Online Payments Fraud Detection Using Machine Learning (Project Report)

1. INTRODUCTION

1.1 Project Overview

Online payment systems are widely used today through UPI, net banking, debit/credit cards, and digital wallets. With the increase in online transactions, fraud activities such as unauthorized transfers, identity theft, and illegal money transfers are also increasing.

This project focuses on building a Machine Learning-based fraud detection system that can classify a transaction as Fraudulent or Legitimate based on transaction details like transaction amount, old balance, new balance, transaction type, etc.

The system uses a trained ML model integrated with a Flask web application where users can enter transaction values and instantly get fraud prediction results.

1.2 Purpose

The purpose of this project is:
- To detect fraudulent online payment transactions.
- To minimize financial losses caused by fraud.
- To build a real-time fraud prediction system using Machine Learning.
- To provide a user-friendly web interface for fraud detection.

2. IDEATION PHASE

2.1 Problem Statement

Fraudulent online transactions are increasing rapidly, causing financial loss and security threats. Traditional fraud detection systems are not efficient in handling large-scale transactions in real time.

Problem:
To build a Machine Learning model that can accurately detect fraudulent transactions based on transaction-related features.

-
-
-
2.2 Empathy Map Canvas

User: Bank / Payment System Admin
 Thinks: Fraud detection should be fast and accurate.
 Feels: Worried about financial loss and customer trust.
 Says: 'We need an automated fraud detection system.'
- Does: Monitors transaction activity and investigates suspicious cases.
- Pain Points: Manual checking takes too much time, fraud cases increase.
- Gains: Automated fraud detection saves time and prevents loss.

2.3 Brainstorming

Ideas discussed:
- Use ML algorithms for prediction.
- Train model using real payment transaction dataset.
- Build web UI for easy transaction input.
- Use Flask to integrate ML model with UI.
- Display result instantly.

Final chosen idea:
Develop a Machine Learning model + Flask web application for fraud prediction.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User opens fraud detection website.
2. User enters transaction details.
3. User clicks 'Predict'.
4. Backend sends input to ML model.
5. ML model predicts fraud or not fraud.
6. Result is displayed on UI.

3.2 Solution Requirement

Functional Requirements:
- User must enter transaction input values.
- System must process input and predict fraud.
- Output should display fraud/not fraud.
- System must load trained model automatically.

Non-Functional Requirements:
- System must be fast and responsive.
- Prediction accuracy should be high.
- UI should be simple and user-friendly.

- System must handle multiple test inputs.

3.3 Data Flow Diagram (DFD)

Level 0 DFD:
User → Web UI → Flask Server → ML Model → Result → User

Level 1 DFD:
1. User enters transaction data.
2. Flask receives data from UI.
3. Data is preprocessed (scaling/encoding).
4. ML model predicts output.
5. Result returned to UI.

3.4 Technology Stack

Frontend: -
HTML
- CSS
- Bootstrap

Backend:
- Python Flask

Machine Learning:
- Pandas
- NumPy
- Scikit-learn

Model Saving:
- Payment (.pkl)

Development Tools: -
VS Code
- Jupyter Notebook

4. PROJECT DESIGN

4.1 Problem Solution Fit

Fraud detection requires analyzing large transaction data and identifying suspicious patterns. Machine Learning is suitable because it can learn fraud patterns from previous data and predict fraud in real time.

-
-
-
4.2 Proposed Solution

The proposed system is a Machine Learning model integrated with a Flask web application.
- Dataset is collected and preprocessed.
  ML model is trained using transaction features.
  Model is saved as a .pkl file.
  Flask loads the model and predicts fraud based on user input.
- UI displays prediction results.

4.3 Solution Architecture

Architecture Flow:
1. User enters transaction details on UI.
2. Flask backend receives input.
3. Input is converted into numeric form.
4. Model predicts fraud or not fraud.
5. Output displayed on webpage.

Components:
- User Interface (HTML form)
- Flask Server
- Preprocessing module
- ML Prediction Model
- Output Display module

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Phase 1: Data Collection
- Dataset collection from online sources.
- Data exploration and understanding.

Phase 2: Data Preprocessing - Handling
missing values.
- Encoding transaction types.
- Feature scaling.

Phase 3: Model Training
- Splitting data into train/test.
- Training using ML algorithm.
- Evaluating accuracy.

Phase 4: Model Deployment -
Saving model using Pickle. -
Creating Flask backend.
- Integrating with HTML UI.

Phase 5: Testing
- Testing predictions with random values.
- Checking output accuracy.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

The system was tested on: -
Fraud transactions
- Legitimate transactions
- Random transaction values

Testing Results:
- The model predicts output quickly.
- UI response time is fast.
- Accuracy depends on dataset and algorithm used.

Performance Metrics used:
- Accuracy Score
- Confusion Matrix
- Precision
- Recall
- F1 Score

7. RESULTS 7.1 Output

Screenshots

The output page shows: -
Entered transaction details -
Prediction result:
Legitimate Transaction OR Fraudulent Transaction

Screenshots should include: -
Home page input form
- Output prediction page
- Flask terminal running

-
-
-

8. ADVANTAGES & DISADVANTAGES Advantages

- Fast fraud detection.
- Automated prediction reduces manual work.
 Helps prevent financial loss.
 Easy-to-use UI.
 Model can be retrained with new data.

Disadvantages

- Accuracy depends on dataset quality.
- Fraud patterns may change over time.
- Imbalanced dataset can reduce fraud prediction accuracy.
- Requires continuous model updating.

9. CONCLUSION

This project successfully implements an Online Payments Fraud Detection system using Machine Learning. The system analyzes transaction data and predicts whether a transaction is fraudulent or legitimate. A trained ML model is integrated into a Flask web application, providing an interactive UI for prediction. This solution helps improve security in online payment systems.

10. FUTURE SCOPE

Future improvements can include:
- Use Deep Learning models for better accuracy.
- Implement real-time fraud monitoring dashboard.
- Add database storage for transaction logs.
- Implement user authentication and admin panel.
- Deploy project online using cloud platforms.
- Improve model performance with feature engineering.

11. APPENDIX

Source Code: -
app.py
- train_model.py
- HTML files for UI

Dataset Link:
Online Payments Fraud Detection Dataset

GitHub Link:

GitHub - Srinu-Namana/Online-Payment-Fraud-Detection-Using-Machine-Learning