ONLINE PAYMENTS FRAUD DETECTION SYSTEM USING (XGBOOST)

Project Documentation

1. Introduction

1.1 Project Title

Online Payments Fraud Detection System Using Machine Learning (XGBoost Algorithm)

1.2 Team Members

S.No	Name	Role
1	Annem Manasa	Team Leader / Model Development
2	Balagani Rishika	Data Preprocessing & Feature Engineering
3	Dhulipalla Mrudula	Model Training & Evaluation
4	Garikapati Tulasi	Testing, Deployment & Documentation

2. Project Overview

2.1 Purpose

The purpose of this project is to develop an intelligent fraud detection system using the XGBoost machine learning algorithm to identify fraudulent online payment transactions. The system aims to improve financial security by accurately detecting fraud patterns and minimizing false predictions.

2.2 Objectives

- Analyze transaction data to detect fraudulent activities.
- Implement XGBoost algorithm for classification.
- Improve fraud detection accuracy.
- Minimize false positives and false negatives.
- Deploy the trained model for real-time prediction.

3. Architecture

3.1. Overview

The Online Payments Fraud Detection System follows a Machine Learning-based architecture where transaction data flows through multiple stages including preprocessing, model prediction, and result generation.

The architecture is designed to ensure fast prediction, scalability, and reliability for real-time fraud detection.

3.2. Architecture Components

The system consists of the following major components:

1. User Interface (Frontend Layer)

- Accepts transaction details from the user.
- Fields include transaction type, amount, old balance, new balance, etc.
- Sends input data to the backend server for processing.

2. Backend Server (Flask Application Layer)

- Handles HTTP requests.
- Receives transaction data from frontend.
- Performs data preprocessing (encoding, scaling if required).
- Loads the trained XGBoost model file.
- Sends processed data to the ML model for prediction.

3. Data Preprocessing Module

This module performs:

- Handling missing values
- Encoding categorical variables
- Feature selection
- Data formatting to match trained model structure

Proper preprocessing ensures accurate predictions.

4. Machine Learning Model (XGBoost Classifier)

- Trained using historical transaction dataset.

- Stored as a serialized model file (.json or .pkl).
- Loaded during application runtime.
- Predicts whether transaction is:
    - **0 → Genuine**
    - **1 → Fraudulent**

XGBoost uses gradient boosting to improve prediction accuracy and handle imbalanced datasets effectively.

5. Prediction Output Layer

- Displays fraud prediction result to user.
- Shows result as:
    - "Fraud Detected"
    - "Transaction is Safe"
- Response time is within a few seconds, making it suitable for real-time systems.

4. Setup Instructions

1. System Requirements

Hardware Requirements

- Minimum 4 GB RAM
- 10 GB free disk space
- Intel i3 processor or above

Software Requirements

- Operating System: Windows / Linux / macOS
- Python 3.8 or above
- VS Code / Any Python IDE
- Web Browser (Chrome / Edge)

2. Required Python Libraries

Install the following libraries:

- Flask
- Pandas
- NumPy
- XGBoost
- Scikit-learn
- Matplotlib

3. Installation Steps

Step 1: Install Python

Download and install Python from the official website: https://www.python.org/downloads/

Verify installation: python –version

Step 2: Create Virtual Environment (Recommended)

Open terminal inside project folder and run: python -m venv venv

Step 3: Install Required Packages

Install dependencies using pip:

```
pip install flask pandas numpy  xgboost scikit-learn matplotlib
```

4. Project Structure

Your project folder should look like this:

```
Online-Payments-Fraud-Detection/
│
├── app.py
├── xgb_fraud_detection_model.json
├── templates/
│     └── index.html
```

```
├── static/
│   └── styles.css
├── dataset.csv
└── requirements.txt
```

5. Running the Application

Navigate to the project directory and run: python app.py

If successful, you will see:

Running on http://127.0.0.1:5000/

Open your browser and visit:

http://127.0.0.1:5000

5. Folder Structure

1. app.py

- Main Flask application file
- Loads trained XGBoost model
- Handles user input
- Sends prediction results

2. model_training.py

- Used to train XGBoost model
- Includes:
    - Data preprocessing
    - Train-test split
    - Model evaluation
    - Saving trained model

3. xgb_fraud_detection_model.json

- Serialized trained XGBoost model
- Loaded during runtime for prediction

4. dataset/ Folder

Contains:

- transactions.csv (training dataset)

5. templates/ Folder

Contains HTML files:

- index.html → Input form for transaction details

Flask automatically searches this folder for HTML templates.

6. static/ Folder

Contains static files:

- CSS files
- Images
- JS files (if any)

7. requirements.txt

Contains list of required Python libraries:

Example:

flask

pandas

numpy

xgboost

scikit-learn

matplotlib

Overview of Project Structure:

```
Fraud-Detection-ML/

 │

 ├──── data/

 │      └──── transactions.csv

 │
```

```
├── notebooks/
│   └── EDA.ipynb
│
├── models/
│   └── xgb_model.json
│
├── src/
│   ├── preprocessing.py
│   ├── train.py
│   ├── predict.py
│
├── app.py
├── requirements.txt
└── README.md
```

6. Running the Application

1. Go to Project Folder

```
cd Online-Payments-Fraud-Detection
```

2. Install Required Libraries

```
pip install -r requirements.txt
```

3. Run the Application

```
python app.py
```

4. Open in Browser

   http://127.0.0.1:5000

5. Stop the Server

CTRL + C

7. API Documentation

1. Overview

The system provides a REST-based API built using the Flask framework.
The API receives transaction details, processes the data using a trained XGBoost model, and returns a fraud prediction result.

The API supports real-time fraud detection.

2. Type of Authentication Implemented

In this project, basic authentication mechanisms can be implemented using:

- Username and Password Authentication
- Session-Based Authentication (Flask Sessions)
- API Key Authentication (for API access)

3. Login-Based Authentication (Web Application)

Process Flow

1. User enters username and password.
2. Credentials are validated in backend.
3. If valid:
   - User session is created.
   - Access to prediction page is granted.
4. If invalid:
   - Access is denied.
   - Error message is displayed.

4. Session Management

- Flask session is used to maintain login state.
- Session is stored securely.
- User must log in before accessing /predict endpoint.

- Logout clears the session.

6. Security Measures

- Input validation before prediction
- Restricted access to model file
- Authentication required before prediction
- Session timeout for inactive users
- Password hashing (if database is used)

7. Importance of Authentication in Fraud Detection Systems

- Protects financial systems
- Prevents misuse of fraud prediction API
- Ensures data privacy
- Enhances system reliability

8. User Interface

1. Overview

The User Interface (UI) provides an interactive platform where users can enter transaction details and receive fraud prediction results.

The UI is designed to be simple, user-friendly, and responsive to ensure smooth interaction and fast prediction display.

2. UI Components

The web interface consists of the following main components:

1. Header Section

- Displays project title
- Shows system name: Online Payments Fraud Detection System

2. Transaction Input Form

The form collects transaction details required for prediction.

Input Fields:

Field Name	**Description**
Transaction Type	Dropdown (CASH_IN, CASH_OUT, DEBIT, PAYMENT, TRANSFER)
Amount	Transaction amount
Old Balance (Sender)	Sender’s balance before transaction
New Balance (Sender)	Sender’s balance after transaction
Old Balance (Receiver)	Receiver’s balance before transaction
New Balance (Receiver)	Receiver’s balance after transaction

3. Predict Button

- Submits the form data to the backend.
- Triggers fraud prediction.
- Sends POST request to /predict.

4. Result Display Section

After prediction, the result is displayed clearly:

- Transaction is Safe
- Fraud Detected

The result is highlighted with appropriate styling for better visibility.

3. UI Workflow

User
↓
Enter Transaction Details
↓
Click Predict
↓
Backend Processes Data
↓
Prediction Result Displayed

4. UI Technologies Used

- HTML
- CSS
- Flask (Jinja Templates)
- Basic JavaScript (optional)

5. UI Features

- Simple and clean layout
- Dropdown selection for transaction type
- Form validation before submission
- Instant prediction display
- Responsive design
- User-friendly interaction

6. Form Validation

Before sending data to backend:

- All fields are required
- Numeric fields accept only numbers
- Transaction type must be selected
- Empty submissions are prevented

This ensures valid input for accurate model prediction.

7. Importance of User Interface in ML System

- Makes complex ML model accessible to non-technical users
- Provides real-time interaction
- Improves usability and user experience
- Enables smooth data input and result visualization

9. Testing

The testing strategy for the Online Payments Fraud Detection System was designed to ensure accuracy, security, reliability, and performance of the complete MERN stack application integrated with the Machine Learning model.

The system was tested in multiple stages:

1. Unit Testing

Individual components such as API routes, controllers, and frontend components were tested separately to verify correct functionality.

2. Integration Testing

Integration testing was performed to ensure proper communication between:

- Frontend (React)
- Backend (Node.js + Express)
- Database (MongoDB)

- Machine Learning prediction module

This ensured that data flows correctly from user input to prediction result storage.

3. Functional Testing

Each feature was tested based on functional requirements:

- User registration and login
- Fraud prediction
- Transaction storage
- Dashboard display

4. Security Testing

Security testing ensured:

- Password hashing using bcrypt
- JWT token authentication
- Protected API routes
- Prevention of unauthorized access

5. Performance Testing

The system was tested for:

- Prediction response time
- Server response under multiple requests
- Database query efficiency

The prediction response time was maintained within a few seconds to provide near real-time fraud detection.

6. User Interface Testing

Manual UI testing ensured:

- Responsive design
- Proper navigation
- Clear error messages
- Smooth user experience

Tools used for testing:

Tool	**Purpose**
Python	Model evaluation

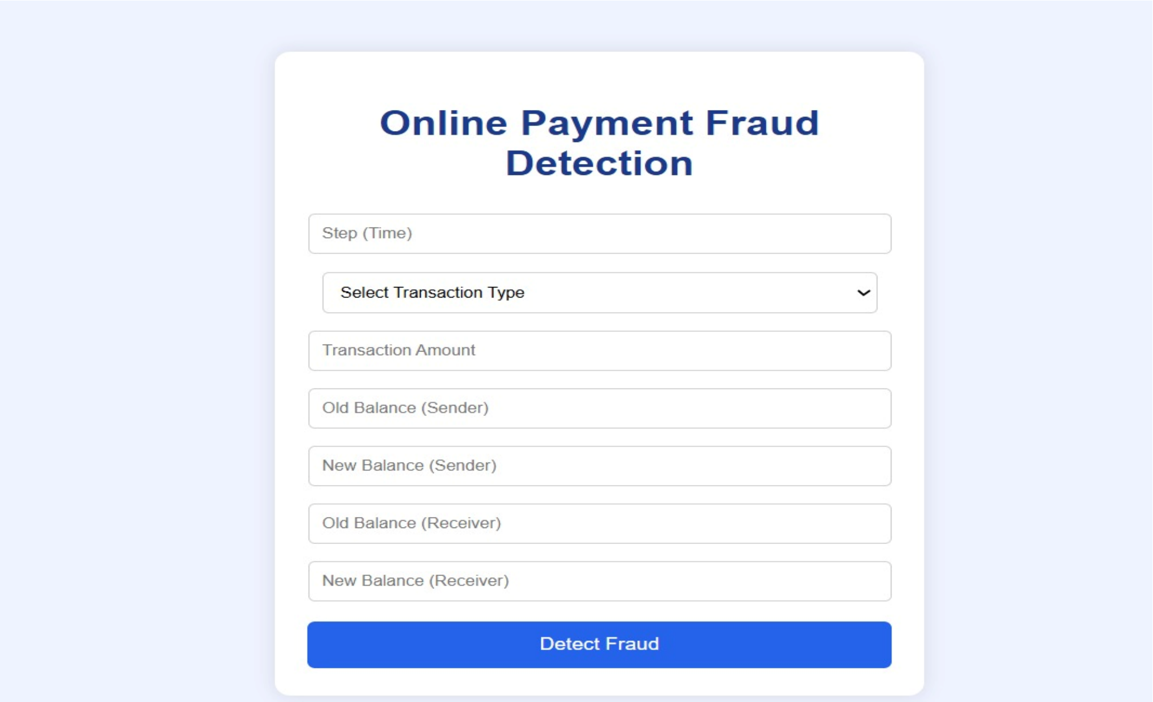
Tool	Purpose
XGBoost	Model training & prediction
Scikit-learn	Performance metrics
Pandas & NumPy	Data handling
Matplotlib	Visualization of confusion matrix
Postman	API testing
Web Browser	UI testing

Testing Outcome

- The XGBoost model achieved high accuracy.
- Fraud detection performance was stable.
- No major runtime errors were found.
- The system provides real-time prediction within a few seconds.

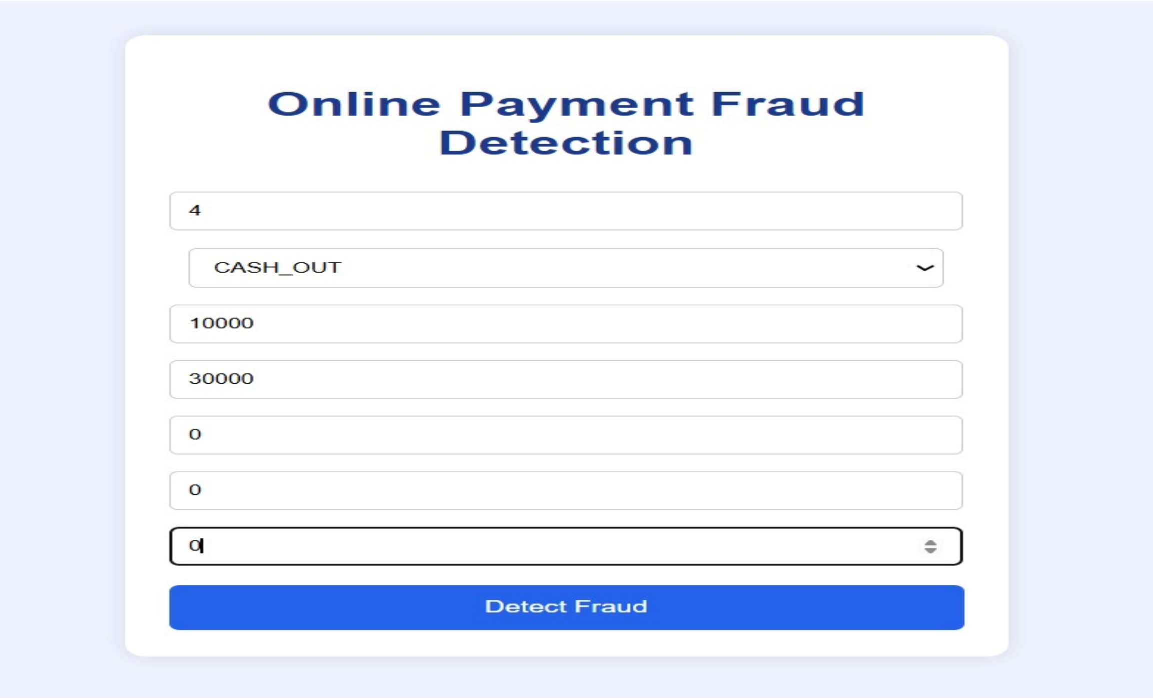
10. Screenshots or Demo

Home Page:



Online Payment Fraud Detection

Transaction Input Form:



Online Payment Fraud Detection

4

CASH_OUT

10000

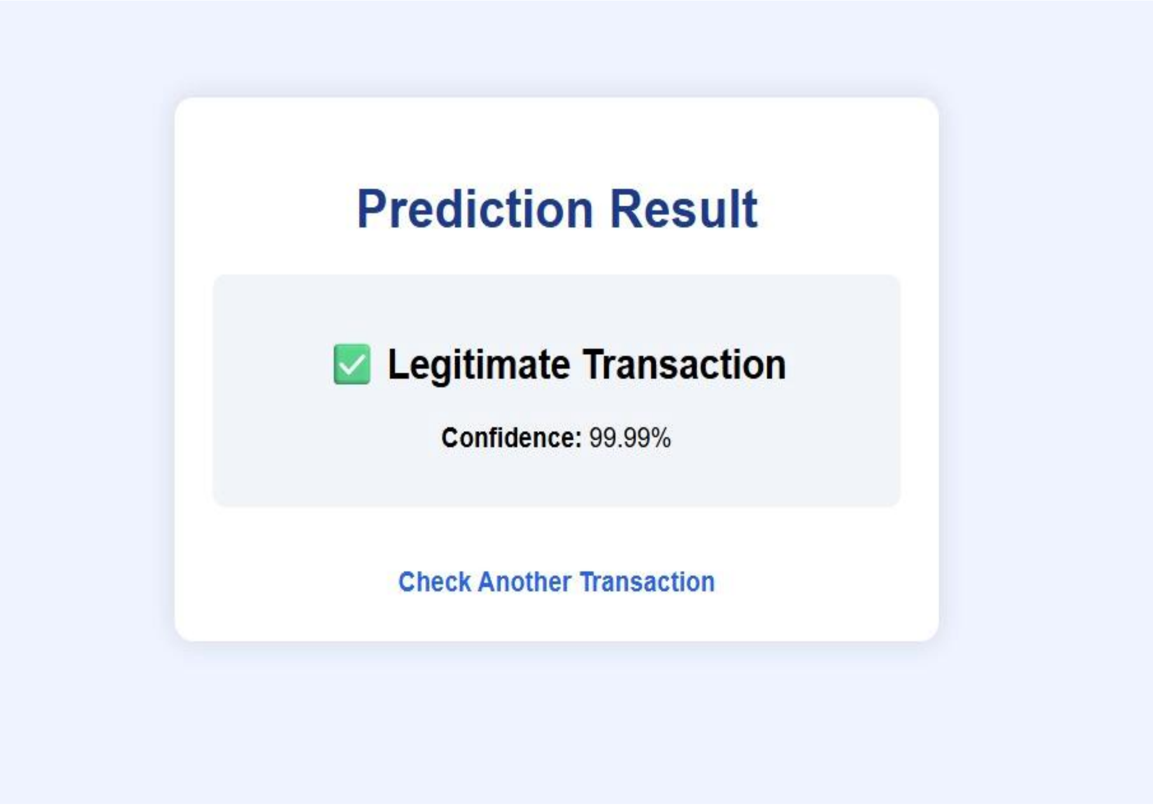
30000

0

0

0

Prediction Results:

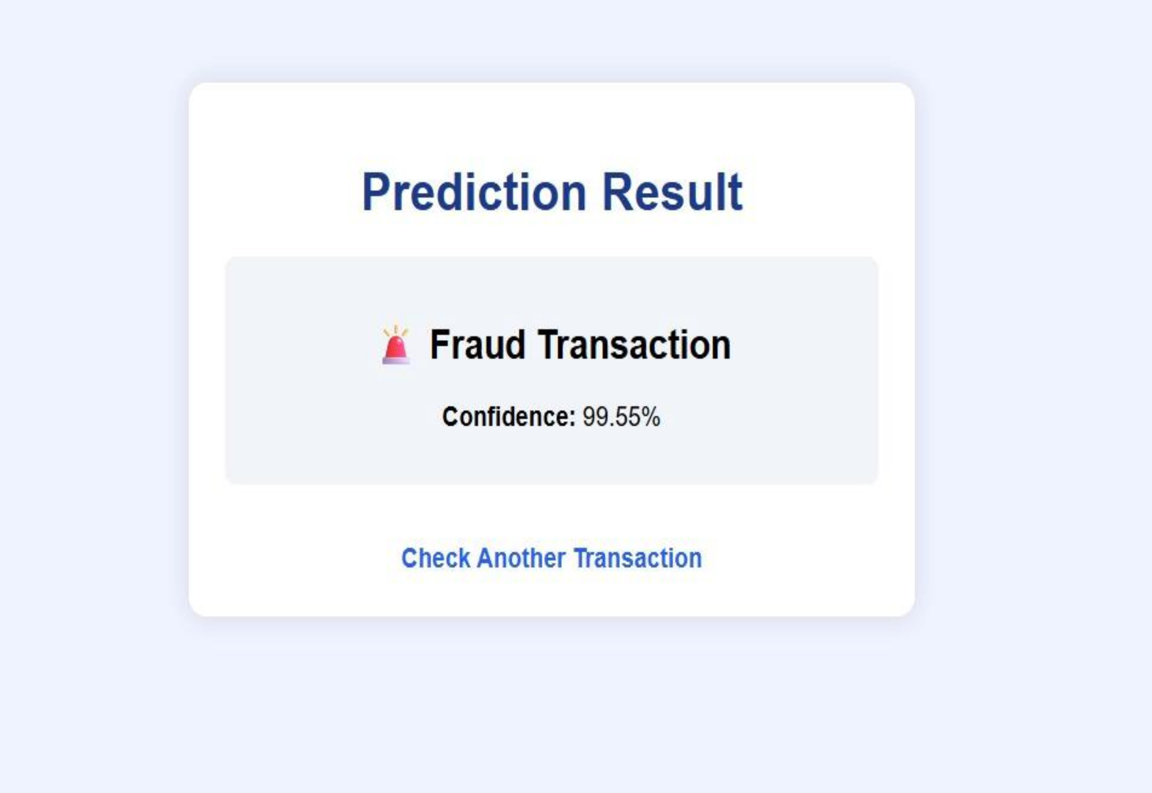


Prediction Result

✅ Legitimate Transaction

Confidence: 99.99%

-



Prediction Result

🚨 Fraud Transaction

Confidence: 99.55%

11. Known Issues

- Model performance depends on training data.
- Requires periodic retraining.
- May generate false positives.
- Heavy traffic may increase response time.
- No real-time banking API integration yet.

12. Future Enhancements

- Integration with real-time banking systems
- Deep Learning-based fraud detection
- Fraud risk scoring dashboard
- Admin panel with analytics
- Email and SMS alert system

- Role-based access control
- Cloud auto-scaling
- Advanced reporting system
- Multi-language support
- AI-based anomaly detection improvements

1. Data Collection
2. Data Preprocessing
3. Feature Engineering
4. Train-Test Split (80% - 20%)
5. Model Training using XGBoost
6. Model Evaluation
7. Model Deployment

5. Machine Learning Model

5.1 XGBoost Algorithm

XGBoost (Extreme Gradient Boosting) is an ensemble learning algorithm based on gradient boosting. It provides high performance, speed, and regularization to prevent overfitting. It is particularly effective for structured and imbalanced datasets.

6. Model Evaluation

- Accuracy
- Precision
- Recall

- F1-Score
- Confusion Matrix

The model achieved high accuracy with balanced precision and recall, ensuring reliable fraud detection performance.

7. Testing Strategy

The testing strategy focused on validating the performance and robustness of the XGBoost model using train-test split validation, cross-validation, and performance evaluation on unseen data.

Tools Used:

- Python
- XGBoost Library
- Scikit-learn
- Pandas & NumPy
- Matplotlib

8. Results

The XGBoost model successfully detected fraudulent transactions with high accuracy. The system supports real-time prediction and demonstrates strong generalization capability.

9. Conclusion

The Online Payments Fraud Detection System using XGBoost effectively demonstrates the practical implementation of machine learning in financial fraud detection. The system is robust, accurate, and suitable for real-world deployment.