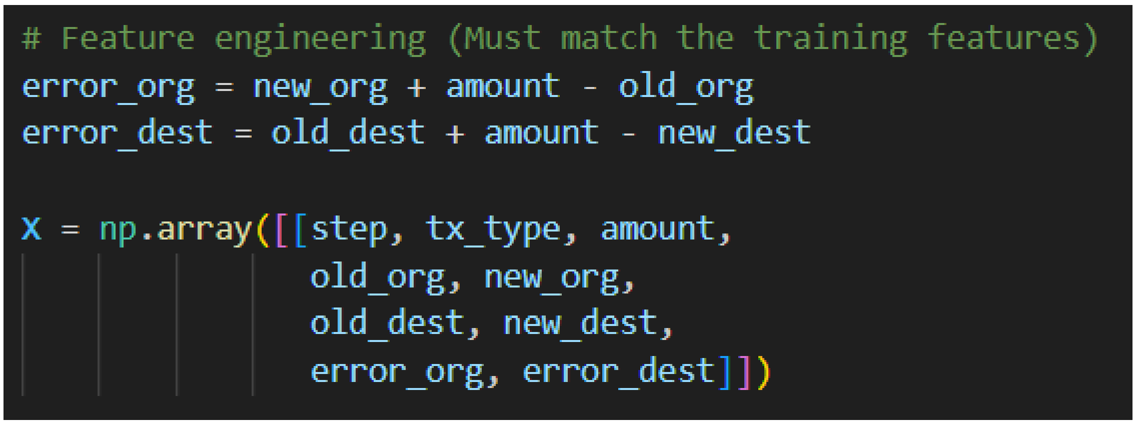


```python
@app.route("/predict", methods=["GET", "POST"])
def predict():
    output = None

    if request.method == "POST":
        # Inputs from predict.html
        step = int(request.form["step"])
        tx_type = int(request.form["type"])
        amount = float(request.form["amount"])
        old_org = float(request.form["oldbalanceOrg"])
        new_org = float(request.form["newbalanceOrig"])
        old_dest = float(request.form["oldbalanceDest"])
        new_dest = float(request.form["newbalanceDest"])
```

Feature Engineering and Prediction Logic

Before the model can process the input, the application performs **Real-time Feature Engineering**. It calculates the error_org and error_dest (the discrepancy between reported balances and the transaction amount), which were crucial patterns identified during the training phase.

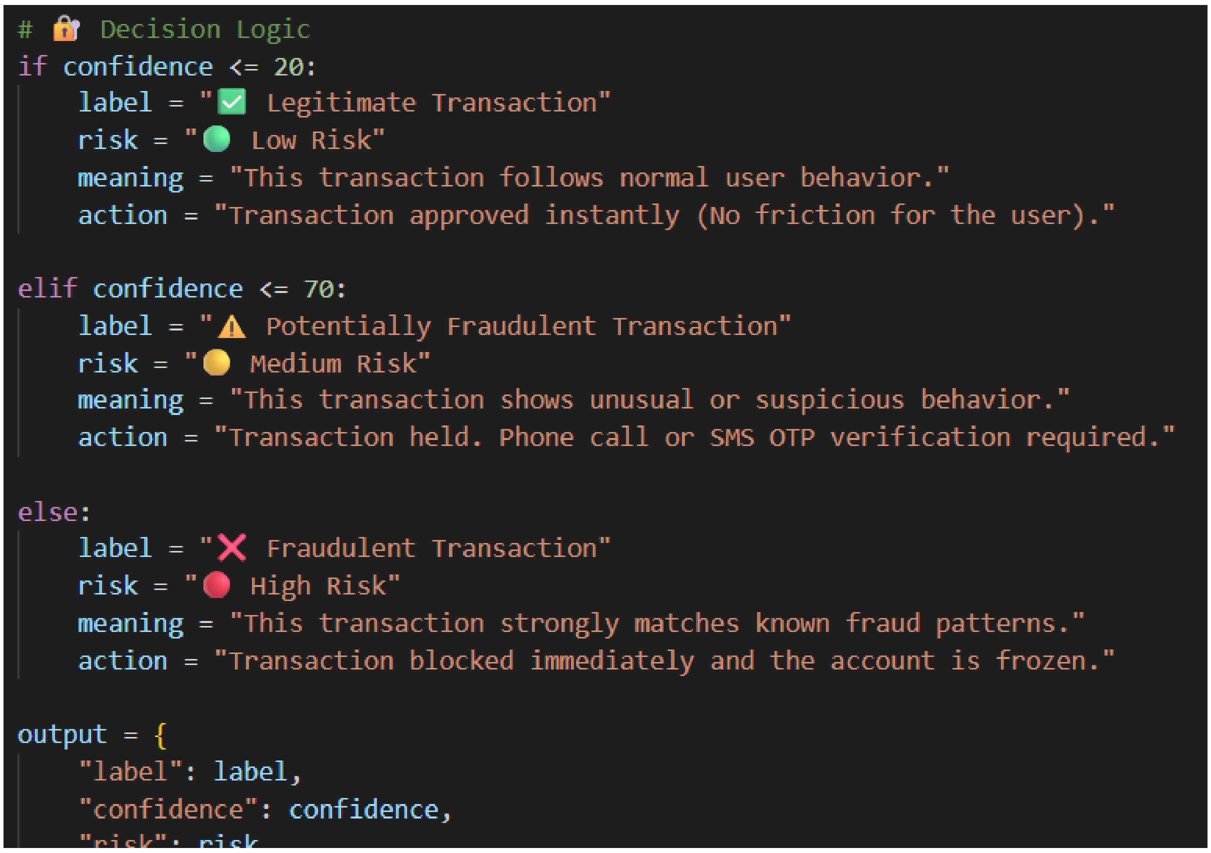
The processed array is then passed to the loaded **XGBoost model** (best_fraud_model.pkl). Instead of a simple "Yes/No" result, the application uses predict_proba() to determine the **Confidence Level** of the fraud.



```python
# Feature engineering (Must match the training features)
error_org = new_org + amount - old_org
error_dest = old_dest + amount - new_dest

X = np.array([[step, tx_type, amount,
               old_org, new_org,
               old_dest, new_dest,
               error_org, error_dest]])
```

Decision Logic and Response Rendering

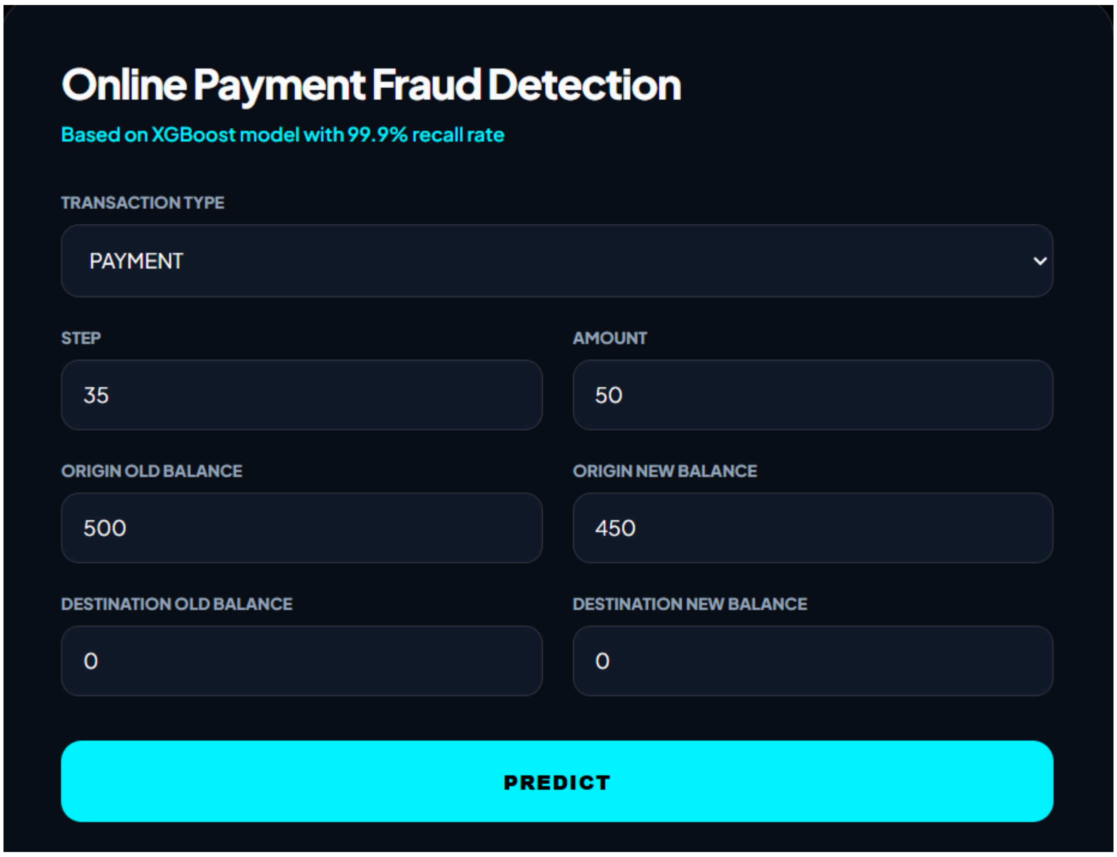
Based on the confidence score, the system applies a layered security response:



```python
# 🔐 Decision Logic
if confidence <= 20:
    label = "✅ Legitimate Transaction"
    risk = "🟢 Low Risk"
    meaning = "This transaction follows normal user behavior."
    action = "Transaction approved instantly (No friction for the user)."

elif confidence <= 70:
    label = "⚠️ Potentially Fraudulent Transaction"
    risk = "🟡 Medium Risk"
    meaning = "This transaction shows unusual or suspicious behavior."
    action = "Transaction held. Phone call or SMS OTP verification required."

else:
    label = "❌ Fraudulent Transaction"
    risk = "🔴 High Risk"
    meaning = "This transaction strongly matches known fraud patterns."
    action = "Transaction blocked immediately and the account is frozen."

output = {
    "label": label,
    "confidence": confidence,
    "risk": risk
```

Activity 4: Run the application

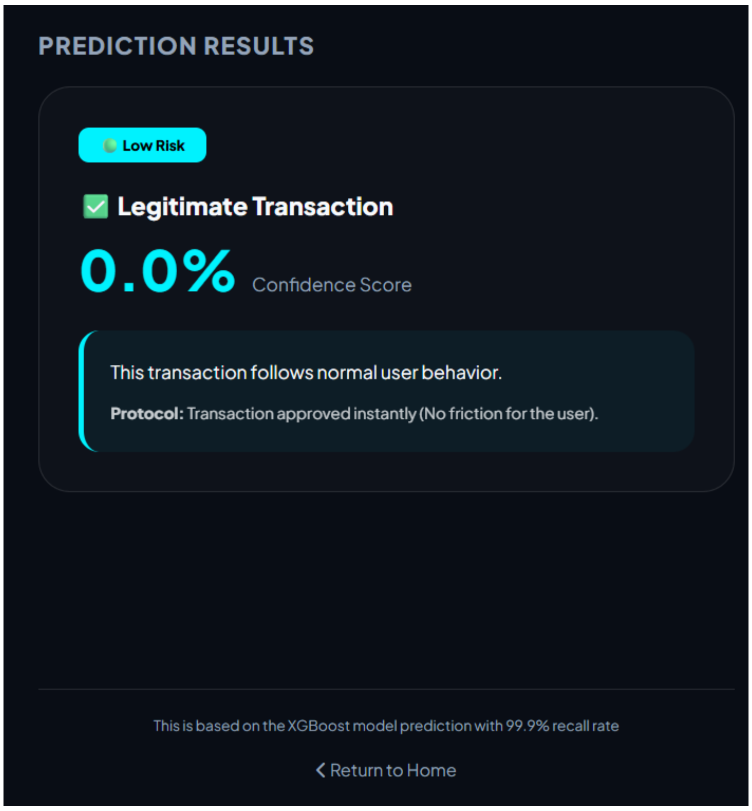
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the Run predict button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.



Online Payment Fraud Detection

Based on XGBoost model with 99.9% recall rate

TRANSACTION TYPE	PAYMENT
STEP	35
AMOUNT	50
ORIGIN OLD BALANCE	500
ORIGIN NEW BALANCE	450
DESTINATION OLD BALANCE	0
DESTINATION NEW BALANCE	0


PREDICTION RESULTS

Low Risk

✅ Legitimate Transaction

0.0% Confidence Score

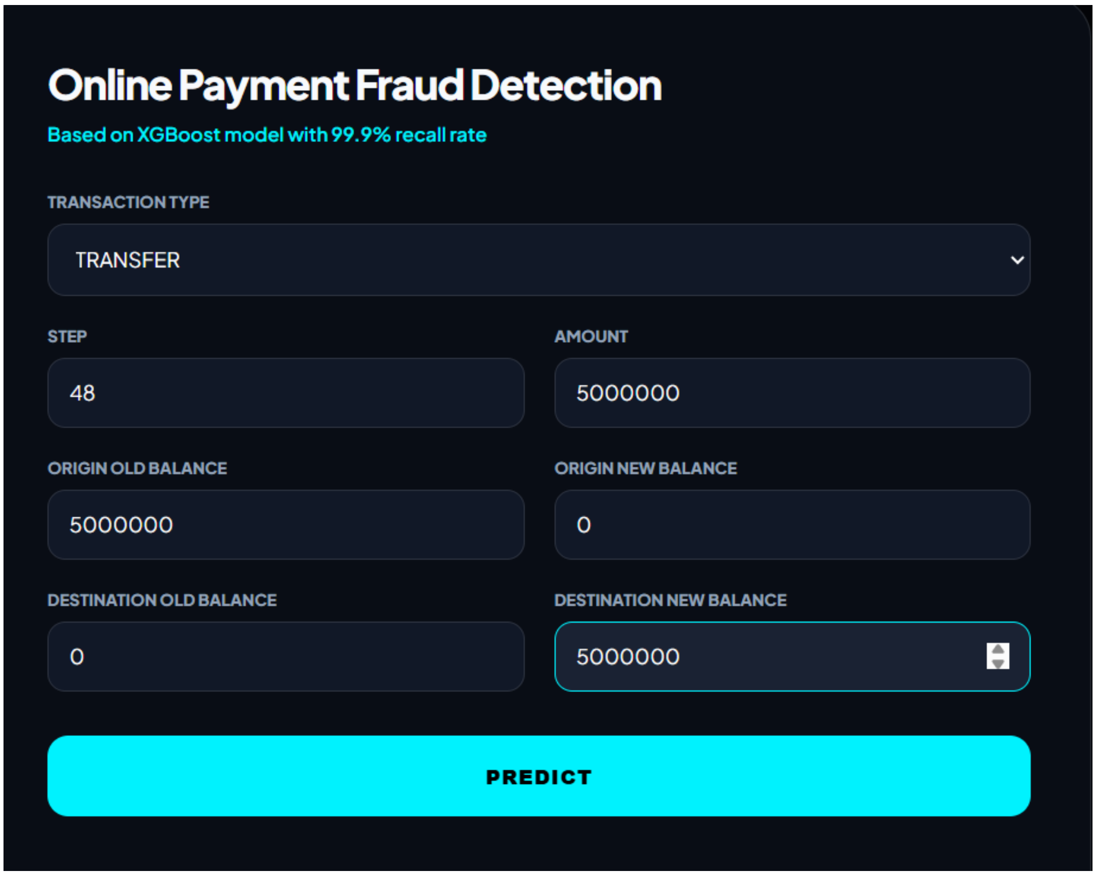
This transaction follows normal user behavior.

Protocol: Transaction approved instantly (No friction for the user).

This is based on the XGBoost model prediction with 99.9% recall rate

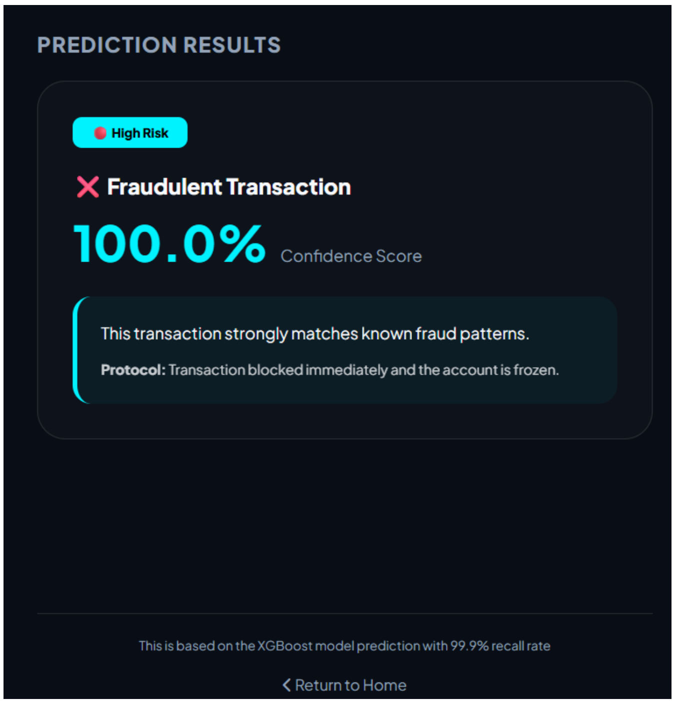
‹ Return to Home

Fig: Legit Transaction


Online Payment Fraud Detection

Based on XGBoost model with 99.9% recall rate

TRANSACTION TYPE	TRANSFER
STEP	48
AMOUNT	5000000
ORIGIN OLD BALANCE	5000000
ORIGIN NEW BALANCE	0
DESTINATION OLD BALANCE	0
DESTINATION NEW BALANCE	5000000



PREDICTION RESULTS

High Risk

Fraudulent Transaction

100.0% Confidence Score

This transaction strongly matches known fraud patterns.

Protocol: Transaction blocked immediately and the account is frozen.

This is based on the XGBoost model prediction with 99.9% recall rate

Fig: Fraud Transaction