PROJECT REPORT

Project Title:

Online Payments Fraud Detection using Machine Learning

Team ID:

LTVIP2026TMIDS80559

Submitted By:

- S Sai Uday Kiran (Team Leader)
- Akula Yunus
- D Shadik
- Bommisetti Sunil
- Boddu MuniKumar

Institution:

Sri Venkateswara College of Engineering, Dept. of IT

Submitted To:

SmartInternz - AI/ML Virtual Internship Program 2026

1. INTRODUCTION

1.1 Project Overview

In today's rapidly growing digital economy, online transactions have become an essential part of everyday financial activities. However, with the increasing volume of online payments, fraudulent activities have also risen significantly. Financial fraud not only causes monetary losses but also affects customer trust and organizational reputation.

The project **"Online Payments Fraud Detection Using Machine Learning"** aims to develop an intelligent system capable of identifying fraudulent transactions in real-time using data-driven techniques. The system analyzes transaction features such as transaction type, amount, account balances, and transaction behavior patterns to determine whether a transaction is legitimate or fraudulent.

To achieve this, a **Random Forest classification algorithm** was implemented due to its robustness, high accuracy, and ability to handle large datasets with multiple features. The model was trained using historical transaction data and integrated into a Flask-based web application that allows users to input transaction details and receive instant fraud prediction results.

1.2 Purpose

The primary goal of this project is to design and implement a machine learning-based fraud detection system capable of automatically classifying transactions as fraudulent or non-fraudulent. It aims to reduce financial losses caused by fraudulent activities, improve the efficiency of transaction monitoring, automate fraud detection without human intervention, and provide real-time predictions through a web-based interface.

Unlike traditional rule-based fraud detection systems that often fail to identify new or evolving fraud patterns, this project leverages machine learning techniques such as Random Forest. By learning from historical transaction data, the system can recognize patterns and detect anomalies more accurately and effectively.

Additionally, the project provides hands-on experience in data preprocessing, exploratory data analysis, model training and evaluation, comparison of machine learning models, Flask-based web application development, and deployment-ready ML system integration. Overall, it demonstrates the practical application of Artificial Intelligence in the financial domain to ensure secure and reliable digital payment systems.

2. IDEATION PHASE

2.1 Problem Statement

With the rapid expansion of digital payment platforms, online transactions have become the dominant method of financial exchange. However, this growth has also resulted in a significant rise in fraudulent activities, including unauthorized transfers, identity theft, and other suspicious transactions that threaten both users and financial institutions.

Traditional fraud detection systems mainly rely on predefined rule-based mechanisms. While effective for known fraud patterns, these systems struggle to detect new and evolving fraud techniques. They often generate high false positives and false negatives, reducing reliability. Moreover, manual transaction monitoring is inefficient and impractical due to the enormous volume of daily digital transactions.

To overcome these limitations, there is a strong need for an intelligent, automated, and scalable fraud detection solution. Such a system should be capable of analyzing transaction patterns in real time, identifying suspicious behavior accurately, minimizing financial losses, and strengthening overall transaction security. This project addresses the problem by implementing a Random Forest-based machine learning model that classifies transactions as fraudulent or legitimate with high accuracy and improved adaptability.

2.2 Empathy Map Canvas

To better understand end-user requirements, an empathy analysis was conducted focusing on key stakeholders such as bank security analysts, financial institutions, digital payment platforms, and fraud investigation teams. A representative user persona, Rahul, a fraud monitoring analyst at a digital payments company, was created to illustrate the typical user; he is responsible for reviewing suspicious transactions daily, identifying potential fraud cases, and taking necessary actions to prevent financial losses.

- **Says** – "We need faster and more accurate fraud detection."
- **Thinks** – "Missing a fraud case could result in huge financial loss."
- **Does** – Reviews transaction logs and flags suspicious activities.
- **Feels** – Stressed due to high transaction volume and evolving fraud techniques.
- **Hears** – Complaints from customers about unauthorized transactions.
- **Sees** – Thousands of transactions occurring every minute.
- **Pains** – Manual workload, delayed detection, false alerts.
- **Gains** – Automated alerts, accurate fraud detection, reduced workload.

The proposed AI-driven system helps reduce manual dependency and improves confidence in fraud detection decisions.

2.3 Brainstorming

During the initial stage of the project, multiple approaches were evaluated to solve the fraud detection problem effectively.

Key Areas Explored:

- **Problem-Solution Mapping**

  Identified the major issue as increasing fraud in online payment systems and the

inefficiency of traditional rule-based systems.

- **Algorithm Evaluation**

  Several classification algorithms were considered:

- Logistic Regression
- Decision Tree
- Random Forest

  After comparison, **Random Forest** was selected due to:

- High accuracy
- Ability to handle imbalanced datasets
- Reduced overfitting
- Better feature importance handling

- **Data Handling Strategy**

  The dataset was analyzed for:

- Missing values
- Class imbalance
- Feature correlation

Necessary preprocessing steps were applied to clean and transform the data.

- **Model Optimization**

  Hyperparameter tuning was performed to improve performance and achieve better generalization.

- **Web Integration Planning**

  A Flask-based web application was chosen to:

- Provide a simple UI
- Accept transaction inputs
- Display fraud prediction results instantly

Through iterative testing and refinement, the system achieved **99.24% test accuracy**, confirming its effectiveness.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The Customer Journey Map represents how users interact with the Online Payments Fraud Detection system from initial access to final decision-making.

User Persona:

Anjali, a fraud monitoring officer at a digital payments company, reviews suspicious transactions daily.

Stage	User Action	Pain Points	Solution via System
1. Awareness	Learns about ML fraud detection tool	Manual review is slow	AI-based automation
2. Access	Opens the fraud detection web app	Complex systems are hard to use	Simple Flask-based UI
3. Input	Enters transaction details	Large amount of data to check	Structured input fields
4. Classification	Waits for prediction	Fear of inaccurate results	Random Forest high accuracy (99.24%)
5. Result Interpretation	Views fraud/legitimate prediction	Unsure about reliability	Clear classification result
6. Decision	Takes action (block/approve)	Risk of financial loss	AI-supported decision-making

3.2 Solution Requirements

Functional Requirements:

1. **Transaction Input Interface**
   The system must allow users to enter transaction details including:
   - Transaction type
   - Amount
   - Old balance
   - New balance
2. **Fraud Classification**
   The Random Forest model must predict whether the transaction is:
   - Fraudulent
   - Non-Fraudulent
3. **Result Display**
   The system must clearly display:
   - Prediction result
   - Fraud status
4. **User-Friendly Interface**
   The system must be easy to operate for non-technical users.

Non-Functional Requirements (Technical):

1. **Performance**
   - Prediction time should be less than 2 seconds
   - Efficient processing of large datasets
2. **Accuracy**
   - Achieved test accuracy of 99.24%
3. **Scalability**
   - Can be extended to real-time banking systems
4. **Reliability**
   - Should handle incorrect inputs gracefully
5. **Security**
   - Input validation must be implemented in Flask

3.3 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) represents how data moves through the Online Fraud Detection — from image upload to prediction output. It highlights key system components and their interactions.

External Entity:

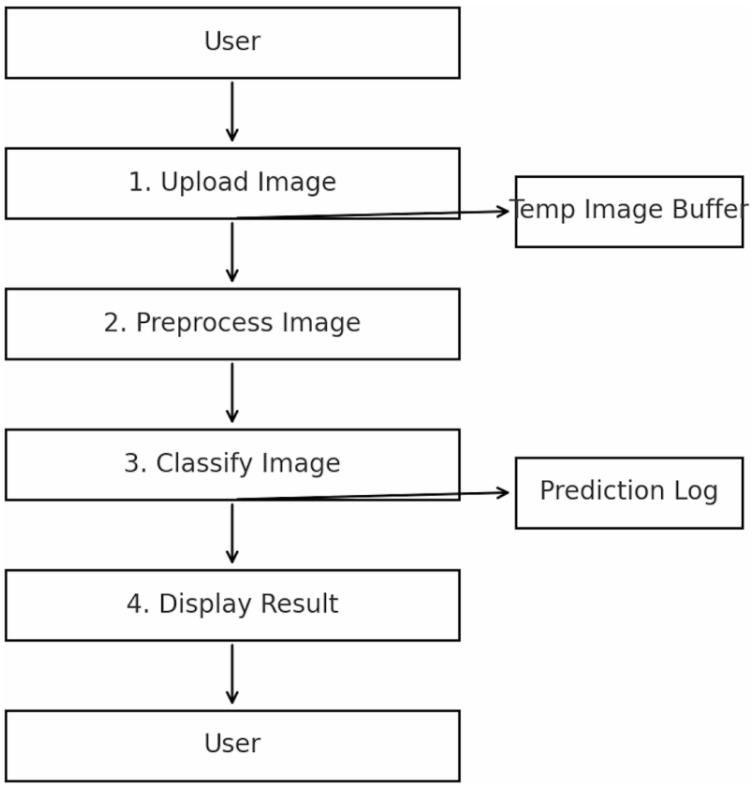
User

Processes:

1. Input Transaction Data
2. Data Preprocessing
3. Random Forest Classification
4. Result Generation

Flow:

User → Web Interface → Flask Backend → ML Model → Prediction → Display Result



3.4 Technology Stack

The project is developed using a well-structured technology stack that supports data analysis, machine learning model development, and web deployment. Each component plays a crucial role in building an efficient and scalable fraud detection system.

Programming Language:

Python is used as the core programming language due to its simplicity, readability, and

extensive support for data science and machine learning libraries. It enables seamless integration between data processing, model development, and web application deployment.

Machine Learning Libraries:

Scikit-learn is utilized to implement the Random Forest algorithm for fraud classification, providing robust tools for model training, testing, and evaluation. Pandas is used for data manipulation and preprocessing, such as handling missing values, feature selection, and data transformation. NumPy supports numerical computations and array operations, ensuring efficient mathematical processing. Matplotlib and Seaborn are used for Exploratory Data Analysis (EDA), helping visualize transaction patterns, correlations, and class distributions to better understand the dataset.

Web Framework:

Flask is used as the backend web framework to integrate the trained machine learning model into a web-based application. It handles routing, user input processing, and real-time prediction functionality, making the system interactive and user-friendly.

Frontend:

HTML and CSS are used to design the user interface of the web application. They provide a clean and simple layout for entering transaction details and displaying prediction results in an understandable format.

Development Tools:

Jupyter Notebook is used for data exploration, preprocessing, and model experimentation. VS Code serves as the primary code editor for application development and integration. Anaconda is used for managing Python environments and dependencies, ensuring smooth installation and execution of required libraries.

Section 4: Project Design

4.1 Problem-Solution Fit

The increasing number of online transactions has significantly raised the risk of fraudulent activities in digital payment systems. Traditional fraud detection systems rely on rule-based mechanisms that are unable to adapt to new fraud patterns. These systems often generate high false positives and fail to detect sophisticated fraud attempts.

The proposed solution applies Machine Learning techniques to analyze historical transaction data and automatically classify transactions as fraudulent or legitimate.

By evaluating multiple classification algorithms including:

- Decision Tree
- Random Forest
- XGBoost

The Random Forest algorithm demonstrated superior performance in terms of accuracy, stability, and generalization capability.

With a test accuracy of **99.24%**, the model effectively identifies fraudulent transactions with high reliability. This confirms that the proposed solution strongly fits the real-world fraud detection problem.

4.2 Proposed Solution

The proposed system is a Machine Learning-based web application that detects fraudulent online transactions using a trained Random Forest model.

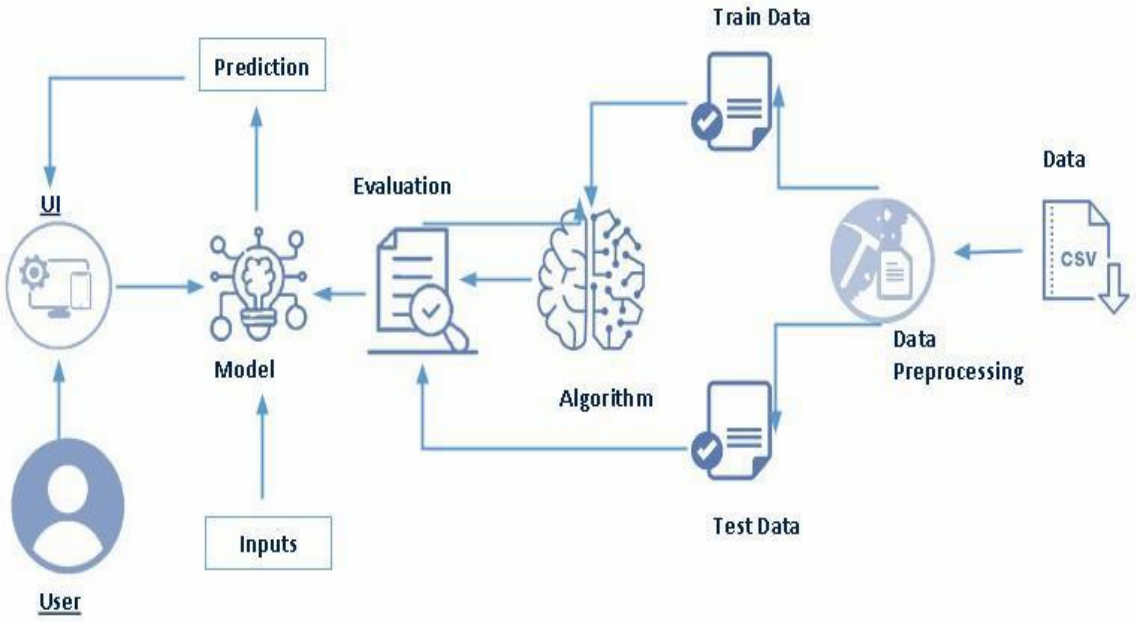
Core Components of the System:

- Data Preprocessing Module
  - Handling categorical variables
  - Feature selection
  - Train-test splitting
- Model Training Module
- Comparison of Decision Tree, Random Forest, and XGBoost
  - Selection of best-performing model
  - Model saving using .pkl format
- Prediction Module
  - Accepts transaction details as input
  - Applies trained Random Forest model
  - Returns fraud prediction result
- Web Application Integration
  - Built using Flask
  - Provides user interface for input and output
  - Displays prediction result instantly

The system is currently deployed locally and designed to be easily extended for cloud deployment.

4.3 Solution Architecture

Below is a high-level view of the Online Payment Fraud Detection architecture:



5 PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The **Online Payments Fraud Detection Using Machine Learning** project was developed as part of the AI/ML Internship program under SmartInternz. The project was completed in a structured and systematic manner using a team-based approach.

The development process was divided into multiple phases including data preprocessing, model building, evaluation, and web application integration. Tasks were distributed among team members based on their strengths to ensure efficient collaboration and timely completion.

Sprint-wise Breakdown with Team Assignments

Week 1 – Dataset Understanding & Preprocessing

- Dataset Analysis and Exploration
  Duration: 2 days
  Assigned to: Team Member 1
  Outcome: Understood dataset structure, features, fraud distribution, and performed exploratory data analysis.
- Data Cleaning & Preprocessing
  Duration: 3 days
  Assigned to: Team Member 2
  Outcome: Removed unnecessary columns, handled missing values, encoded categorical variables, and prepared data for training.

Week 2 – Model Development & Evaluation

- Model Implementation
  Duration: 3 days
  Assigned to: Team Member 1
  Algorithms Implemented:
  - Decision Tree
  - Random Forest
  - XGBoost

  Outcome: Compared model performances.
- Model Selection & Evaluation
  Duration: 2 days
  Assigned to: Team Member 2
  Outcome:
  - Selected Random Forest as final model
  - Achieved 99.24% test accuracy
  - Evaluated using performance metrics

Week 3 – Web Application Development

- Frontend Design (HTML & CSS)
  Duration: 3 days
  Assigned to: Team Member 3
  Outcome: Designed user-friendly transaction input interface.
  - Backend Development using Flask
    Duration: 3 days
    Assigned to: Team Member 2
    Outcome: Integrated trained Random Forest model into Flask application.
  - Integration & Testing
    Duration: 1 day
    Assigned to: All Members
    Outcome: Ensured smooth interaction between UI and ML model.

Week 4 – Testing, Documentation & Deployment

- Functional & Performance Testing
  Duration: 2 days
  Assigned to: All Members
  Outcome: Validated prediction accuracy and system stability.
- Code Optimization & Debugging
  Duration: 1 day
  Assigned to: Team Member 2
  Outcome: Improved performance and removed errors.
- Documentation & Report Preparation
  Duration: 2 days
  Assigned to: Team Member 3
  Outcome: Prepared final project report and presentation materials.

Project Management Tools

- **GitHub –** Version control and collaboration
- **Jupyter Notebook –** Model development and experimentation
- **VS Code –** Backend integration
- **Google Meet / WhatsApp –** Team coordination

6.FUNCTIONAL AND PERFORMANCE TESTING

6.1 Functional Testing

Functional testing was conducted to ensure that each component of the Online Payments Fraud Detection system operates correctly — from transaction input to fraud prediction output.The application was tested under multiple scenarios to validate accuracy, reliability, and user interaction flow.

Test Cases

Test Case ID	Scenario	Input	Expected Output	Status
TC-01	Enter normal transaction details	Valid transaction data	Prediction: Non-Fraud	Passed
TC-02	Enter suspicious transaction details	High amount + abnormal balance	Prediction: Fraud	Passed
TC-03	Leave mandatory fields empty	No input	Error message prompting valid input	Passed
TC-04	Enter invalid data type	Text instead of numeric value	Validation error message	Passed
TC-05	Multiple repeated predictions	Different inputs	Consistent prediction response	Passed
TC-06	Run on different browser	Chrome / Edge	Smooth UI and correct output	Passed

---

6.2 Performance Testing

Performance testing was conducted to evaluate the efficiency and reliability of the fraud detection model.

Evaluation Metrics:

6.2.1 Model Accuracy:

~The Random Forest model achieved **99.24% test accuracy**, demonstrating strong classification capability.

6.2.2 Average Inference Time:

~Average prediction time is approximately **1–2 seconds** on a local CPU-based Flask server.

6.2.3 Model Size:

~No noticeable UI lag or backend crash during multiple prediction cycle

6.2.4 System Responsiveness:

~No UI lag or crash across multiple tests and devices

6.2.5 Generalization Capability:

~The model was tested on unseen test data, confirming its ability to detect fraud patterns effectively.

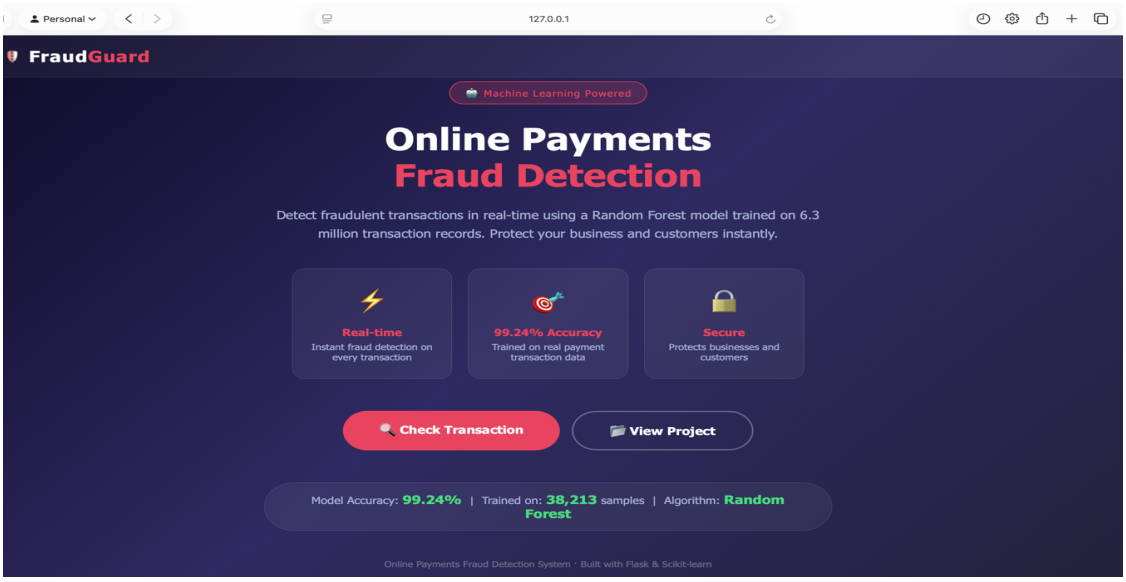
This testing phase confirms that the system is accurate, efficient, and reliable for practical fraud detection scenarios.

7 RESULTS

7.2 Output Screenshots

During the final stage of development, the Online Payments Fraud Detection system was executed locally using a Flask-based web application. The system was tested with multiple transaction inputs to validate fraud detection performance and interface functionality.

• Figure 1: Home Page – Landing Interface



The home page serves as the landing interface of the Online Payments Fraud Detection system.

The page contains:

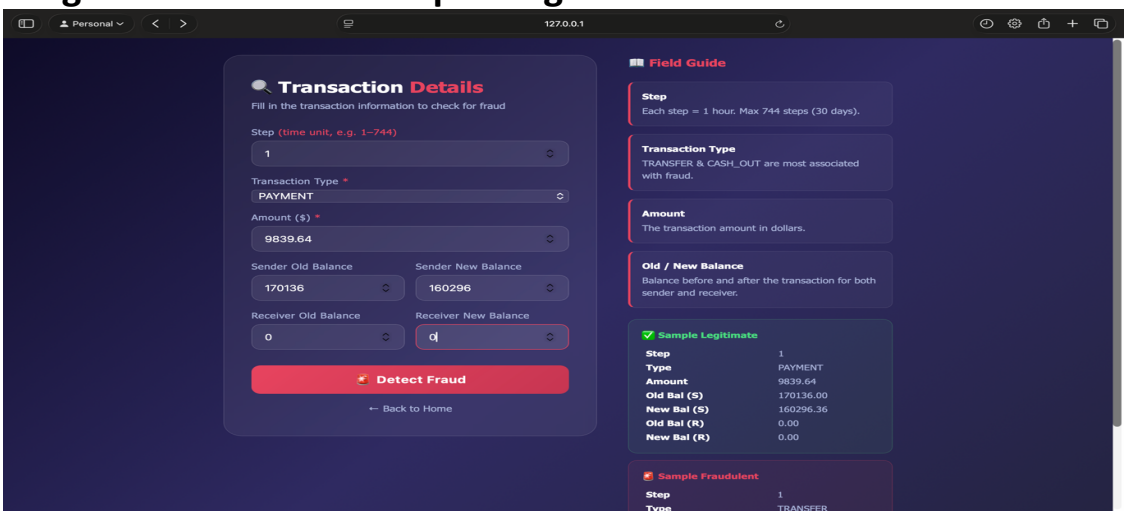
- Project Title: *Online Payments Fraud Detection*
- A brief system description explaining the purpose of the application
- A prominent **"Start Fraud Detection"** button
- Clean gradient background with centered content layout

The description clearly states that the system uses machine learning to analyze transaction behaviour and detect fraudulent activities.

The design emphasizes:

- Simplicity
- Professional appearance
- Clear call-to-action button
- Easy navigation

• Figure 2: Transaction Input Page



After clicking the “Start Fraud Detection” button, users are redirected to the prediction form page.

This page allows users to enter transaction details including:

- Step (Unit of time)
- Transaction Type (CASH_IN, CASH_OUT, TRANSFER, etc.)
- Transaction Amount
- Old Balance of Sender (OldbalanceOrg)
- New Balance of Sender (NewbalanceOrig)
- Old Balance of Receiver (OldbalanceDest)
- New Balance of Receiver (NewbalanceDest)

A **“Check Fraud”** button submits the input to the backend model for prediction.

Design Features:

- Structured form layout
- Clear labels and placeholders
- Dropdown selection for transaction type
- Numeric validation for amount and balances
- Centered card-style interface

The interface is simple and user-friendly, ensuring ease of use even for non-technical users.

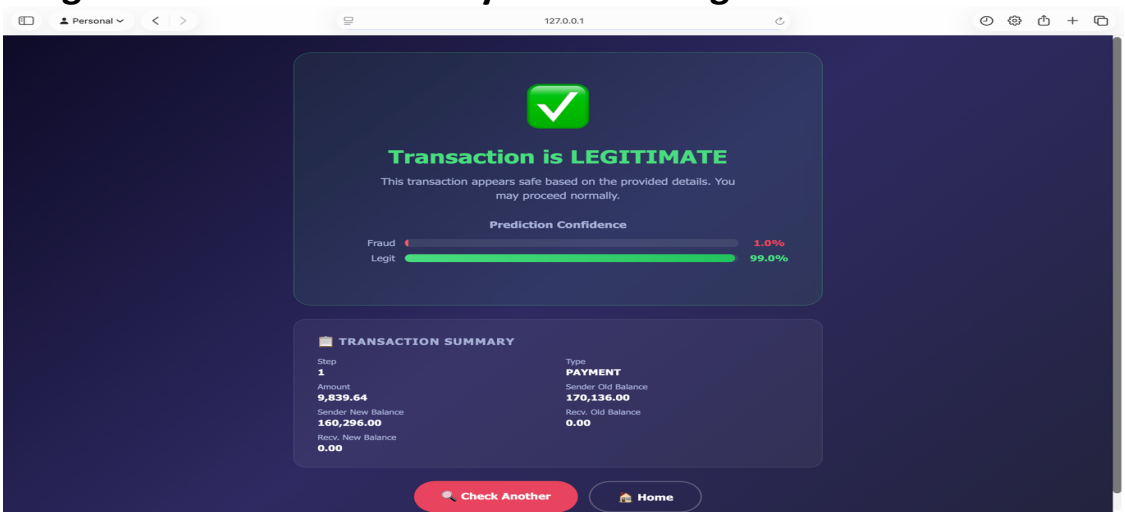
- **Fraud Prediction Output**

Once the user submits transaction details:

1. The Flask backend receives input data.
2. Data is preprocessed in the required format.
3. The trained Random Forest model performs classification.
4. The prediction result (Fraud / Non-Fraud) is displayed to the user.

The prediction is generated within approximately 1–2 seconds, ensuring real-time fraud detection capability.

• Figure 3: Transaction Analysis Result Page



After submitting transaction details, the system performs the following steps:

1. Input data is received by the Flask backend
2. Data is preprocessed into model-compatible format
3. The trained Random Forest model performs classification
4. Prediction result is displayed on the result page

The result page clearly shows:

- **Fraudulent Transaction** (with warning symbol and red highlight)
  OR
- **Safe Transaction** (with green confirmation message)

It also provides:

- Clear explanation message
- "Check Another Transaction" button for repeated testing

This visual feedback enhances user understanding and supports quick decision-making.

Summary

The final Random Forest model achieved a test accuracy of 99.24% with an inference time of approximately 1–2 seconds, delivering stable predictions across multiple test cases and effectively detecting fraudulent patterns. This high accuracy demonstrates that the model successfully learned transaction behavior patterns and generalizes well to unseen data.

8 ADVANTAGES & DISADVANTAGES

8.2 Advantages

1. **Real-Time Fraud Detection**
    - The system generates predictions within **1–2 seconds**, enabling quick decision-making.
2. **User-Friendly Interface**
    - The Flask-based web application provides a clean and structured interface for entering transaction details.
3. **Multiple Model Comparison**
    - Different machine learning algorithms (Decision Tree, Random Forest, XGBoost) were evaluated before selecting the final model, ensuring optimal performance.
4. **Extensible Architecture**
    - The modular design supports easy extension to include more rice types or switch to different crop classification tasks.
5. **Scalable Architecture**
    - The modular structure allows easy integration into real-world banking systems or cloud environments.
6. **Reduced Manual Effort**
    - Automates fraud detection and reduces dependency on human monitoring.
7. **Cost-Effective Solution**
    - Does not require complex hardware; runs efficiently on local systems.

8.3 Disadvantages

1. **Imbalanced Dataset Risk**
    - Fraud detection datasets are typically highly imbalanced, which may affect prediction reliability in real-world scenarios.
2. **Limited to Structured Data**
    - The current system analyzes numerical transaction data only and does not consider behavioral biometrics or advanced anomaly signals.
3. **Not Yet Deployed on Cloud**
    - The application is currently tested locally and not deployed in a production environment.
4. **No Continuous Learning Mechanism**
    - The model does not automatically retrain with new incoming transaction data.
5. **No User Authentication Layer**
    - The current version lacks login or role-based access control.

9 CONCLUSION

The Online Payments Fraud Detection Using Machine Learning project successfully demonstrates the application of Artificial Intelligence in the financial domain.

By implementing and comparing multiple machine learning algorithms, the Random Forest model was selected as the most effective classifier. The final model achieved **99.24% test accuracy**, indicating strong generalization capability and reliable fraud detection performance.

The integration of the trained model into a Flask-based web application enables real-time prediction of fraudulent transactions. The system provides a user-friendly interface that simplifies fraud detection for financial analysts and organizations.

Key Achievements:

- Evaluated multiple ML algorithms
- Selected optimal model based on performance
- Achieved high prediction accuracy
- Developed complete end-to-end ML pipeline
- Successfully integrated model with web application

This project highlights the importance of data-driven fraud detection systems in enhancing digital payment security and minimizing financial losses.

10 FUTURE SCOPE

Although the current system performs effectively, several improvements can enhance its capabilities:

---

1. Cloud Deployment

- The system can be deployed on cloud platforms such as AWS, Azure, or Google Cloud. This would allow integration with real-time banking systems and enable large-scale transaction monitoring.

---

2. Real-Time Transaction Processing

- Instead of manually entering transaction details, the model can be connected to live payment APIs to monitor and classify transactions instantly.

---

3. Improved Imbalance Handling

- Since fraud datasets are highly imbalanced, techniques such as SMOTE or cost-sensitive learning can be implemented to improve detection of rare fraud cases.

---

4. Continuous Model Retraining

- Fraud patterns change over time. Implementing automatic model retraining using new transaction data can help maintain high accuracy. Behavioral and Contextual Features
- Additional features like user login patterns, device information, IP address tracking, and geographic data can improve fraud prediction capability.

---

5. Fraud Analytics Dashboard

- An administrative dashboard can be developed to visualize fraud statistics, trends, and model performance metrics.

---

6. Security Enhancements

- Future systems can include authentication mechanisms, encrypted data transmission, and secure APIs for enterprise-level deployment.

---

7. Hybrid Fraud Detection System

- A combination of rule-based filtering and machine learning models can be implemented for multi-layer fraud detection.

---

11 APPENDIX

The Appendix section provides supporting information related to the implementation, dataset, and tools used in the Online Payments Fraud Detection project.

11.1 Source Code Repository

The complete source code for the Online Payments Fraud Detection system is maintained in a version-controlled repository.

The repository includes:

- Data preprocessing scripts
- Model training notebook
- Implementation of Decision Tree, Random Forest, and XGBoost
- Final Random Forest model (.pkl file)
- Flask backend (app.py)
- HTML templates for user interface

❶GitHub Repository:

https://github.com/Sarvepallisai/ONLINE_PAYMENTS_FRAUD_DETECTION

11.2 Dataset Used

The dataset used for this project contains historical online payment transaction records.

Key Features:

- Step
- Transaction Type
- Amount
- OldbalanceOrg
- NewbalanceOrig
- OldbalanceDest
- NewbalanceDest
- isFraud (Target Variable)

The dataset was:

- Cleaned and preprocessed
- Split into training and testing datasets
- Used to evaluate multiple machine learning models

Dataset Repository (if applicable):

https://www.kaggle.com/datasets/ealaxi/paysim1

11.3 Project Demo Video

A demonstration video was prepared to showcase the working of the system.

The demo includes:

- Overview of the project
- Explanation of dataset
- Model training overview

- Web application walkthrough
- Fraud prediction demonstration
- Code explanation (Flask + Model integration)

Demo Video:

https://drive.google.com/file/d/1RmTVsIicuAD_xrWT8Cc98adhkUxMurq2/view?usp=drivesdk

---

11.4 Tools and Technologies

11.4.1 The following technologies were used in the development of the system:
11.4.2 Python (Programming Language)
11.4.3 Scikit-learn (Machine Learning)
11.4.4 Random Forest Algorithm
11.4.5 Pandas & NumPy (Data Handling)
11.4.6 Matplotlib / Seaborn (Data Visualization)
11.4.7 Flask (Web Framework)
11.4.8 HTML & CSS (Frontend Design)
11.4.9 Jupyter Notebook (Model Development)
