1. Introduction

Project Title:

Online Payments Fraud Detection System

Team Members:

- **Team Leader: Nalluri Yashwanth Chowdary** – Responsible for overall project planning, system architecture design, ML model integration, documentation preparation, and final deployment supervision.
- **Team Member: Nayudu Bhargavi** - Responsible for Machine Learning model development, dataset preprocessing, feature engineering, model training, evaluation, and performance tuning.
- **Team Member: Nayudu Sarat Sai Kumar** – Responsible for Backend development using Flask, API routing, model integration with web application, and prediction logic implementation.
- **Team Member: Polavarapu Srinivasu** – Responsible for Frontend development using HTML, CSS, UI design, form handling, and ensuring smooth communication between frontend and backend.

Online payment systems are widely used in modern digital transactions. However, fraudulent transactions have become a major concern for financial institutions and online platforms. Fraudulent activities cause financial losses, damage customer trust, and affect business reputation.

This project introduces a **Machine Learning-based Fraud Detection System** that predicts whether an online transaction is fraudulent or legitimate using historical transaction data. The system integrates a **Decision Tree Classifier** with a Flask-based web application to provide real-time fraud detection.

The project demonstrates practical implementation of Machine Learning concepts such as data preprocessing, feature selection, model training, evaluation, and deployment.

2. Project Overview

• Purpose:

The primary objective of this project is to design and develop an intelligent fraud detection system capable of identifying suspicious online payment transactions using Machine Learning techniques. With the rapid growth of digital payment platforms, the volume of financial transactions has increased significantly. As a result, the risk of fraudulent activities such as unauthorized transfers, fake transactions, and account manipulation has also grown.

Traditional fraud detection methods rely heavily on manual monitoring and rule-based systems. These approaches are:

- Time-consuming
- Error-prone
- Difficult to scale
- Inefficient for large-scale transaction processing

Manual inspection becomes impractical when millions of transactions occur every day. Human monitoring cannot analyze complex patterns across large datasets in real time. Furthermore, fraudsters continuously evolve their techniques, making static rule-based systems less effective over time.

In addition to solving a real-world financial problem, this project also serves as an academic implementation of core Machine Learning concepts such as data preprocessing, feature selection, model training, testing, evaluation, and deployment. It demonstrates how a trained ML model can be integrated with a Flask web application to create a complete end-to-end system.

• Features:

The Online Payments Fraud Detection System is designed to provide an efficient and intelligent solution for identifying fraudulent transactions. It includes a web-based transaction input form that allows users to enter key transaction details such as transaction type, amount, original balance, and new balance through a simple and structured interface. The design ensures ease of use, enabling both technical and non-technical users to interact with the system effortlessly. The clean layout and organized form fields enhance user experience and reduce input errors.

1. Real-Time Fraud Prediction

Users can input transaction details through a web-based form. The system instantly predicts whether the transaction is Fraud or Not Fraud.

2. Multiple Machine Learning Models

The system compares four algorithms:

- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- Extra Trees Classifier

3. Comprehensive Data Analysis

The project includes:

- Univariate Analysis
- Bivariate Analysis
- Correlation Analysis
- Descriptive Statistics

4. Hyperparameter Tuning

Grid Search Cross-Validation is used to optimize model parameters and improve performance.

5. Production-Ready Deployment

The trained model is saved as payments.pkl and integrated into a Flask web application for real-time prediction.

6. Modular and Scalable Architecture

The project structure separates training and deployment components, allowing easy upgrades and scalability.

3. Architecture

• Frontend:

The frontend of the Online Payments Fraud Detection System is developed using HTML, CSS, and the Flask Template Engine to provide a clean, structured, and user-friendly interface. It consists of essential input components such as a transaction type dropdown menu, amount input field, old balance input, new balance input, a submit button, and a section to display the prediction result.

Developed using:

- HTML
- CSS
- Flask Template Engine

Frontend Components:

- Step (time step input)
- Transaction Type dropdown
- Amount input field
- OldbalanceOrg input
- NewbalanceOrig input
- OldbalanceDest input
- NewbalanceDest input
- Submit button
- Prediction result display

Backend:

The backend of the Online Payments Fraud Detection System is developed using Python with the Flask framework, along with supporting libraries such as NumPy and Scikit-learn. It serves as the core processing unit of the application, handling all server-side operations and machine learning integration. When a user submits transaction details through the web interface, the backend receives the data via a POST request and performs necessary preprocessing steps. This includes converting categorical transaction types into their corresponding numerical values and transforming all inputs into the appropriate data types.

After preprocessing, the backend creates a structured feature array using NumPy in the same format that was used during model training. The pre-trained Decision Tree model, which is loaded once at application startup, is then used to analyze the input data and predict whether the transaction is fraudulent or legitimate. Finally, the prediction result is sent back to the frontend, where it is displayed to the user. This seamless backend workflow ensures accurate, fast, and reliable fraud detection in real time.After training, the model was saved in `.h5` format for reuse during deployment.

Developed using:

- Python
- Flask
- NumPy
- Scikit-learn
- Pandas

Backend Responsibilities:

1. Receive transaction data
2. Convert categorical data into numeric format
3. Create feature array
4. Load trained model
5. Predict fraud
6. Return result to frontend

Machine Learning Model

The system compares multiple algorithms:

- Decision Tree Classifier
- Random Forest Classifier
- Support Vector Machine (SVM)
- Extra Trees Classifier
- Dataset Size: 6,362,620 transactions
- Train/Test Split: 80% / 20%
- Target Variable: isFraud (0 = Not Fraud, 1 = Fraud)
- Evaluation Metrics: Accuracy, Precision, Recall, F1-score, Confusion Matrix

• Database:

Currently, the system does not use a traditional database.

Instead, it uses:

- CSV dataset for training
- payments.pkl file for saved model

Future enhancement may include:

- Transaction history database
- Fraud analytics storage
- User authentication database

4. Setup Instructions

• Prerequisites

Before running the project, ensure the following software and tools are installed:

- Python 3.x
- Flask
- Pandas
- NumPy
- Matplotlib
- Seaborn
- Jupyter Notebook
- Scikit-learn
- VS Code (Recommended)

• Installation:

Follow these steps to set up and run the project:

- ➢ Clone or download the project folder to your local system.
- ➢ Open the project directory in VS Code or any IDE.
- ➢ Install the required dependencies using the following command:
  - pip install -r requirements.txt
- ➢ Train the model if it is not already trained (using the Jupyter notebook or training script).
- ➢ Run the Flask application using the command:
  - jupyter notebook training/ONLINE_PAYMENTS_FRAUD_DETECTION.ipynb
- ➢ Run the Flask application:
  - cd flask
  - python app.py
- ➢ Once the server starts successfully, open a web browser.
- ➢ Navigate to the following URL:
  - http://127.0.0.1:5000/

- ➢ Enter transaction details in the form and submit to view the fraud prediction result.

5. Folder Structure

The project is organized in a structured manner to clearly separate the dataset, backend logic, machine learning components, and frontend templates. This modular design improves maintainability, readability, scalability, and deployment efficiency.

1. Root Directory Files

- The root directory contains the main folders required for application execution and model training.
- **/data** This folder contains the dataset file PS_20174392719_1491204439457_logs.csv. It includes historical transaction records used for training and testing the fraud detection model.
- **flask/** This folder contains the complete Flask web application.
- **training**/ This folder contains the Jupyter Notebook used for building and training the machine learning model.
- **training_ibm/** This folder contains the IBM-based implementation notebook for fraud prediction.

2. Flask Folder

This directory contains backend application files and frontend templates.

- **app.py**
  This is the main Flask application file. It handles routing, receives user input, preprocesses transaction data, loads the trained model (payments.pkl), performs predictions, and displays the result on the frontend.
- **app_ibm.py**
  This file contains the IBM-based Flask implementation for fraud prediction.
- **payments.pkl**
  This file stores the trained machine learning model in serialized format. It allows the application to use the trained model directly for prediction without retraining.
- **Templates Folder (flask/templates/)**
  This directory contains HTML files rendered by Flask.
- **home.html** – Displays the project homepage and navigation.
- **predict.html** – Contains the transaction input form with fields for transaction type, amount, and account balances.
- **submit.html** – Displays the fraud prediction result dynamically after submission.

3. Training Folder

- ONLINE PAYMENTS FRAUD DETECTION.ipynb

This Jupyter Notebook contains the complete machine learning workflow including data preprocessing, feature engineering, model training, evaluation, and saving the trained model.

- **payments.pkl**
  The trained model generated from the notebook and saved for deployment.

4. Training IBM Folder (training_ibm/)

online payments fraud prediction using ibm.ipynb

This notebook contains the fraud detection implementation developed using IBM tools and environment.

.

6. Machine Learning Implementation

Model Selection: Decision Tree

- Load dataset
- Remove unnecessary columns (nameOrig, nameDest)
- Check for null values
- Perform univariate and bivariate analysis
- Encode categorical variables
- Split dataset (80/20)
- Train multiple ML models
- Compare accuracy
- Apply Grid Search CV
- Save best model using pickle

```
pickle.dump(model, open("payments.pkl", "wb"))
```

7. Fraud Detection Workflow

User Input →Form Submission →Flask Backend →Data Transformation →Feature Array Creation →Model Prediction →Result Display

1. [[4, 5000, 10000, 5000]]

   Prediction Output: Fraud or No Fraud

8. Setup Instructions

Prerequisites

- Python 3.x
- Flask

- NumPy
- Scikit-learn
- VS Code (Recommended)

Installation Steps

- Clone project
- Install dependencies:
- ➢ pip install -r requirements.txt
- Train model (if needed)
- Run Flask app:
- ➢ python app.py
- Open browser:
- ➢ http://127.0.0.1:5000/

9.API Documentation

The Online Payments Fraud Detection System provides a RESTful API endpoint to handle transaction prediction requests. The main endpoint used for fraud detection is **POST /predict**, which processes transaction details submitted from the frontend form.

The request method used is **POST**, ensuring secure transmission of user-submitted data. The content type is **form-data**, as the transaction details are sent through an HTML form. The API accepts four key parameters: **type** (transaction type), **amount** (transaction amount), **oldbalanceOrg** (account balance before the transaction), and **newbalanceOrg** (account balance after the transaction). These parameters are required to perform fraud analysis based on the trained machine learning model.

Endpoint

POST /predict

Request Format

- Method: POST
- Content Type: form-data
- Parameters:
    - type
    - amount
    - oldbalanceOrg
    - newbalanceOrg
    - oldbalanceDest
    - newbalanceDest

Processing Steps

1. The backend receives transaction data through a POST request.
2. It validates that all required inputs are provided.

3. The categorical transaction type is converted into its corresponding numeric value.
4. The input values are converted into numeric format (float where required).
5. The trained Decision Tree model predicts whether the transaction is Fraud or No Fraud.
6. The prediction result is returned to the frontend and displayed to the user.

Error Handling

- Missing input
- Invalid numeric values
- Incorrect transaction type
- Internal server errors

10. Testing

Testing includes validating model accuracy using training and validation datasets. Metrics such as accuracy and loss are monitored. Functional testing ensures image upload and prediction features work correctly under different scenarios.

Model Testing

Model performance is evaluated using **training accuracy** and **testing accuracy** to measure how well the model learns from data and generalizes to new transactions. A **confusion matrix** is used to analyze prediction results, including correctly and incorrectly classified fraud and non-fraud transactions. These metrics help in understanding the reliability and effectiveness of the fraud detection model.

During training, the following metrics are monitored:

- Training Accuracy
- Testing Accuracy
- Confusion Matrix

Functional Testing

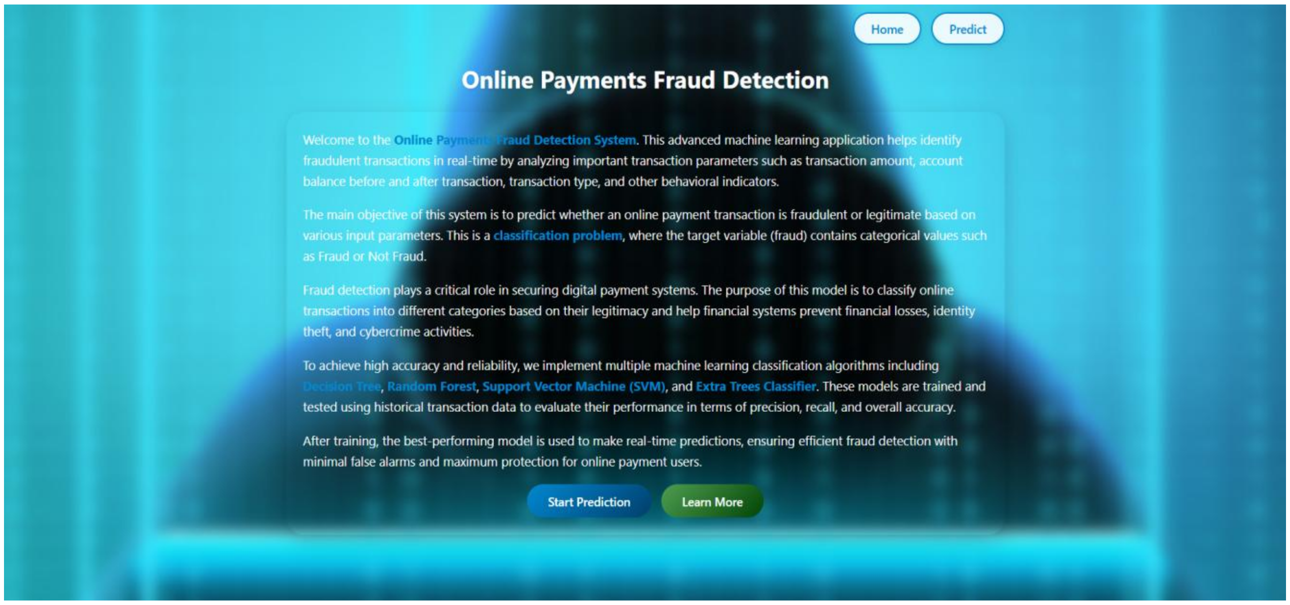
Functional testing ensures that the web application operates smoothly. It verifies proper **form submission**, correct **input validation**, accurate **prediction output**, and overall **server stability**. Error handling is also tested to ensure the system responds appropriately to invalid or missing inputs.

Functional testing ensures that:

- Form submission

- Input validation
- Correct prediction output
- Server stability

11. Screenshots or Demo



Online Payments Fraud Detection

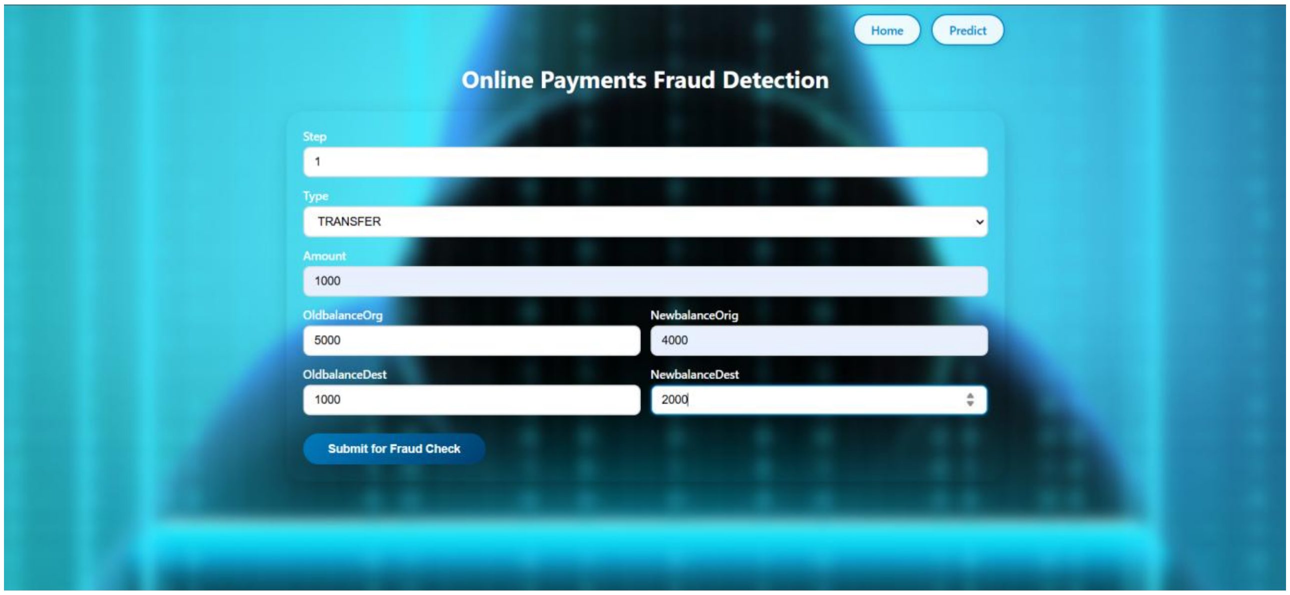
Welcome to the **Online Payments Fraud Detection System**. This advanced machine learning application helps identify fraudulent transactions in real-time by analyzing important transaction parameters such as transaction amount, account balance before and after transaction, transaction type, and other behavioral indicators.

The main objective of this system is to predict whether an online payment transaction is fraudulent or legitimate based on various input parameters. This is a **classification problem**, where the target variable (fraud) contains categorical values such as Fraud or Not Fraud.

Fraud detection plays a critical role in securing digital payment systems. The purpose of this model is to classify online transactions into different categories based on their legitimacy and help financial systems prevent financial losses, identity theft, and cybercrime activities.

To achieve high accuracy and reliability, we implement multiple machine learning classification algorithms including **Decision Tree**, **Random Forest**, **Support Vector Machine (SVM)**, and **Extra Trees Classifier**. These models are trained and tested using historical transaction data to evaluate their performance in terms of precision, recall, and overall accuracy.

After training, the best-performing model is used to make real-time predictions, ensuring efficient fraud detection with minimal false alarms and maximum protection for online payment users.



Online Payments Fraud Detection

Step: 1

Type: TRANSFER

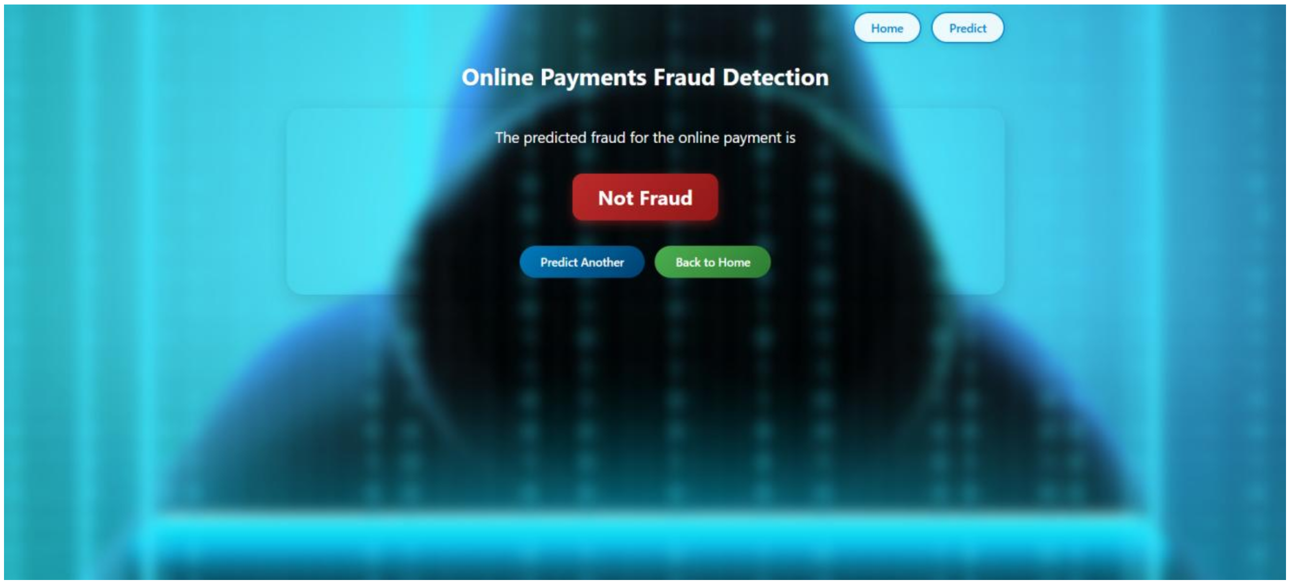
Amount: 1000

OldbalanceOrg: 5000

NewbalanceOrig: 4000

OldbalanceDest: 1000

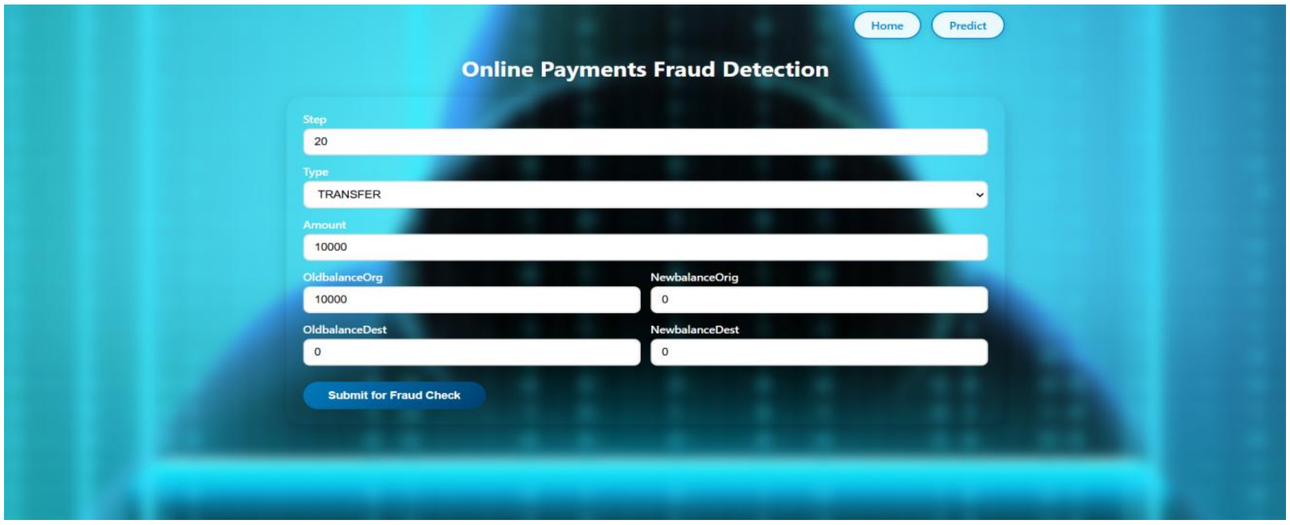
NewbalanceDest: 2000



Online Payments Fraud Detection

The predicted fraud for the online payment is

Not Fraud



Online Payments Fraud Detection

Step: 20

Type: TRANSFER

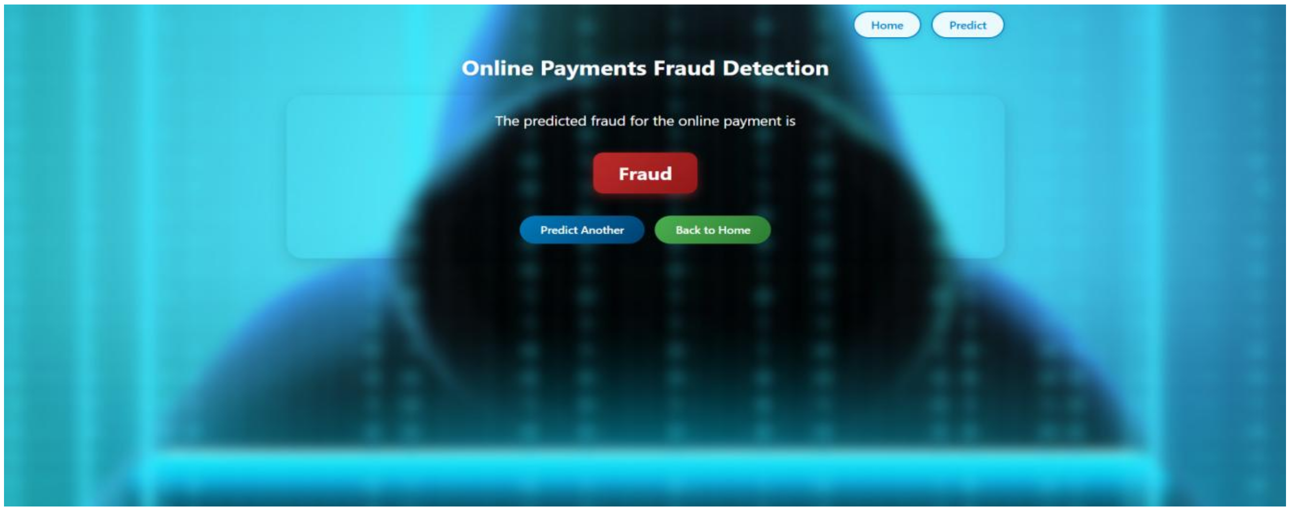
Amount: 10000

OldbalanceOrg: 10000

NewbalanceOrig: 0

OldbalanceDest: 0

NewbalanceDest: 0



Online Payments Fraud Detection

The predicted fraud for the online payment is

Fraud

12.Known Issues

1. Model accuracy depends on dataset quality
2. Imbalanced dataset may affect fraud detection
3. Limited features used
4. No real-time bank integration
5. No authentication implemented
6. The model may produce false positives, marking legitimate transactions as fraud.
7. The system does not currently provide fraud probability scores, only classification results.
8. Performance may degrade with extremely large transaction volumes.
9. The application is designed for academic demonstration and not optimized for enterprise-scale deployment.
10. Lack of continuous learning mechanism to update the model automatically with new fraud patterns.

13.Future Enhancements

1. Use advanced algorithms (Random Forest, XGBoost)
2. Handle imbalanced data using SMOTE
3. Add authentication system
4. Deploy on cloud (AWS/Azure/GCP)
5. Add database for transaction logs
6. Real-time streaming fraud detection
7. Improve UI/UX
8. Integrate email or SMS alerts for detected fraudulent transactions.
9. Develop a continuous model retraining pipeline for adaptive fraud detection.
10. Add role-based access control for enterprise usage.
11. Implement monitoring and logging systems for system performance tracking.