ONLINE PAYMENTS FRAUD DETECTION SYSTEM USING MACHINE LEARNING

A Machine Learning Based Web Application

Project Report

Developed By

- Kartheeka Chevvakula
- Depuru Joshika Reddy
- Karasimangalam Himasree
- Karasimangalam Lavanya

Academic Year: 2022 – 2026

(Sri Venkateswara College of Engineering (SVCE) – CSE (Data Science)

ABSTRACT

With the rapid growth of digital payment systems and online financial transactions, the risk of fraudulent activities has significantly increased. Online payment fraud leads to severe financial losses for individuals, businesses, and financial institutions. Detecting fraudulent transactions accurately and efficiently has become a critical challenge in the financial technology domain.

This project presents an Online Payments Fraud Detection System using Machine Learning techniques to identify fraudulent transactions based on transaction-related features. The dataset used in this project contains various transaction attributes such as step, transaction type, amount, old balance, and new balance information. Exploratory Data Analysis (EDA) was performed to understand data distribution, detect class imbalance, and analyze correlations between features.

Multiple Machine Learning models were trained and evaluated, including Logistic Regression, Decision Tree, and Random Forest classifiers. The performance of each model was assessed using accuracy, confusion matrix, and classification report metrics such as precision, recall, and F1-score. Among the trained models, the Random Forest classifier achieved the highest performance with an accuracy of 99.97%, making it the best model for fraud detection in this system.

The selected model was deployed using the Flask web framework to create a user-friendly web application. The system allows users to input transaction details and receive real-time predictions indicating whether a transaction is fraudulent or legitimate, along with a confidence score. This project demonstrates the effectiveness of Machine Learning in detecting online payment fraud and provides a practical implementation for real-time fraud detection systems.

PROBLEM STATEMENT

The rapid expansion of digital payment platforms and online financial transactions has significantly increased the risk of fraudulent activities. Fraudulent online transactions result in substantial financial losses for customers, merchants, and financial institutions. Detecting such fraud in real time has become a major challenge due to the increasing transaction volume and the evolving techniques used by fraudsters.

One of the key challenges in fraud detection is the highly imbalanced nature of transaction datasets. In real-world financial data, legitimate transactions greatly outnumber fraudulent transactions. This imbalance makes it difficult for traditional classification models to accurately identify fraud cases without being biased toward the majority class. As a result, achieving high accuracy alone is not sufficient; the system must also maintain strong precision and recall for fraud detection.

The objective of this project is to develop a robust Machine Learning-based system capable of accurately classifying online payment transactions as either fraudulent or legitimate. The system should effectively handle class imbalance, minimize false positives and false negatives, and provide reliable predictions in a user-friendly web interface. The final solution should support real-time prediction through a deployed Flask application, enabling practical and scalable fraud detection.

1. INTRODUCTION

Online payment systems have become an essential part of modern digital transactions. With the rapid growth of e-commerce, mobile banking, and digital wallets, millions of financial transactions are processed every day. However, this increase in online transactions has also led to a significant rise in fraudulent activities.

Online payment fraud occurs when unauthorized transactions are performed with the intention of financial gain. Fraudsters exploit system vulnerabilities, manipulate transaction data, and use stolen account details to perform illegal activities. Such fraud results in financial losses, reduced customer trust, and operational challenges for financial institutions.

Machine Learning provides an effective solution for detecting fraudulent transactions by analyzing transaction patterns and identifying suspicious behavior. By training models on historical transaction data, it becomes possible to classify new transactions as either legitimate or fraudulent.

This project focuses on developing a Machine Learning-based fraud detection system that analyzes transaction features such as amount, balance changes, and transaction type to predict fraud. The system also includes a Flask-based web application for real-time fraud prediction.

DATASET DESCRIPTION

The dataset used in this project is titled "Online_Payment_Fraud_Detection.csv". It contains detailed transaction-level information representing real-world online payment activities. Each record corresponds to a single financial transaction and includes multiple attributes describing transaction behavior and account balance variations.

The dataset consists of over 6 million transaction records, making it large enough to simulate real-time fraud detection scenarios and evaluate machine learning models effectively.

The primary features used in this study are described below:

• Step – Represents the time step of the transaction, indicating when the transaction occurred.
• Type – Specifies the type of transaction (e.g., PAYMENT, TRANSFER, CASH_OUT).
• Amount – The monetary value involved in the transaction.
• OldbalanceOrg – The sender's account balance before the transaction.
• NewbalanceOrig – The sender's account balance after the transaction.
• OldbalanceDest – The recipient's account balance before the transaction.
• NewbalanceDest – The recipient's account balance after the transaction.
• isFraud – The target variable that indicates whether a transaction is fraudulent (1) or legitimate (0).

A key characteristic of this dataset is class imbalance. Legitimate transactions constitute the vast majority of the data, while fraudulent transactions account for only a very small percentage. This imbalance poses a significant challenge in model training, as machine learning algorithms may become biased toward predicting the majority class. Therefore, appropriate evaluation metrics and class balancing techniques are essential to ensure accurate fraud detection.

SYSTEM ARCHITECTURE

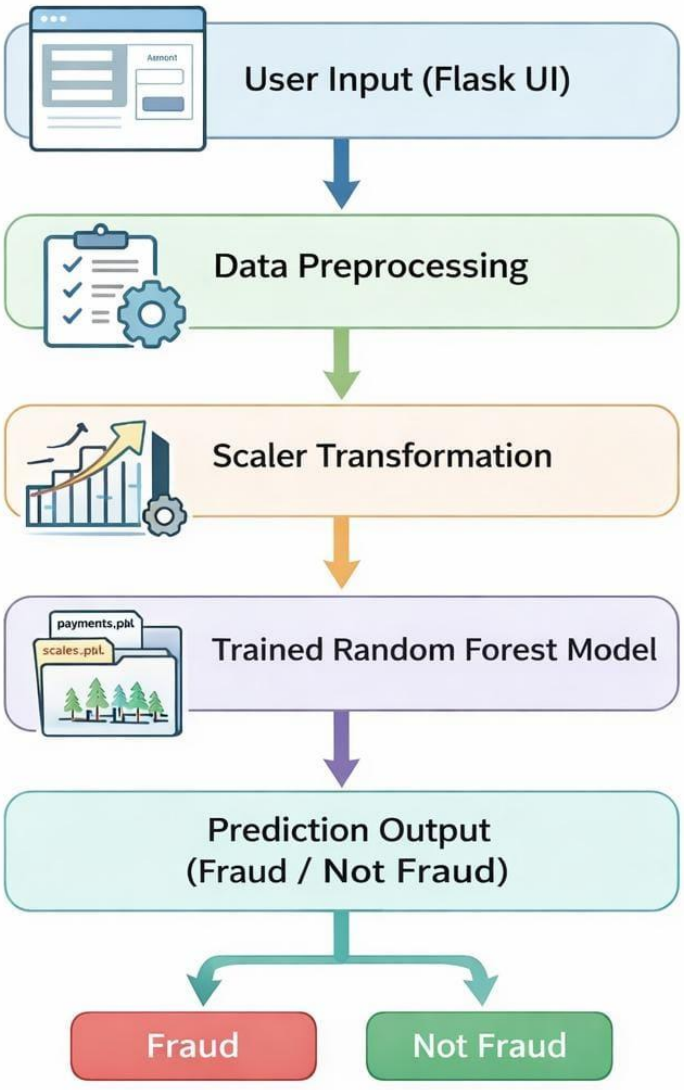


Figure 1: System Architecture of Online Fraud Detection System

The Online Payments Fraud Detection System is designed using a layered architecture that integrates user interaction, data processing, machine learning prediction, and result display.

The system begins with the user interface, where transaction details are entered through a web-based application developed using HTML and CSS. The frontend provides an interactive and user-friendly environment for inputting transaction-related information such as transaction type, amount, and account balances.

The backend of the system is implemented using the Flask framework. Flask handles user requests, processes input data, and communicates with the trained machine learning model to generate predictions.

Once the user submits transaction details, the data undergoes preprocessing. Categorical variables such as transaction type are encoded into numerical format. Numerical features are then standardized using StandardScaler to ensure consistent feature scaling before model prediction.

The processed data is passed to the trained RandomForestClassifier model, which has been selected as the best-performing model during experimentation. The model is stored as payments.pkl, and the scaler is stored as scaler.pkl. These files are loaded during runtime to perform real-time predictions.

Finally, the system generates a prediction output indicating whether the transaction is Fraud or Not Fraud, along with a confidence score. The result is displayed on the web interface for the user.

This architecture ensures efficient data flow, accurate prediction, and seamless integration between the machine learning model and the web application.

EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) was performed to understand the structure, distribution, and relationships within the dataset. EDA helps in identifying patterns, detecting anomalies, and understanding class imbalance before model training.

Fraud Distribution Analysis:

The fraud distribution graph clearly shows that fraudulent transactions are significantly fewer compared to legitimate transactions. This confirms the presence of severe class imbalance in the dataset, which is a critical challenge in fraud detection.

Boxplot Analysis:

The boxplot visualization highlights the presence of outliers in transaction amounts and balance-related features. Fraudulent transactions often exhibit unusual patterns compared to normal transactions, making outlier detection important.

Correlation Heatmap:

The heatmap displays the correlation between numerical features. Strong relationships between balance-related variables can be observed. Understanding these correlations helps in selecting relevant features for model training.

Histogram Distribution:

The histogram shows the distribution of numerical features such as transaction amount and account balances. It helps in identifying skewness and understanding the overall spread of data.

Through EDA, important insights regarding imbalance, feature relationships, and data distribution were obtained, which guided preprocessing and model selection decisions.

DATA PREPROCESSING

Data preprocessing is an important step in building an effective Machine Learning model. Since the dataset contains both numerical and categorical features, proper preprocessing ensures better model performance and accurate predictions.
The following preprocessing steps were performed:

1. Handling Categorical Variables

The Type column represents the transaction category (PAYMENT, TRANSFER, CASH_OUT, etc.).
Since machine learning models cannot process categorical data directly, this feature was converted into numerical format using encoding techniques.

2. Feature Selection

The following features were selected for model training:
Step
Type
Amount
OldbalanceOrg
NewbalanceOrig
OldbalanceDest
NewbalanceDest
The target variable used for prediction is:
isFraud (1 = Fraud, 0 = Not Fraud)

3. Feature Scaling

Since transaction amounts and balances vary widely, feature scaling was applied using StandardScaler.
Scaling helps:
Normalize feature values
Improve model convergence
Increase prediction accuracy
The trained scaler was saved as:

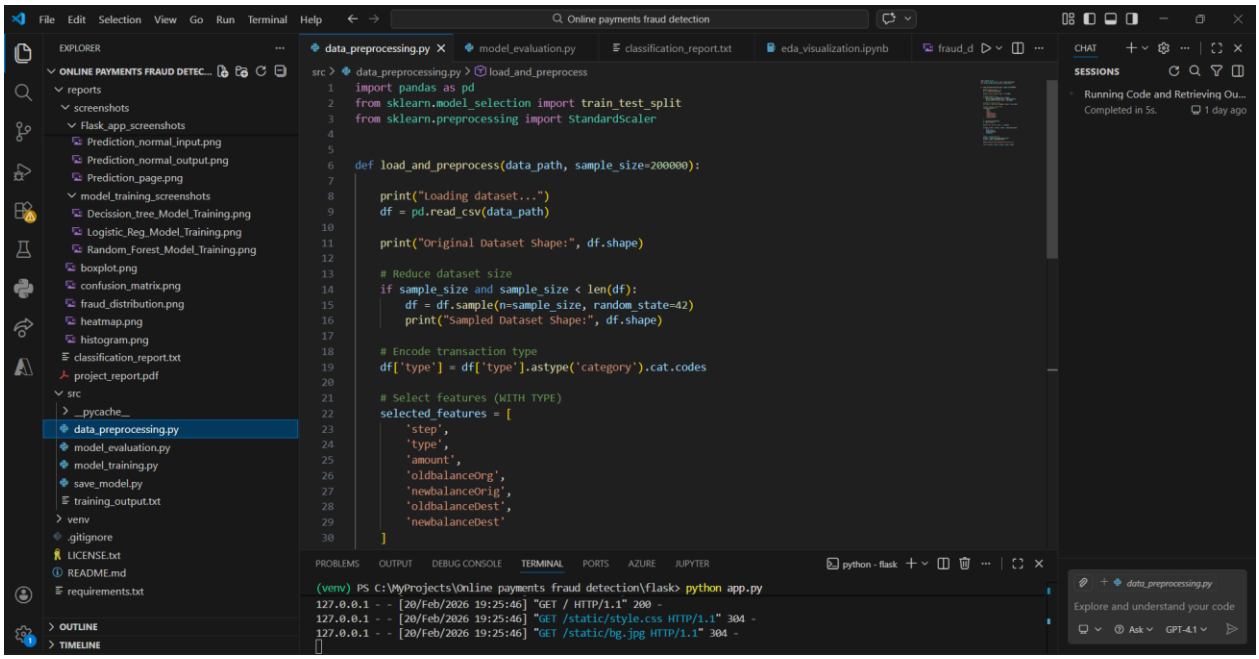
scaler.pkl

4. Train-Test Split

The dataset was divided into:
Training Set (80%)
Testing Set (20%)
This ensures the model is evaluated on unseen data.

5. Handling Class Imbalance

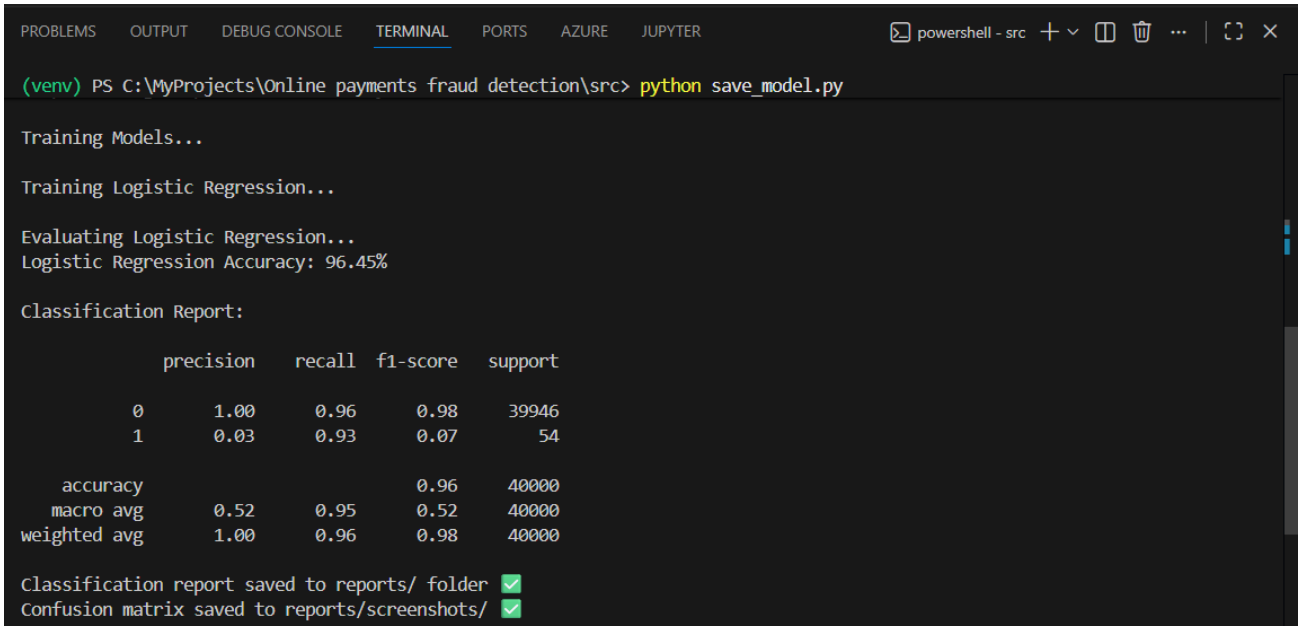
The dataset is highly imbalanced. Fraud cases are very few compared to legitimate transactions.
To handle this, the Random Forest model was trained using:
class_weight = 'balanced'
This helps the model give more importance to fraud cases during training.



MODEL TRAINING

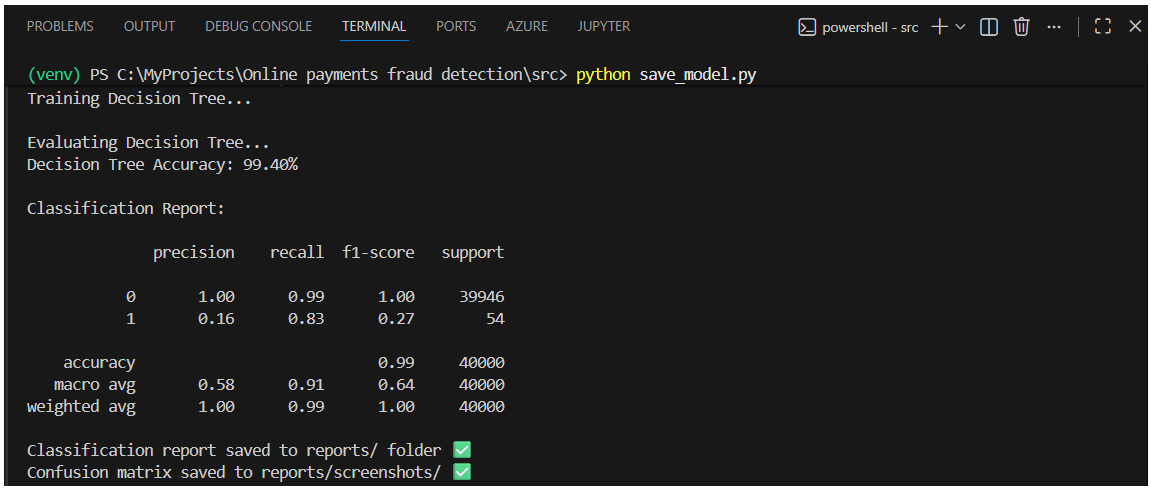
To build an accurate fraud detection system, multiple machine learning models were trained and evaluated.
The following models were used:

1. Logistic Regression



A simple and interpretable linear classification model used as a baseline for comparison.

2. Decision Tree Classifier



A tree-based model that splits data based on feature importance and decision rules.

3. Random Forest Classifier

```
(venv) PS C:\MyProjects\Online payments fraud detection\src> python save_model.py

Evaluating Random Forest...
Random Forest Accuracy: 99.97%

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     39946
           1       0.96      0.80      0.87        54

    accuracy                           1.00     40000
   macro avg       0.98      0.90      0.93     40000
weighted avg       1.00      1.00      1.00     40000

Classification report saved to reports/ folder ✅
Confusion matrix saved to reports/screenshots/ ✅

----------------------------------
Best Model Selected: RandomForestClassifier(class_weight='balanced', max_depth=12, n_estimators=150,
                     n_jobs=-1, random_state=42)
Accuracy: 99.97%
----------------------------------


Model saved at: ../models/payments.pkl
Scaler saved at: ../models/scaler.pkl

Training Pipeline Completed Successfully ✅
```

An ensemble learning model that combines multiple decision trees to improve accuracy and reduce overfitting.

Model Comparison

Each model was trained using the processed dataset and evaluated using accuracy, precision, recall, and F1-score.
After comparing all models:
Random Forest was selected as the best-performing model with an accuracy of 99.97%.
Random Forest performed better because:
It handles imbalanced datasets effectively
It reduces overfitting using ensemble learning
It captures complex feature relationships
The trained model was saved as:

payments.pkl

MODEL EVALUATION

After training the models, performance evaluation was carried out using standard classification metrics. The Random Forest model, which performed best, was evaluated using the test dataset.
The following evaluation metrics were considered:

1. Accuracy
Accuracy represents the percentage of correctly classified transactions.
The Random Forest model achieved:
Accuracy: 99.97%
This indicates that the model correctly classifies almost all transactions.

2. Precision
Precision measures how many predicted fraud transactions were actually fraud.
High precision ensures fewer false fraud alerts.

3. Recall (Very Important for Fraud Detection)
Recall measures how many actual fraud transactions were correctly detected.
In fraud detection systems, recall is extremely important, because missing a fraud transaction (False Negative) can result in financial loss.
The model achieved high recall for fraud class, showing its ability to effectively detect fraudulent transactions.

4. F1-Score
F1-score is the harmonic mean of precision and recall.
It balances both false positives and false negatives.

Confusion Matrix
confusion_matrix.png

Classification Report
classification_report.txt

The classification report confirms that Random Forest achieves excellent performance across precision, recall, and F1-score metrics for both fraud and non-fraud classes.

FLASK WEB APPLICATION

To make the fraud detection system practical and user-friendly, a web-based application was developed using the Flask framework. This allows users to enter transaction details and receive real-time fraud predictions.

System Workflow

The working of the Flask application follows these steps:

- The user enters transaction details in the web interface.
- The input data is sent to the Flask backend.
- Data preprocessing and feature scaling are applied using the saved scaler.
- The trained Random Forest model predicts whether the transaction is Fraud or Not Fraud.
- The prediction result along with the confidence score is displayed to the user.

Frontend Development

The frontend was developed using:

HTML

CSS

The user interface includes:

- Input form for transaction details
- Dropdown for transaction type
- Submit button for prediction
- Result display page showing:
- Prediction (Fraud / Not Fraud)
- Confidence score
- The interface is designed to be simple, clean, and easy to use.

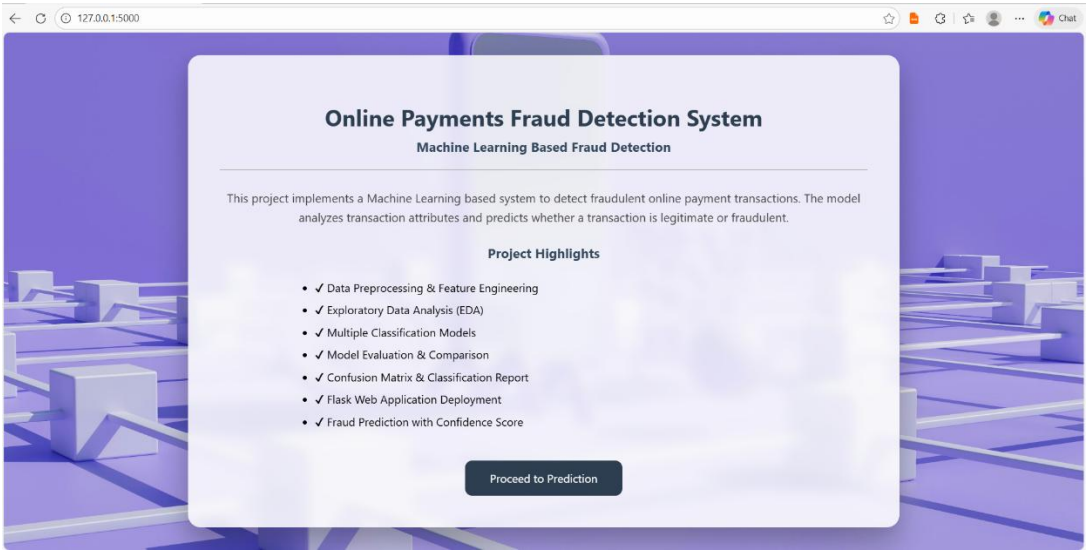
Backend Development

The backend was developed using:

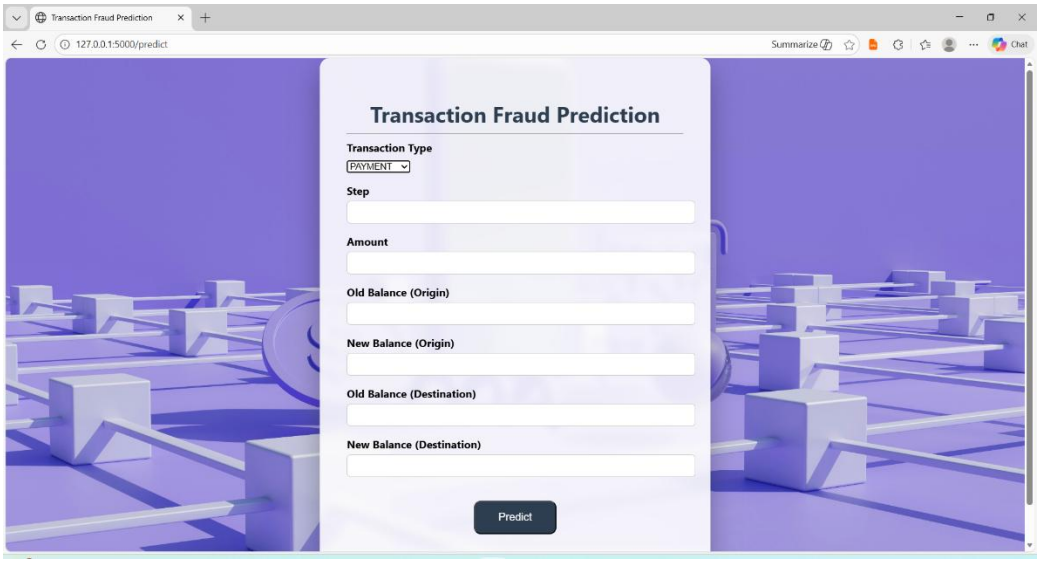
- Python
- Flask
- The backend performs the following operations:
- Loads trained model (payments.pkl)
- Loads scaler (scaler.pkl)
- Receives user input

- Converts input into required format
- Applies scaling transformation
- Generates prediction using Random Forest
- Sends result to frontend

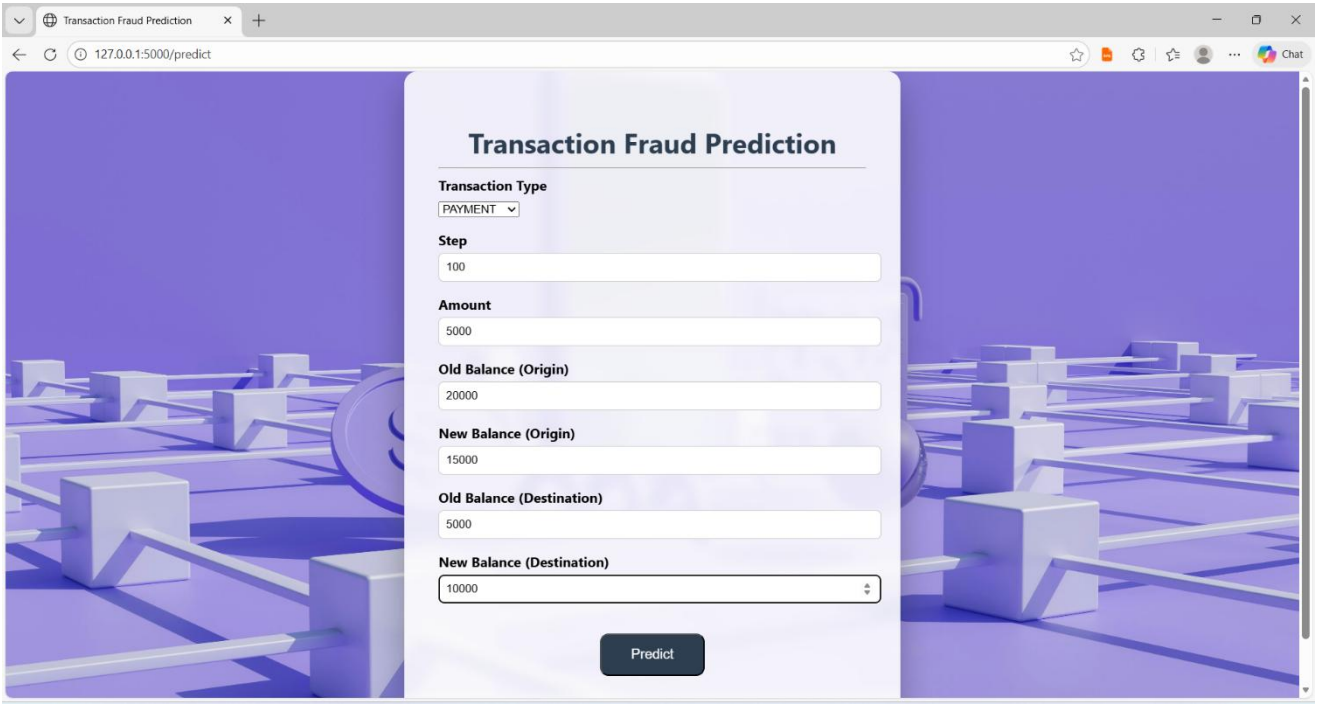
reports/screenshots/Flask_app_screenshots/
Flask_homepage.png



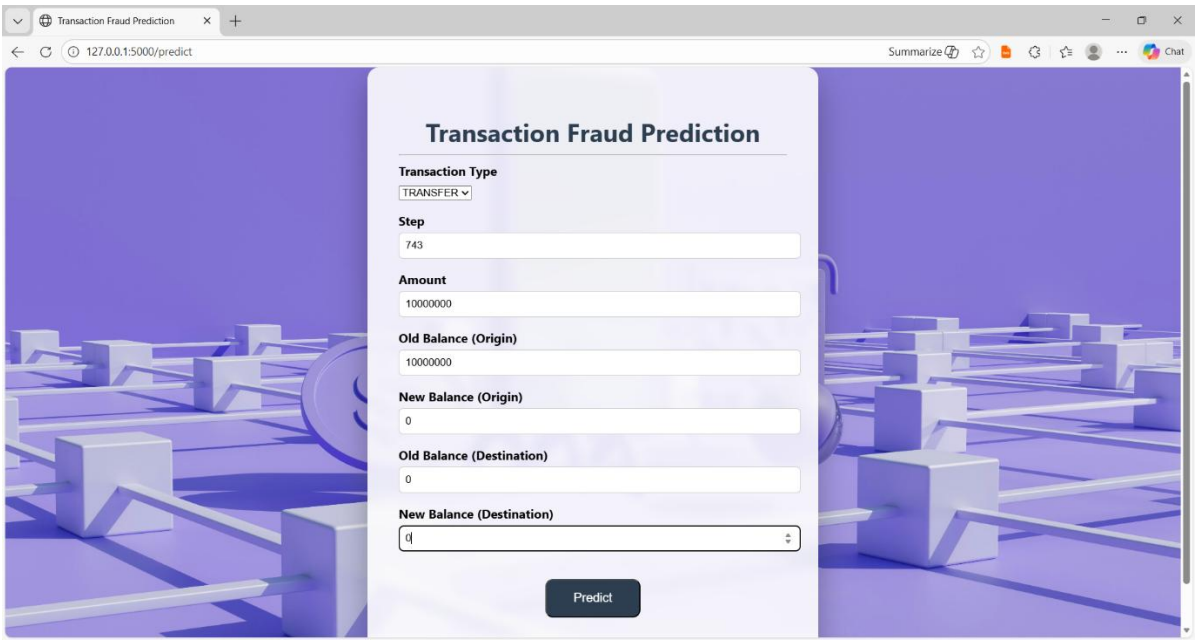
Prediction_page.png



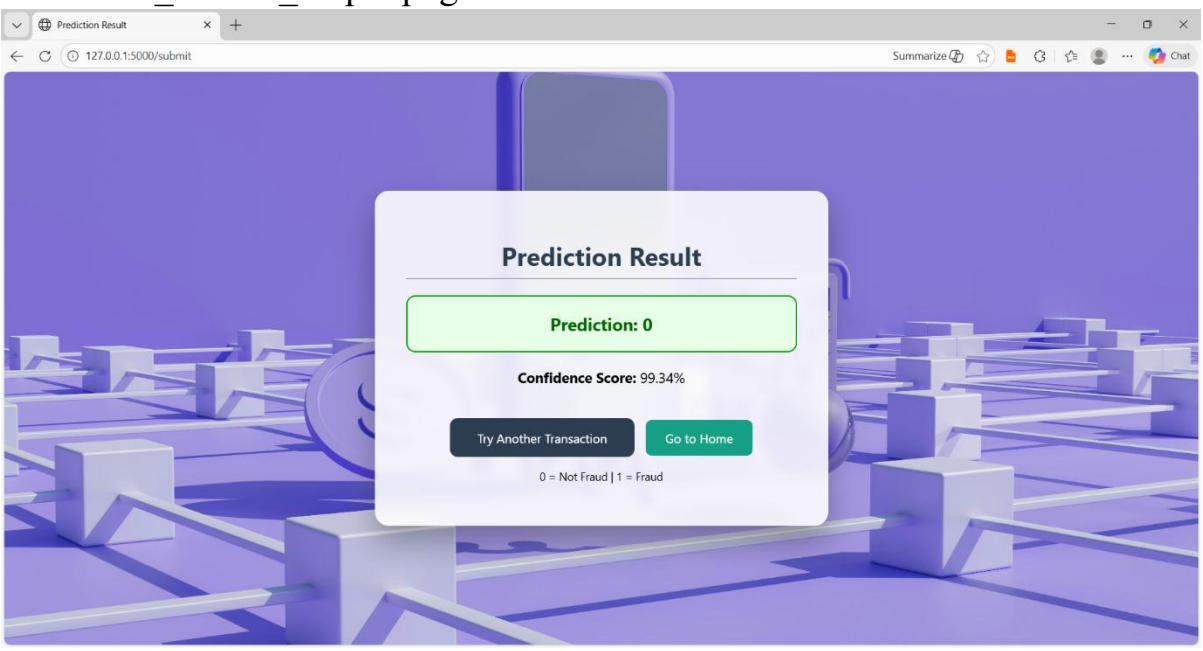
Prediction_normal_input.png



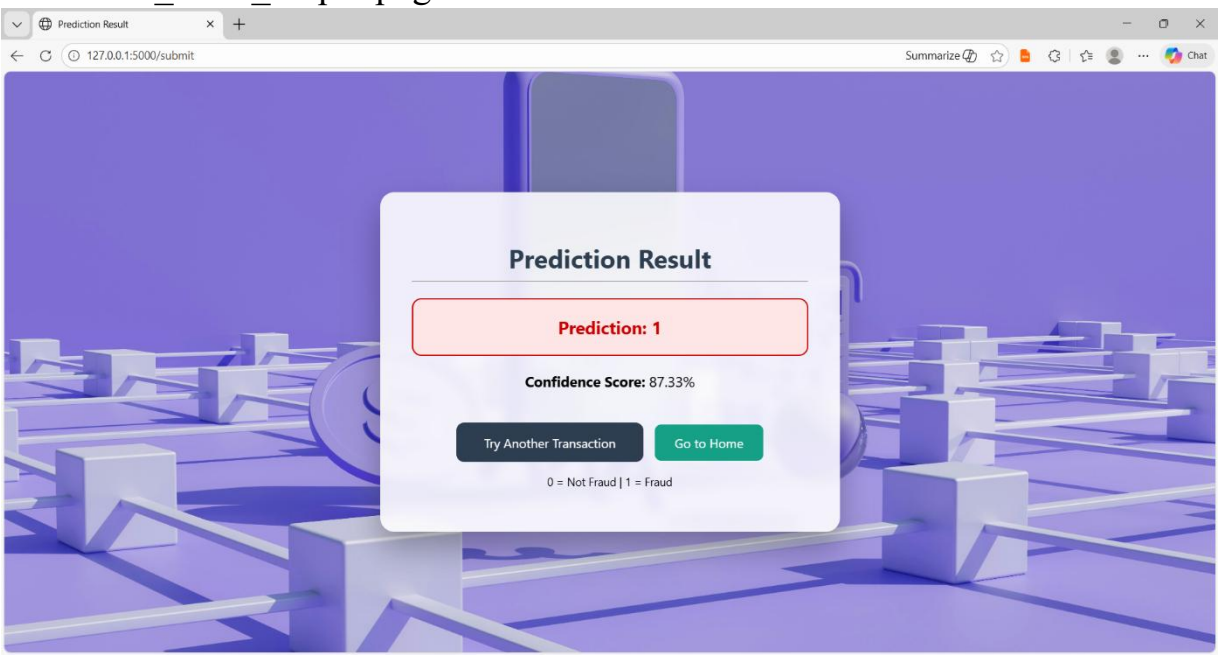
Prediction_fraud_input.png



Prediction_normal_output.png



Prediction_fraud_output.png



The Flask application successfully predicts whether a transaction is Fraud or Not Fraud and displays the confidence score, demonstrating real-time fraud detection capability.

PROJECT STRUCTURE

The project follows a well-organized folder structure to maintain modularity and clarity.

```
Online_Payments_Fraud_Detection/
│
├── data/
│   └── Online_Payment_Fraud_Detection.csv
│
├── flask/
│   ├── templates/
│   ├── static/
│   └── app.py
│
├── models/
│   ├── payments.pkl
│   └── scaler.pkl
│
├── notebooks/
│   └── eda_visualization.ipynb
│
├── reports/
│   ├── screenshots/
│   ├── classification_report.txt
│   └── project_report.pdf
│
├── src/
│   ├── data_preprocessing.py
│   ├── model_training.py
│   ├── model_evaluation.py
│   └── save_model.py
│
├── requirements.txt
└── README.md
```

Explanation of Each Folder

data/
Contains the dataset used for training and evaluation.

flask/
Contains the web application code including:
Templates (HTML files)
Static files (CSS, images)
Main Flask application file (app.py)

models/
Stores the trained model and scaler:
payments.pkl
scaler.pkl

notebooks/
Contains Jupyter notebook used for Exploratory Data Analysis (EDA).

reports/
Contains:
Screenshots
Classification report
Final project report (PDF)

src/
Contains source code for:
Data preprocessing
Model training
Model evaluation
Model saving

TECHNOLOGIES USED

This project was developed using various tools and technologies for data analysis, machine learning, and web deployment.

Programming Language
Python – Used for data preprocessing, model training, evaluation, and backend development.

Libraries and Frameworks

1. Pandas
Used for:
Data loading
Data cleaning
Data manipulation
Handling large datasets efficiently

2. NumPy
Used for:
Numerical computations
Array operations
Mathematical calculations

3. Scikit-learn
Used for:
Model training (Logistic Regression, Decision Tree, Random Forest)
Feature scaling (StandardScaler)
Train-test split
Model evaluation metrics

4. Matplotlib
Used for:
Data visualization
Plotting graphs (histograms, confusion matrix, etc.)

5. Seaborn
Used for:
Heatmaps
Statistical visualizations
Correlation analysis

6. Flask
Used for:
Building web application
Backend routing
Integrating ML model with user interface

Frontend Technologies
HTML – Designing structure of web pages
CSS – Styling and layout of web application

Development Tools
VS Code
Jupyter Notebook
Git & GitHub

These technologies together enabled the development of a complete end-to-end Machine Learning based Fraud Detection System.

CONCLUSION

In this project, a Machine Learning-based Online Payments Fraud Detection System was successfully designed, developed, and deployed.

The dataset used in this project contained over 6 million transaction records and was highly imbalanced. Proper preprocessing techniques such as feature scaling, encoding, and class balancing were applied to improve model performance.

Three machine learning models were trained and evaluated:
Logistic Regression
Decision Tree
Random Forest

Among them, Random Forest achieved the highest performance with 99.97% accuracy and strong fraud detection capability.

The trained model was integrated into a Flask web application, enabling real-time prediction of fraudulent transactions based on user input.

This project demonstrates how Machine Learning and Web Technologies can be combined to build practical and scalable fraud detection systems for financial applications.

FUTURE ENHANCEMENTS

Although the current system performs effectively, several improvements can be made to enhance its capabilities.

1. Cloud Deployment
Deploy the Flask application on cloud platforms such as AWS, Azure, or Heroku for real-world usage.

2. Database Integration
Integrate a database to:
Store transaction history
Log predictions
Monitor fraud trends

3. Improve Fraud Recall
Further optimize the model to improve fraud recall and reduce false negatives.

4. Use Deep Learning Models
Implement advanced models such as:
Neural Networks
LSTM (for sequential transaction analysis)

5. Real-Time Monitoring Dashboard
Develop an admin dashboard to:
Track fraud statistics
Visualize live transaction patterns

These enhancements can make the system more robust, scalable, and suitable for enterprise-level financial applications.