

1. Introduction

INTERNSHIP DOCUMENTATION

ON

“Online Payments Fraud Detection using Machine Learning”

Submitted in partial fulfillment of the requirements of the

LongTerm Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

KADIYALA UDAY KIRAN (22BFA02055)

MANCHU PAVANI (22BFA05362)

SHAIK THASNEEM (22BFA05386)

DURGA SRAVANTHI KATHI (22BFA02053)

TEAM ID:

LTVIP2026TMIDS79347

Online Payments Fraud Detection using Machine Learning

2. Project Overview:

Online payment systems have become an essential part of digital transactions in today's economy. However, the rapid growth of digital payments has also led to a significant increase in fraudulent activities. The **Online Payments Fraud Detection using Machine Learning** project aims to develop an intelligent system that proactively identifies and prevents fraudulent transactions in real time.

This project leverages historical transaction data, customer behavioral patterns, and advanced machine learning algorithms to detect anomalies and suspicious activities. By analyzing key transaction attributes such as transaction amount, geographic location, device information, login patterns, and spending behavior, the system can accurately distinguish between legitimate and fraudulent transactions.

2.1 Purpose

The purpose of the **Online Payments Fraud Detection using Machine Learning** project is to enhance the security of digital financial transactions by identifying and preventing fraudulent activities in real time. With the rapid growth of online payment platforms and digital banking services, fraudsters continuously develop new techniques to exploit system vulnerabilities. This project aims to build an intelligent fraud detection system that analyzes transaction data, user behavior patterns, and historical records to accurately detect suspicious activities before financial damage occurs. By leveraging machine learning algorithms, the system ensures secure, reliable, and trustworthy online payment experiences for both users and businesses.

The primary goals of this project are to develop a high-accuracy fraud detection model, minimize false positives, and enable real-time monitoring of online transactions. Additionally, the system aims to identify fraudulent accounts through behavioral analysis and continuously improve its performance by learning from new transaction data. By achieving these goals, the project seeks to reduce financial losses, protect customer data, strengthen cybersecurity measures, and support digital payment platforms in maintaining customer trust and operational efficiency.

2.2 Features

- **Real-Time Transaction Monitoring**
Continuously analyzes online payment transactions as they occur to detect suspicious activities instantly.
- **Machine Learning-Based Detection**
Uses advanced algorithms such as Logistic Regression, Random Forest, and Neural Networks to classify transactions as legitimate or fraudulent.
- **Behavioral Pattern Analysis**
Examines user spending habits, login patterns, transaction frequency, and device usage to identify unusual behavior.
- **Fraudulent Account Identification**

Detects accounts involved in repeated suspicious activities, multiple failed login attempts, or abnormal transaction spikes.

- **Anomaly Detection System**

Identifies unusual deviations from normal transaction patterns using statistical and AI-based anomaly detection techniques.

- **Risk Scoring Mechanism**

Assigns a fraud probability score to each transaction, enabling prioritized investigation and decision-making.

- **Adaptive Learning Capability**

Continuously updates and improves model performance using new transaction data and confirmed fraud cases.

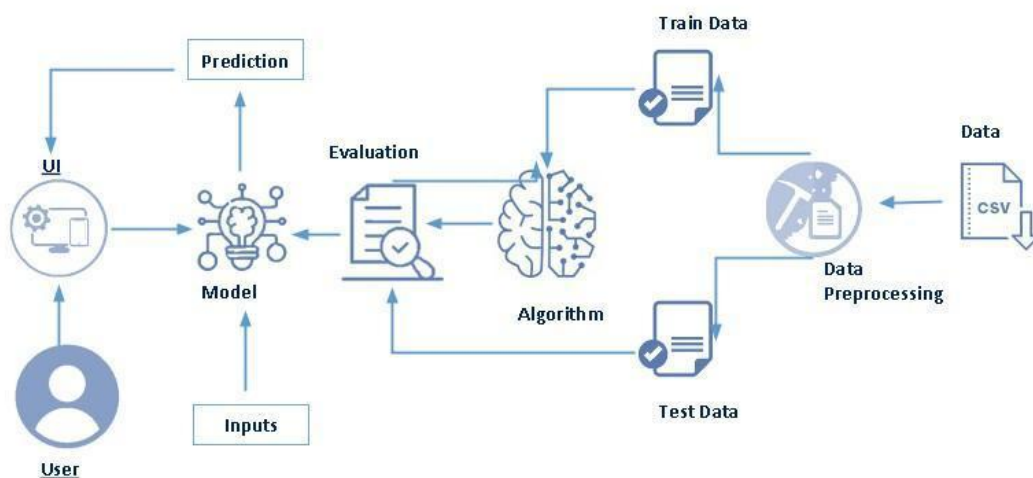
- **Alert and Notification System**

Automatically flags or blocks high-risk transactions and sends alerts to administrators or users.

- **Scalable and Secure Architecture**

Designed to handle large volumes of transactions while ensuring data privacy and security.

3.Architecture:



3.1 Frontend:

The frontend is the user interface where users or administrators interact with the fraud detection system. It allows users to enter transaction details, view prediction results, and The frontend can be developed using:

- **Web Technologies:** HTML, CSS
- **Python-Based UI:** Streamlit / Flask (for ML integration)

It provides:

- Transaction input forms
- Real-time fraud prediction display

3.2 Backend:

The backend handles the core processing and machine learning operations. It receives transaction data from the frontend, preprocesses it, and sends it to the trained machine learning model for fraud prediction.

Backend responsibilities include:

- Data preprocessing (cleaning, encoding, normalization)
- Model training and evaluation
- Real-time prediction handling
- Risk scoring calculation
- API development for frontend communication

Technologies used:

- **Programming Language:** Python
- **Framework:** Flask / FastAPI
- **ML Libraries:** Scikit-learn, Pandas, NumPy

3.3 Database:

The database stores transaction records, user details, and fraud detection results. It also stores historical data used for model training and performance improvement.

Database functions:

- Store transaction history
- Maintain user behavior data
- Store flagged fraud cases

4. Setup Instructions:

4.1 Pre requisites and Installations

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.

4.2 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

4.3 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

4.4 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

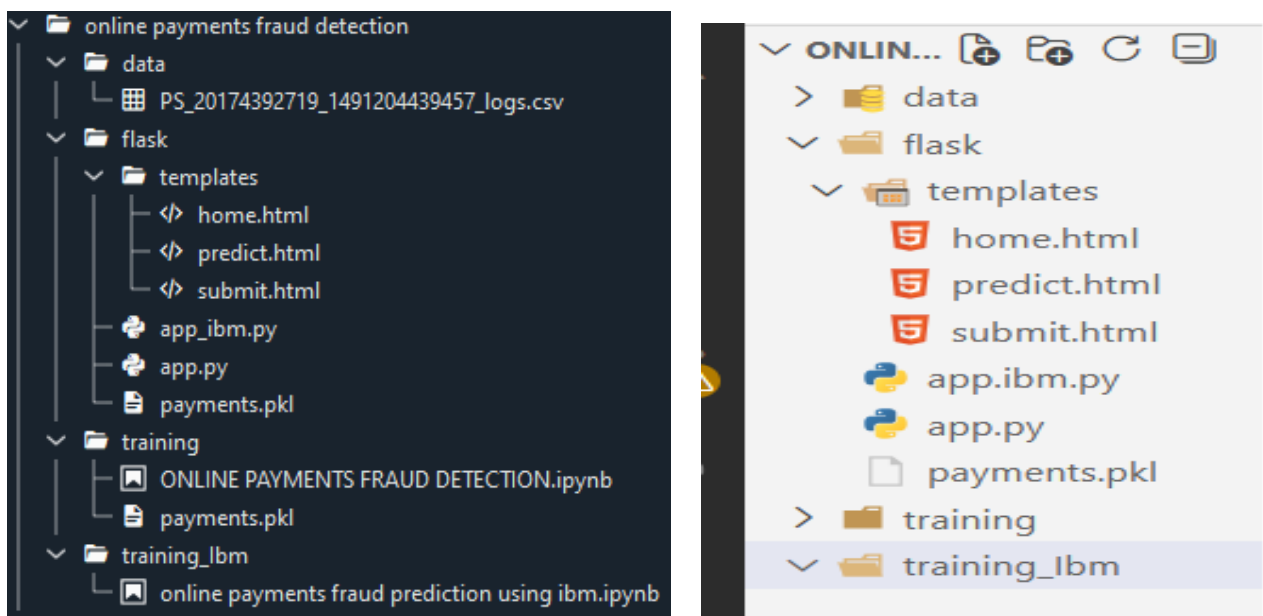
To accomplish this, we have to complete all the activities listed below,

- 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

5. Folder Structure

Create the Project folder which contains files as shown below



We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

- 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.

5.1 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:

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

In this project we have used PS_20174392719_1491204439457_logs.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: [Online Payments Fraud Detection Dataset](#)

5.2 Milestone 2: Visualizing and analyzing data

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

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

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

Importing Libraries¶

```
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 the csv file.

```
# Reading the csv data
df = pd.read_csv(r"C:\Users\user\Desktop\PS_20174392719_1491204439457_logs.csv")
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	IsFraud	IsFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	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
...
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.02	1	0
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.00	1	0
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.59	1	0
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.00	C757947873	0.00	0.00	1	0
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.00	C1827219533	0.00	87999.25	1	0

2430 rows x 11 columns

```
df.columns
```

```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',  
      'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',  
      'isFlaggedFraud'],  
      dtype='object')
```

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

```
df.drop(['isFlaggedFraud'],axis = 1, inplace = True)
```

here, the dataset's superfluous columns are being removed using the drop method.

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.00	0
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.00	0
...
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.02	1
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.00	1
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.59	1
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.00	C757947873	0.00	0.00	1
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.00	C1827219533	0.00	87999.25	1

2430 rows x 10 columns

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

```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0
3	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0	0
4	1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M408069119	0.0	0.0	0

above, the dataset's first five values are loaded using the head method.

```
df.tail()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.0	C79051264	51433.88	108179.02	1
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.0	C1140210295	0.00	0.00	1
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.0	C1759363094	0.00	33676.59	1
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.0	C757947873	0.00	0.00	1
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.0	C1827219533	0.00	87999.25	1

above, the dataset's last five values are loaded using the tail method.

```
plt.style.use('ggplot')
warnings.filterwarnings('ignore')
```

utilising Style use here The Ggplot approach Setting "styles"—basically stylesheets that resemble matplotlibrc files—is a fundamental feature of mpltools. The "ggplot" style, which modifies the style to resemble ggplot, is demonstrated in this dataset.

```
: # checking for correlation
df.corr()
```

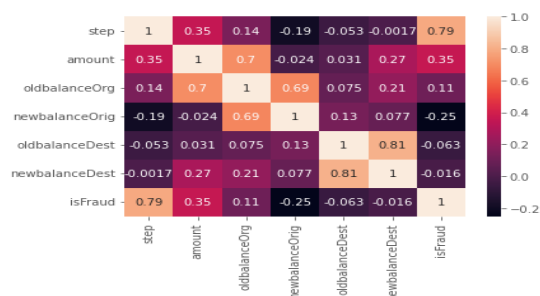
```
:
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	0.352348	0.139868	-0.194391	-0.053366	-0.001745	0.788370
amount	0.352348	1.000000	0.703566	-0.023694	0.030711	0.274788	0.354960
oldbalanceOrg	0.139868	0.703566	1.000000	0.685439	0.075271	0.212087	0.105713
newbalanceOrig	-0.194391	-0.023694	0.685439	1.000000	0.127352	0.077034	-0.250987
oldbalanceDest	-0.053366	0.030711	0.075271	0.127352	1.000000	0.811400	-0.063175
newbalanceDest	-0.001745	0.274788	0.212087	0.077034	0.811400	1.000000	-0.015916
isFraud	0.788370	0.354960	0.105713	-0.250987	-0.063175	-0.015916	1.000000

utilising the corr function to examine the dataset's correlation

Heatmap

```
sns.heatmap(df.corr(),annot=True)
<AxesSubplot:>
```



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

5.3 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

HandlingObject 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)
```

Here, I'm using the shape approach to figure out how big my dataset is

```
df.drop(['nameOrig', 'nameDest'],axis=1,inplace=True)
df.columns
Index(['step', 'type', 'amount', 'oldbalanceOrig', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud'],
      dtype='object')
```

```
df.head()
```

	step	type	amount	oldbalanceOrig	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9.194174	170136.0	160296.36	0.0	0.0	is not Fraud
1	1	PAYMENT	7.530630	21249.0	19384.72	0.0	0.0	is not Fraud
2	1	PAYMENT	9.364617	41554.0	29885.86	0.0	0.0	is not Fraud
3	1	PAYMENT	8.964147	53860.0	46042.29	0.0	0.0	is not Fraud
4	1	PAYMENT	8.868944	183195.0	176087.23	0.0	0.0	is not Fraud

here, the dataset's superfluous columns (nameOrig,nameDest) are being removed using the drop method.

5.4 Milestone : 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.

Algorithms

Random Forest Classifier

Decision Tree Classifier

ExtraTrees Classifier

SupportVectorMachine Classifier

Xgboost Classifier

6. 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

6.1 Building Html Pages:

For this project create three HTML files namely

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

and save them in the templates folder.

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



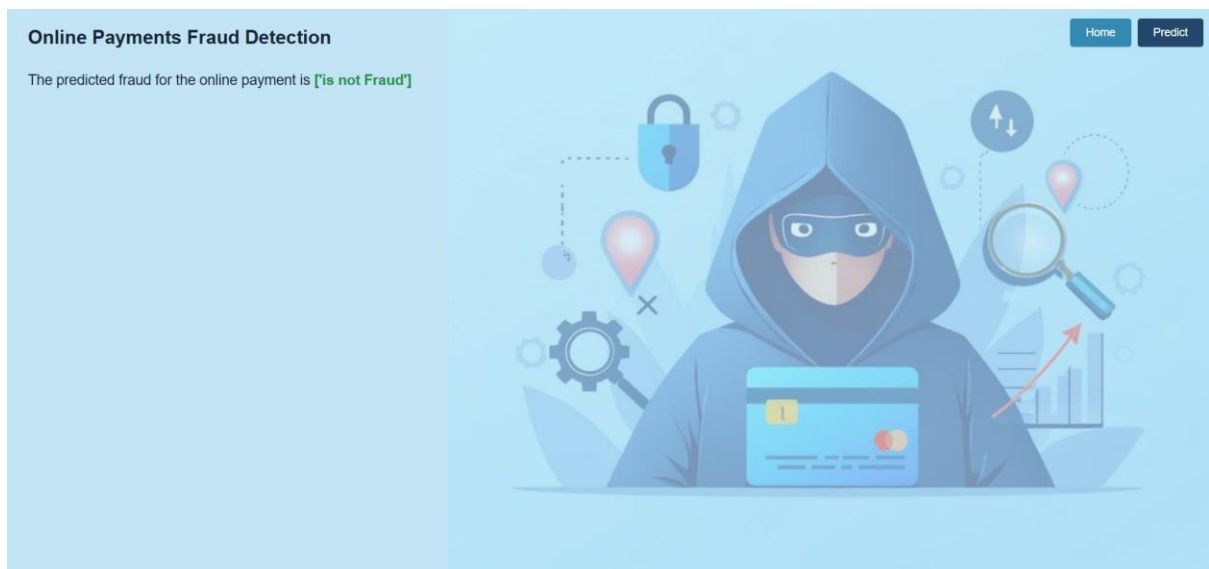
Now when you click on predict button from top right corner you will get redirected to predict.html

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



Now when you click on submit button from left bottom corner you will get redirected to submit.html

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



6.2 Build Python code

Import the libraries

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

model = pickle.load(open(r"C:/Users/user/payments.pkl", 'rb'))
```

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.

```
model = pickle.load(open(r"C:/Users/user/payments.pkl", 'rb'))

app = Flask(__name__)
```

Render HTML page:

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

@app.route("/home")
def about1():
    return render_template('home.html')
```

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

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

Retrieves the value from UI:

```
@app.route("/predict")
def home1():
    return render_template('predict.html')

@app.route("/pred", methods=['POST', 'GET'])
def predict():
    x = [[x for x in request.form.values()]]
    print(x)

    x = np.array(x)
    print(x.shape)

    print(x)
    pred = model.predict(x)
    print(pred[0])
    return render_template('submit.html', prediction_text=str(pred))
```

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

Main Function:

```
if __name__ == "__main__":
    app.run(debug=False)
```

6.3 Running 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 the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
In [11]: runfile('C:/Users/user/Desktop/online payments fraud detection/flask/app.py',
wdir='C:/Users/user/Desktop/online payments fraud detection/flask')
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

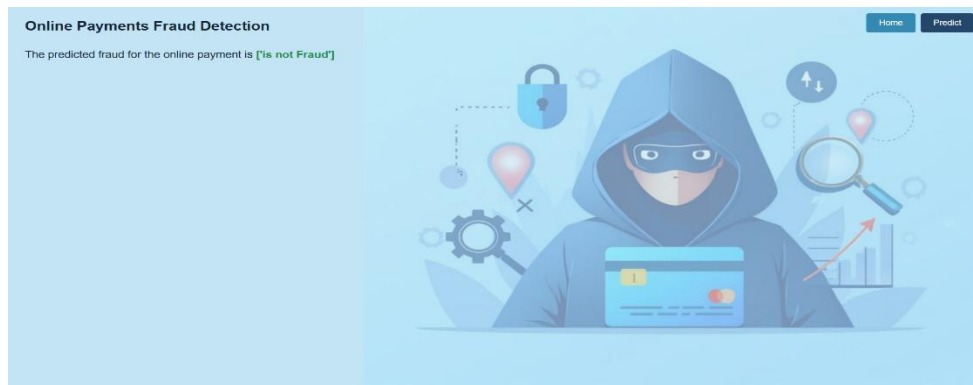
7.User Interface



User Interface Of Home Page

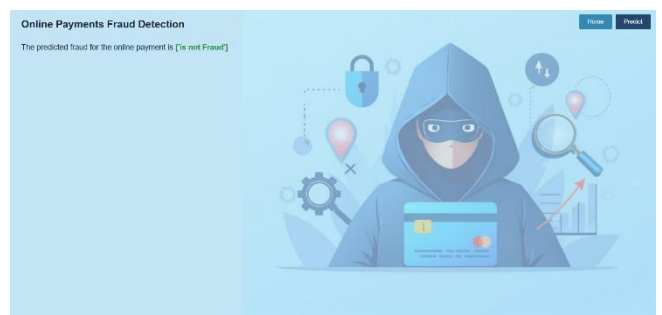
The screenshot shows the prediction page of the web application. It has a sidebar on the left with input fields for various transaction parameters: "Step" (with a note "step: represents a unit of time where 1 step equals 1"), "Type" (type of online transaction), "Amount" (the amount of the transaction), "OldbalanceOrg" (balance before the transaction), "NewbalanceOrig" (balance after the transaction), "OldbalanceDest" (initial balance of recipient before the transaction), and "NewbalanceDest" (the new balance of recipient after the transaction). A green "Predict" button is at the bottom of the sidebar. The main area on the right features the same illustration of a person in a blue hoodie and mask holding a credit card, surrounded by icons of a magnifying glass, a gear, and a bar chart. The background is a light blue gradient.

User Interface Of Prediction Page



User Interface Of Result Page

8.Demostration of application



NOT fraud transaction



Fraud Transaction

Demo video link:

<https://drive.google.com/file/d/1EOoIDf13WdD0VLAnepim0ItkfqKPqevo/view?usp=drivesdk>

9. Known Issues

- **Imbalanced Dataset Problem**
Fraud transactions are usually very rare compared to legitimate ones, which can reduce model accuracy and increase false positives or false negatives.
- **False Positives**
Some legitimate transactions may be incorrectly flagged as fraud, which can inconvenience users.
- **Model Dependency on Historical Data**
The system's performance depends heavily on the quality and quantity of past transaction data.
- **Evolving Fraud Techniques**
Fraud patterns continuously change, and the model may require frequent retraining to remain effective.
- **Real-Time Processing Limitations**
High transaction volumes may slightly affect prediction response time without proper system optimization.
- **Limited Explainability (Basic Version)**
Some machine learning models may not clearly explain why a transaction was classified as fraud.

10. Future Enhancements

- **Deep Learning Models**
Integrate advanced models like Neural Networks or LSTM to improve fraud detection accuracy.
- **Real-Time Streaming Integration**
Use streaming technologies for faster processing of high-volume transactions.
- **Explainable AI (XAI)**
Provide clear explanations for why a transaction was flagged as fraud.
- **Multi-Factor Authentication (MFA)**
Add OTP or biometric verification for high-risk transactions.
- **Cloud Deployment**
Deploy the system on scalable platforms like Amazon Web Services or Microsoft Azure for better performance and availability.