

Online Payments Fraud Detection using Machine learning

Team ID : LTVIP2026TMIDS40761

Team Leader : Grandhi Anjana Priya

Team member : Bayapuri Srinivasa Prasanna
Kumar

Team member : Bathini Venkata Rohit

Team member : Chukka Charan Bhuvanesh

1. Abstract

Online payment systems have become an essential part of modern digital transactions. However, the increase in online payments has also led to a rise in fraudulent activities. This project aims to develop an efficient fraud detection system using machine learning techniques. By analyzing transaction data, the system can classify transactions as fraudulent or legitimate, helping financial institutions reduce financial losses and improve security.

2. Introduction

With the rapid growth of e-commerce and digital banking, online payment fraud has become a major concern. Traditional rule-based systems are not sufficient to handle complex fraud patterns. Machine

learning provides an effective solution by learning from historical transaction data and detecting hidden patterns. This project focuses on building a fraud detection model using supervised learning algorithms.

3. Problem Statement

The main objective of this project is to design and implement a system that can accurately identify fraudulent online payment transactions.

The system should:

- Detect fraud in real-time or near real-time.
- Minimize false positives and false negatives.
- Improve accuracy using machine learning techniques.

4. Objectives

- To study online payment fraud patterns.
- To preprocess and analyze transaction datasets.
- To apply machine learning algorithms for fraud detection.
- To evaluate model performance.
- To deploy a simple web-based prediction system.

5. Scope of the Project

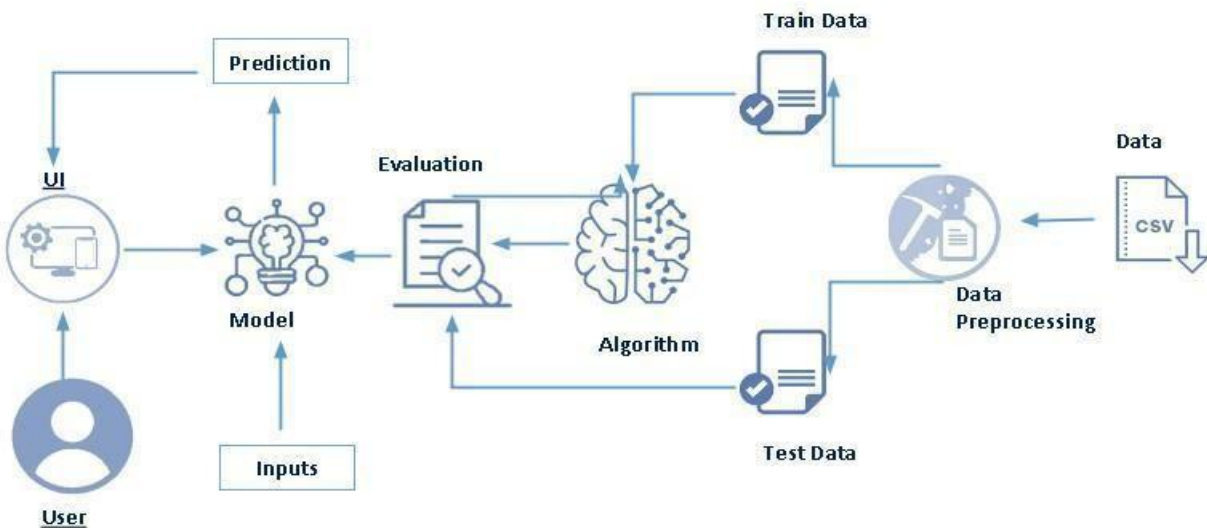
This project is limited to analyzing structured transaction data. It focuses on binary classification (fraud / not fraud). The system can be extended in the future to include deep learning models, real-time monitoring, and mobile applications.

6. Literature Survey

Several studies have shown that machine learning techniques such as Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines are effective in detecting fraud. Research also

highlights the importance of handling class imbalance and feature selection to improve model accuracy.

7. System Architecture



8. Dataset Description

The dataset contains information about online payment transactions. Typical attributes include:

- Step (time index)
- Transaction Type
- Amount
- Sender Balance (Old and New)
- Receiver Balance (Old and New)
- Fraud Label (isFraud)

Dataset Link:

<https://www.kaggle.com/datasets/rupakroy/onlinepayments-fraud-detection-dataset>

```

PS_20174392719_1491204439457_log.csv
1  step,type,amount,nameOrig,oldbalanceOrig,newbalanceOrig,nameDest,oldbalanceDest,newbalanceDest,isFraud,isFlaggedFraud
2  1,PAYMENT,9839.64,C1231006815,170136.0,160296.36,M1979787155,0.0,0.0,0,0
3  1,PAYMENT,1864.28,C1666544295,21249.0,19384.72,M2044282225,0.0,0.0,0,0
4  1,TRANSFER,181.0,C1305486145,181.0,0.0,C553264065,0.0,0.0,1,0
5  1,CASH_OUT,181.0,C840083671,181.0,0.0,C38997010,21182.0,0.0,1,0
6  1,PAYMENT,11668.14,C2048537720,41554.0,29885.86,M1230701703,0.0,0.0,0,0
7  1,PAYMENT,7817.71,C90045638,53860.0,46042.29,M573487274,0.0,0.0,0,0
8  1,PAYMENT,7107.77,C154988899,183195.0,176087.23,M408069119,0.0,0.0,0,0
9  1,PAYMENT,7861.64,C1912850431,176087.23,168225.59,M633326333,0.0,0.0,0,0
10 1,PAYMENT,4024.36,C1265012928,2671.0,0.0,M1176932104,0.0,0.0,0,0
11 1,DEBIT,5337.77,C712410124,41720.0,36382.23,C195600860,41898.0,40348.79,0,0
12 1,DEBIT,9644.94,C1900366749,4465.0,0.0,C997608398,10845.0,157982.12,0,0
13 1,PAYMENT,3099.97,C249177573,20771.0,17671.03,M2096539129,0.0,0.0,0,0
14 1,PAYMENT,2560.74,C1648232591,5070.0,2509.26,M972865270,0.0,0.0,0,0
15 1,PAYMENT,11633.76,C1716932897,10127.0,0.0,M801569151,0.0,0.0,0,0
16 1,PAYMENT,4098.78,C1026483832,503264.0,499165.22,M1635378213,0.0,0.0,0,0
17 1,CASH_OUT,229133.94,C905080434,15325.0,0.0,C476402209,5083.0,51513.44,0,0
18 1,PAYMENT,1563.82,C761750706,450.0,0.0,M1731217984,0.0,0.0,0,0
19 1,PAYMENT,1157.86,C1237762639,21156.0,19998.14,M1877062907,0.0,0.0,0,0
20 1,PAYMENT,671.64,C2033524545,15123.0,14451.36,M473053293,0.0,0.0,0,0
21 1,TRANSFER,215310.3,C1670993182,705.0,0.0,C1100439041,22425.0,0.0,0,0
22 1,PAYMENT,1373.43,C20804602,13854.0,12480.57,M1344519051,0.0,0.0,0,0
23 1,DEBIT,9302.79,C1566511282,11299.0,1996.21,C1973538135,29832.0,16896.7,0,0
24 1,DEBIT,1065.41,C1959239586,1817.0,751.59,C515132998,10330.0,0.0,0,0
25 1,PAYMENT,3876.41,C504336483,67852.0,63975.59,M1404932042,0.0,0.0,0,0
26 1,TRANSFER,311685.89,C1984094095,10835.0,0.0,C932583850,6267.0,2719172.89,0,0
27 1,PAYMENT,6061.13,C1043358826,443.0,0.0,M1558079303,0.0,0.0,0,0
28 1,PAYMENT,9478.39,C1671590089,116494.0,107015.61,M58488213,0.0,0.0,0,0
29 1,PAYMENT,8009.09,C1053967012,10968.0,2958.91,M295304806,0.0,0.0,0,0
30 1,PAYMENT,8901.99,C1632497828,2958.91,0.0,M33419717,0.0,0.0,0,0
31 1,PAYMENT,9920.52,C764826684,0.0,0.0,M1940055334,0.0,0.0,0,0
32 1,PAYMENT,3448.92,C2103763750,0.0,0.0,M335107734,0.0,0.0,0,0
33 1,PAYMENT,4206.84,C215078753,0.0,0.0,M1757317128,0.0,0.0,0,0
34 1,PAYMENT,5885.56,C840514538,0.0,0.0,M1804441305,0.0,0.0,0,0
35 1,PAYMENT,5307.88,C1768242710,0.0,0.0,M1971783162,0.0,0.0,0,0
36 1,PAYMENT,5031.22,C247113419,0.0,0.0,M151442075,0.0,0.0,0,0
37 1,PAYMENT,24213.67,C1238616099,0.0,0.0,M70695990,0.0,0.0,0,0

```

9. Methodology

9.1 Data Preprocessing

Data preprocessing is the essential process of cleaning, transforming, and organizing raw data into a clean, usable format for machine learning algorithms.

9.2 Model Selection

The following algorithms are used:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)

```
from sklearn.model_selection import train_test_split

# X = Features (all columns except target)
X = df.drop("isFraud", axis=1)

# y = Target column
y = df["isFraud"]

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("X_train shape:", X_train.shape)
print("X_test shape :", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape :", y_test.shape)
```

```
... X_train shape: (1110298, 8)
X_test shape : (277575, 8)
y_train shape: (1110298,)
y_test shape : (277575,)
```

9.3 Model Training

The dataset is divided into training and testing sets. Models are trained using the training dataset and validated using the testing dataset.

```
> v
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# 1) Create Random Forest model
rf_model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)

# 2) Train the model
rf_model.fit(X_train, y_train)

# 3) Predict on test data
y_pred_rf = rf_model.predict(X_test)

# 4) Accuracy
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", rf_accuracy)

# 5) Confusion Matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
print("\nConfusion Matrix:\n", cm_rf)

# 6) Confusion Matrix Plot
plt.figure(figsize=(6,4))
sns.heatmap(cm_rf, annot=True, fmt="d", cmap="Blues")
plt.title("Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# 7) Classification Report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_rf))

... Random Forest Accuracy: 0.9997622264252904

Confusion Matrix:
[[277260  1]
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

Click to add a breakpoint n Tree model
dt_model = DecisionTreeClassifier(
    criterion='gini', # or 'entropy'
    max_depth=None, # you can limit depth like max_depth=5 to avoid overfitting
    random_state=42
)

# 2) Train the model
dt_model.fit(X_train, y_train)

# 3) Predict on test data
y_pred_dt = dt_model.predict(X_test)

# 4) Accuracy
dt_accuracy = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Accuracy:", dt_accuracy)

# 5) Confusion Matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
print("\nConfusion Matrix:\n", cm_dt)

# 6) Confusion Matrix Plot
plt.figure(figsize=(6,4))
sns.heatmap(cm_dt, annot=True, fmt="d", cmap="Greens")
plt.title("Decision Tree - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# 7) Classification Report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_dt))

... Decision Tree Accuracy: 0.9996037107088175

Confusion Matrix:
[[277218  43]
```

9.4 Model Evaluation

Performance is evaluated using:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

10. Implementation

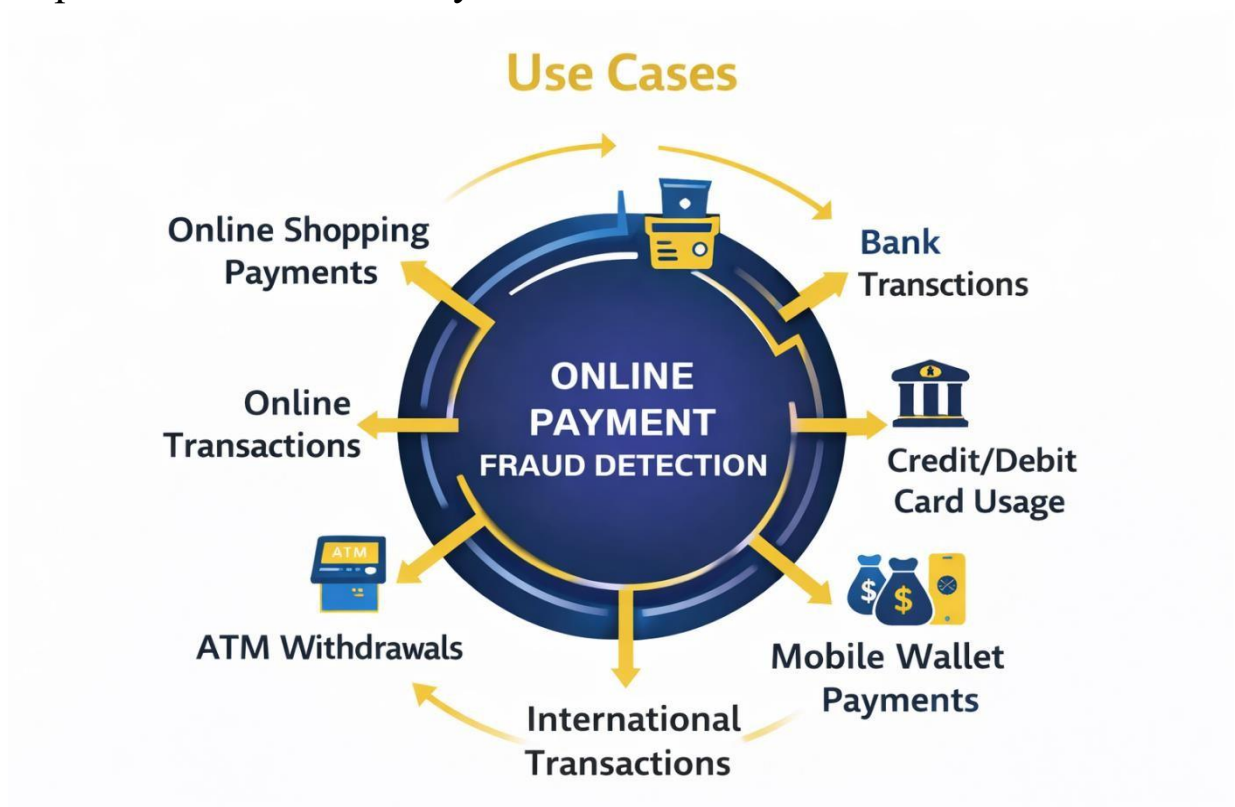
The system is implemented using Python and machine learning libraries. A web application is developed using Flask to allow users to enter transaction details and get predictions.

Technologies Used:

- Python
- Pandas, NumPy
- Scikit-learn

11. Advantages

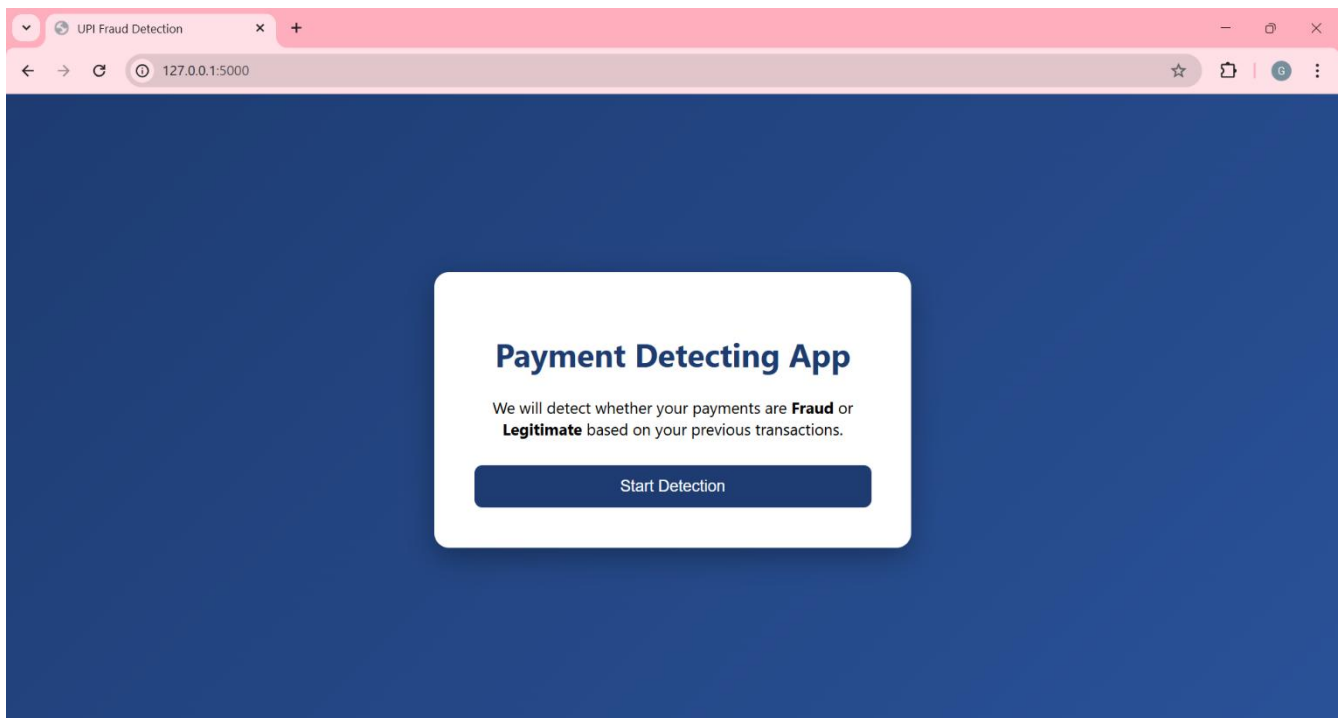
- Automated fraud detection
- High accuracy
- Reduces manual monitoring
- Improves financial security

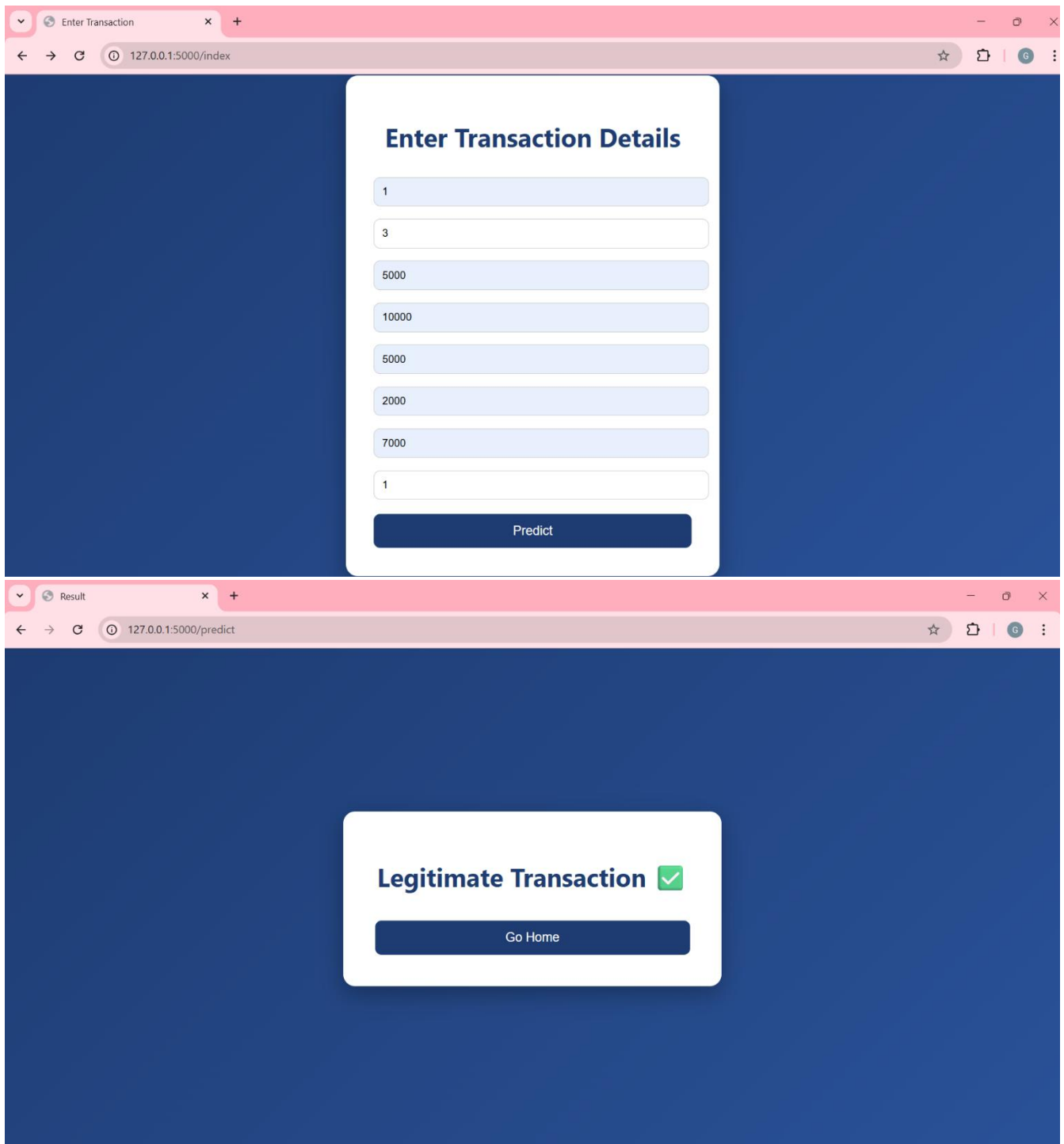


12. Results

The trained models are evaluated on test data. Random Forest and SVM show better performance compared to other models. The system is able to detect most fraudulent transactions with high accuracy. Graphs and confusion matrices are used to analyze results.

13. Output Screenshots





14. Conclusion

This project demonstrates how machine learning can be effectively used to detect online payment fraud. By analyzing transaction data, the system helps in identifying suspicious activities and preventing

financial losses. With further improvements, the system can be deployed on a larger scale.

15. Future Enhancements

- Use deep learning models
- Real-time fraud monitoring
- Integration with banking systems
- Mobile application support
- AI-based adaptive learning
-

16. Appendix

GitHub Repository Link:

Dataset Link:

<https://www.kaggle.com/datasets/rupakroy/onlinepayments-fraud-detection-dataset>