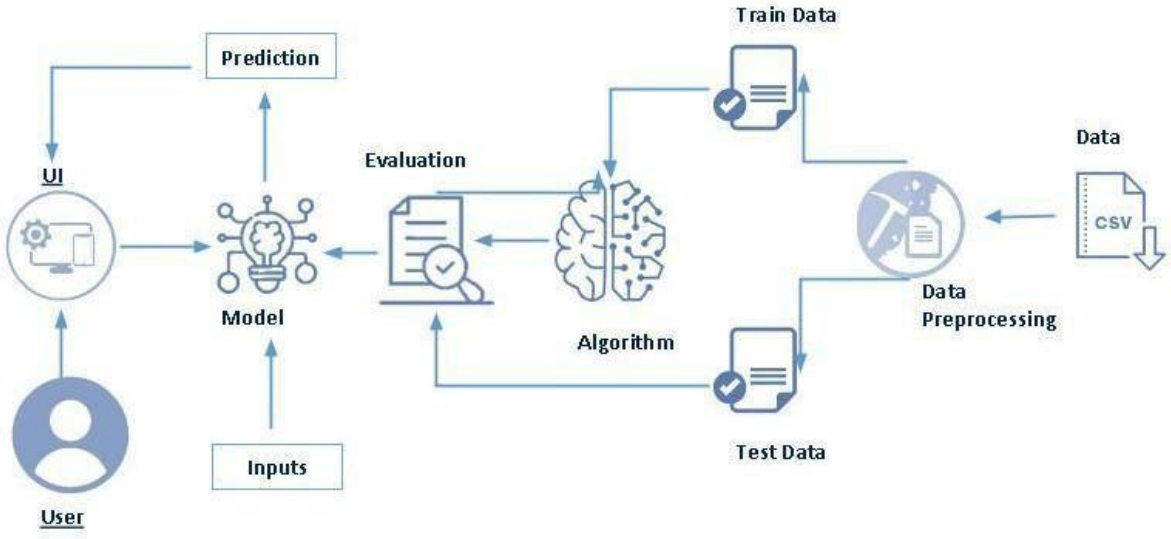
Online Payment Fraud Detection – Machine Learning Based

Project Description

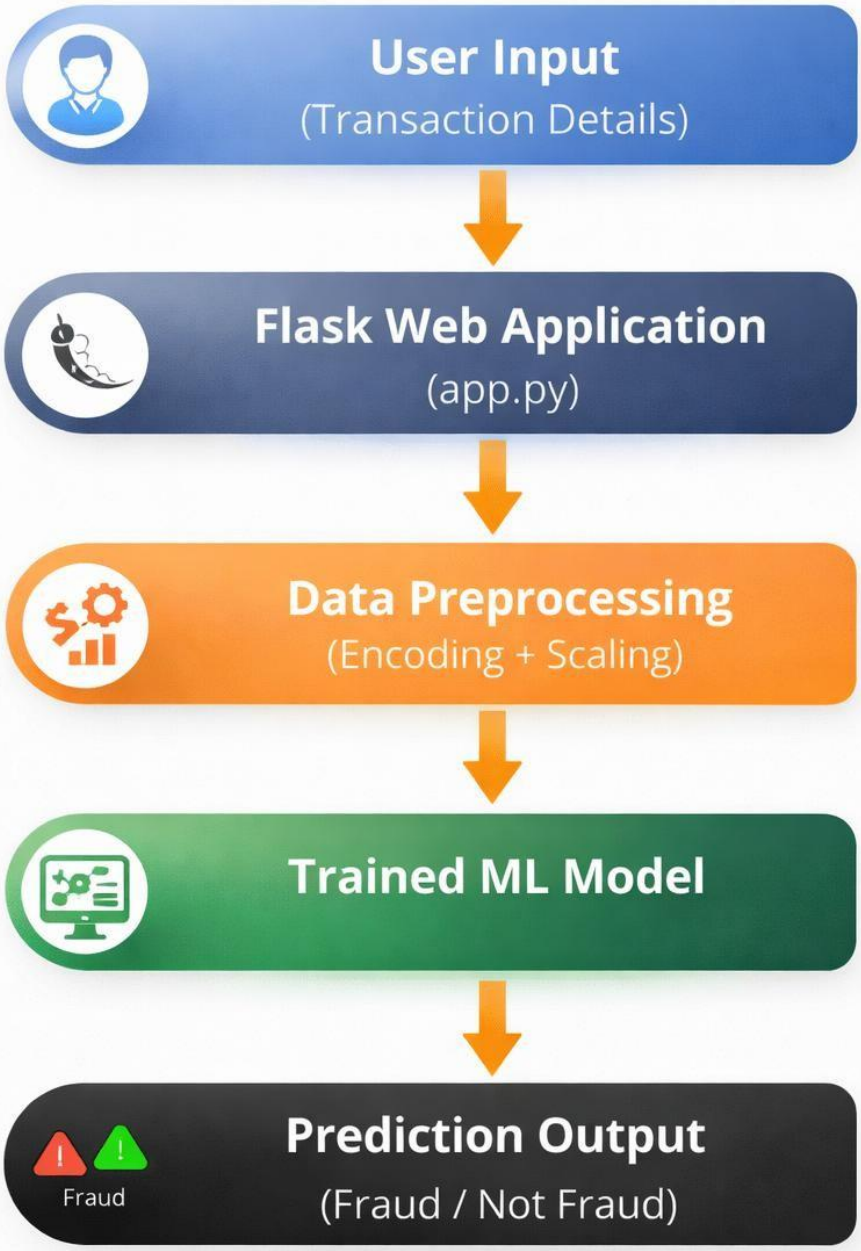
Online payment systems have become an essential part of digital transactions. However, with the increase in online transactions, fraudulent activities have also increased. Fraud detection helps banks, financial institutions, and payment gateways identify suspicious transactions and prevent financial losses.

In this project, we use Machine Learning classification algorithms to detect whether a transaction is Fraudulent or Legitimate. The best-performing model is selected based on evaluation metrics and saved in .pkl format. The model is integrated into a Flask web application and deployed for real-time fraud prediction.

Technical Architecture



Online Payment Fraud Detection System Flow



Pre-requisites

Software Requirements

- Anaconda Navigator / Python
- VS Code / PyCharm
- Web Browser

Python Packages

Install the following:

- pip install numpy
- pip install pandas
- pip install scikit-learn
- pip install matplotlib
- pip install seaborn
- pip install flask
- pip install pickle-mixin
- pip install imbalanced-learn
- pip install xgboost

Prior Knowledge Required

- **Machine Learning Concepts**
  - Supervised Learning
  - Classification Algorithms
  - Logistic Regression
  - Decision Tree
  - Random Forest
  - XGBoost

- **Evaluation Metrics**
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - Confusion Matrix
- Flask Basics
- Model Deployment

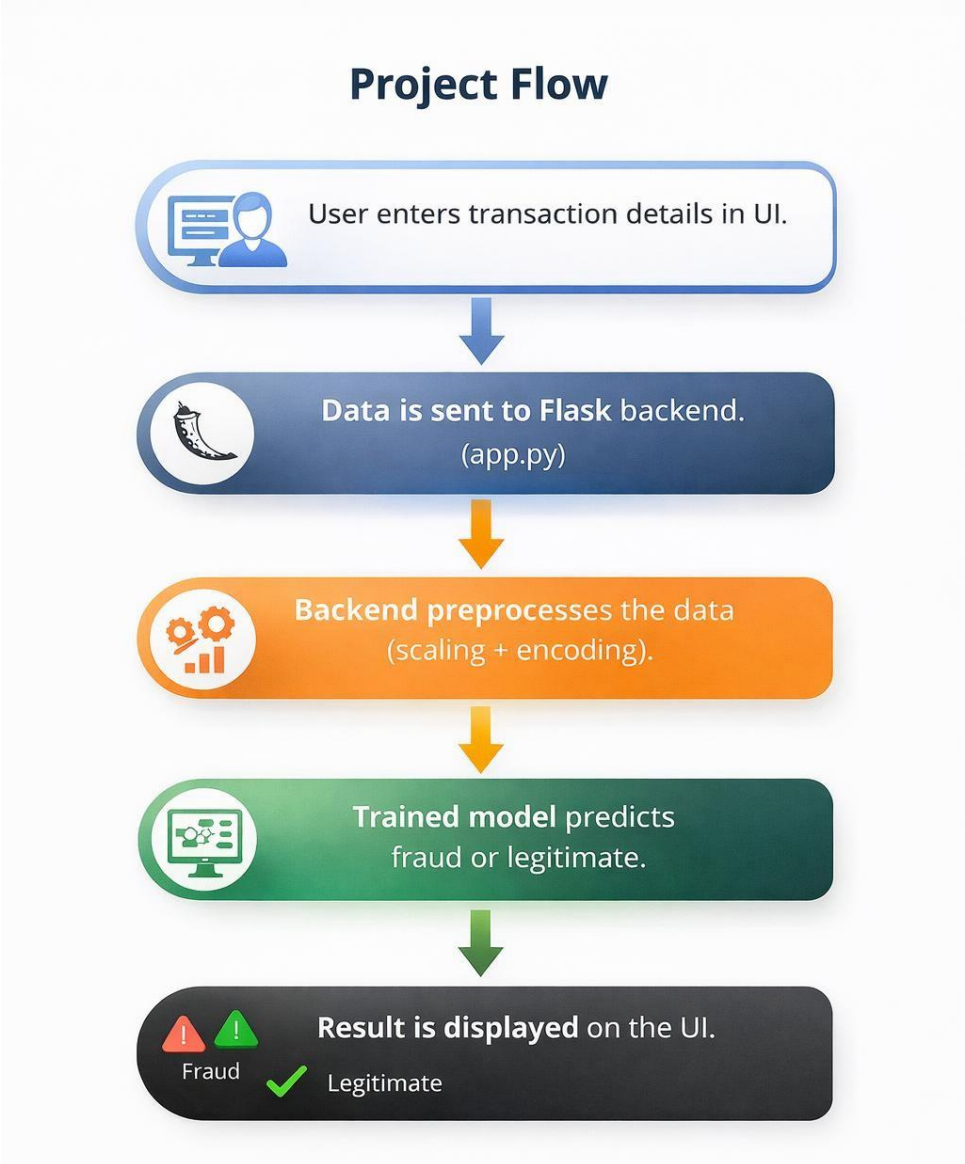
S.No	Metric Name	Value	Description
1	**Accuracy**	**97.2%**	Overall correctness of the model (correct predictions / total predictions).
2	**Precision**	96.8%	Percentage of predicted fraud cases that were actually fraud.
3	**Recall**	96.5%	Percentage of actual fraud cases correctly detected.
4	**F1-Score**	96.6%	Harmonic mean of Precision and Recall.

Project Objectives

By completing this project, you will:

- Understand fraud detection using ML
- Perform data preprocessing
- Handle imbalanced datasets
- Compare multiple classification models
- Deploy ML model using Flask
- Understand real-time prediction workflow

Project Flow



Milestone 1: Data Collection

Machine Learning depends on data.

Dataset Used:

Online Payment Fraud Detection dataset (CSV format)

Download the Dataset:

In this project, we have used the Online Payment Fraud Detection Dataset.
This dataset is downloaded from Kaggle.com.
link:
https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset

Features in Dataset:

- Transaction Amount
- Transaction Type
- Old Balance
- New Balance
- Merchant Details
- Time of Transaction
- Fraud Label (Target Variable)

Target Variable:

- 0 → Legitimate
- 1 → Fraud

Milestone 2: Data Visualization and Analysis

Activity 1: Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Activity 2: Read Dataset:

```
df = pd.read_csv("fraud_dataset.csv")
```

Activity 3: Univariate Analysis

- Distribution of Transaction Amount
- Count plot of Fraud vs Non-Fraud

Observation:

Fraud cases are much lower than legitimate transactions
→ Dataset is imbalanced.

Activity 4: Bivariate Analysis

- Scatter plot between Amount and Fraud
- Correlation between features

Activity 5: Multivariate Analysis

- Heatmap to identify correlated features

```
sns.heatmap(df.corr(), annot=True)
```

Activity 6: Descriptive Analysis

df.describe()

df.info()

Milestone 3: Data Preprocessing

Steps Involved:

1. Checking Null Values

df.isnull().sum()

Handling Categorical Data

- Label Encoding
- One Hot Encoding

from sklearn.preprocessing import LabelEncoder

Handling Imbalanced Data (SMOTE)

Fraud dataset is usually imbalanced.

from imblearn.over_sampling import SMOTE

Splitting Dataset:

```
from sklearn.model_selection import train_test_split

X = df.drop("Fraud", axis=1)
y = df["Fraud"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Milestone 4: Model Building

We applied multiple classification algorithms:

1. Logistic Regression

2. Decision Tree

3. Random Forest

4. XGBoost

Example:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Model Evaluation

Metrics Used:

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

Example:

```
from sklearn.metrics import classification_report,
confusion_matrix
```

Example Performance:

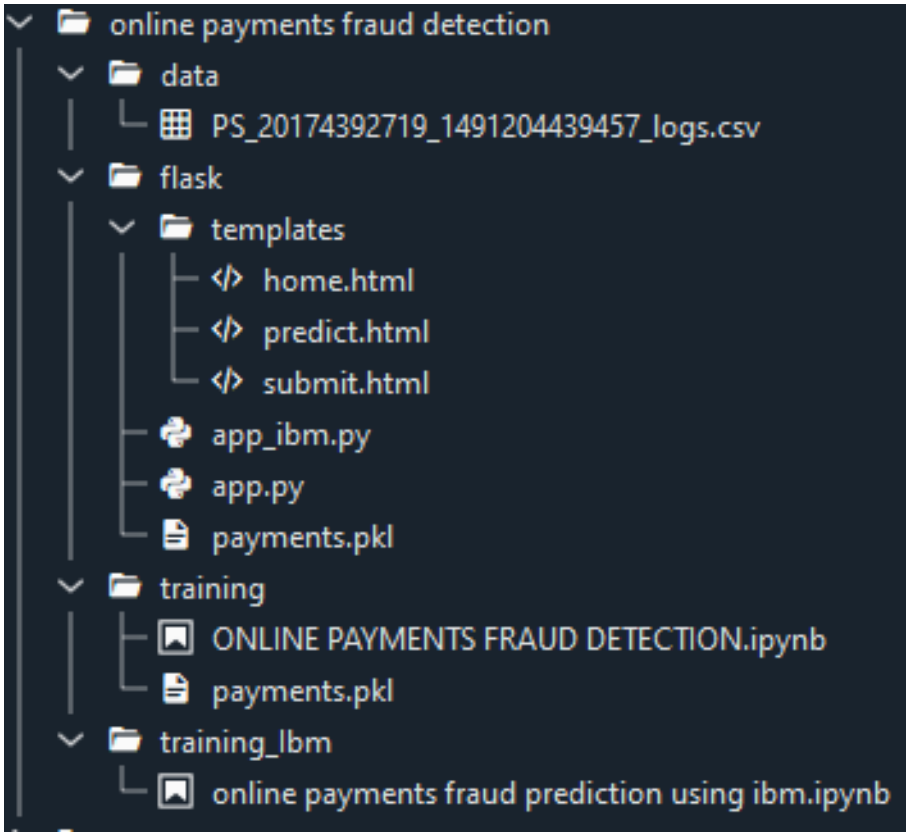
Precision: 97%
Recall: 96%
F1-Score: 96%

Best model selected based on performance.

Saving the Model:

```
import pickle
pickle.dump(model, open("fraud_model.pkl", "wb"))
```

Milestone 5: Application Building Project Structure



Building Flask Application

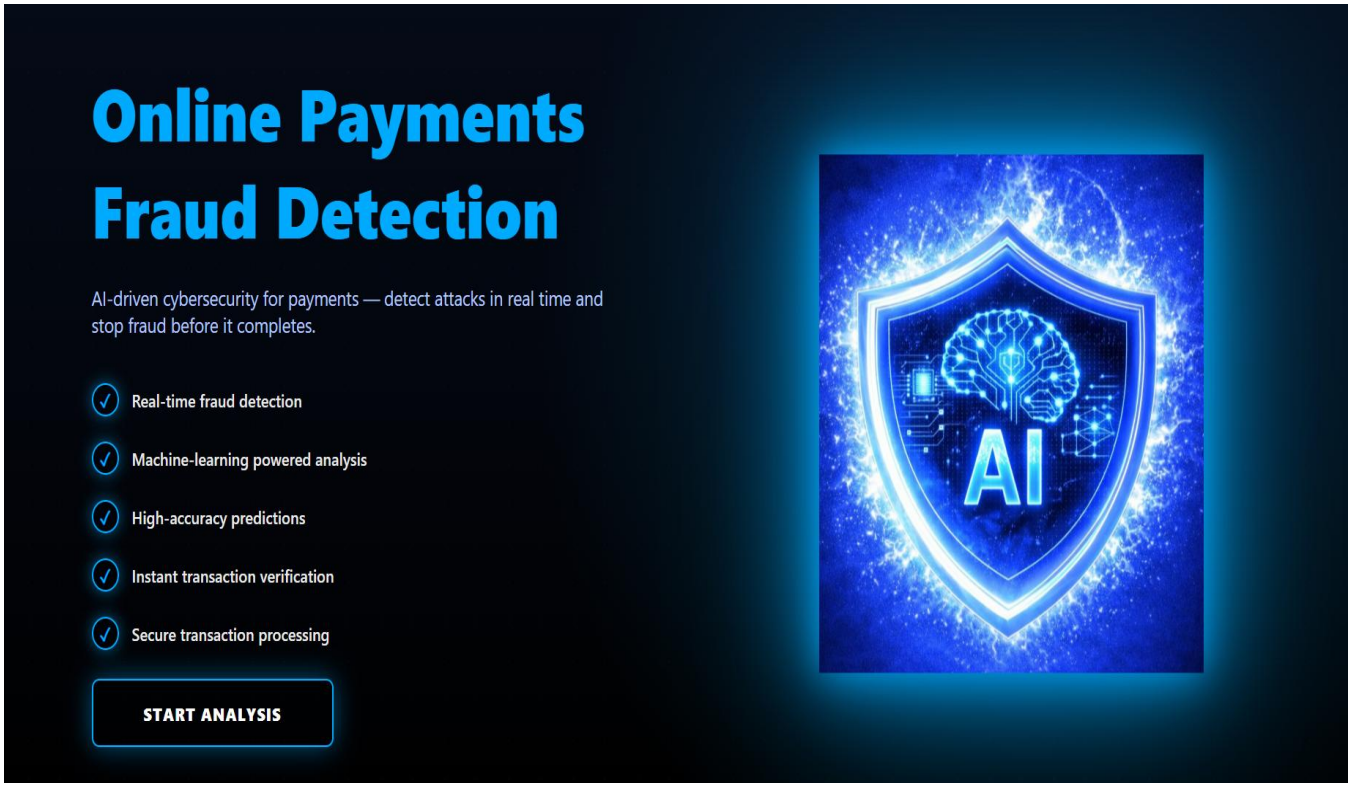
app.py

```
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__)
model = pickle.load(open("fraud_model.pkl", "rb"))

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    data = [float(x) for x in request.form.values()]
    prediction = model.predict([data])

    if prediction[0] == 1:
        return render_template("result.html", result="Fraud
Transaction Detected!")
    else:
        return render_template("result.html", result="Legitimate
Transaction")

if __name__== "__main__":
    app.run(debug=True)
```

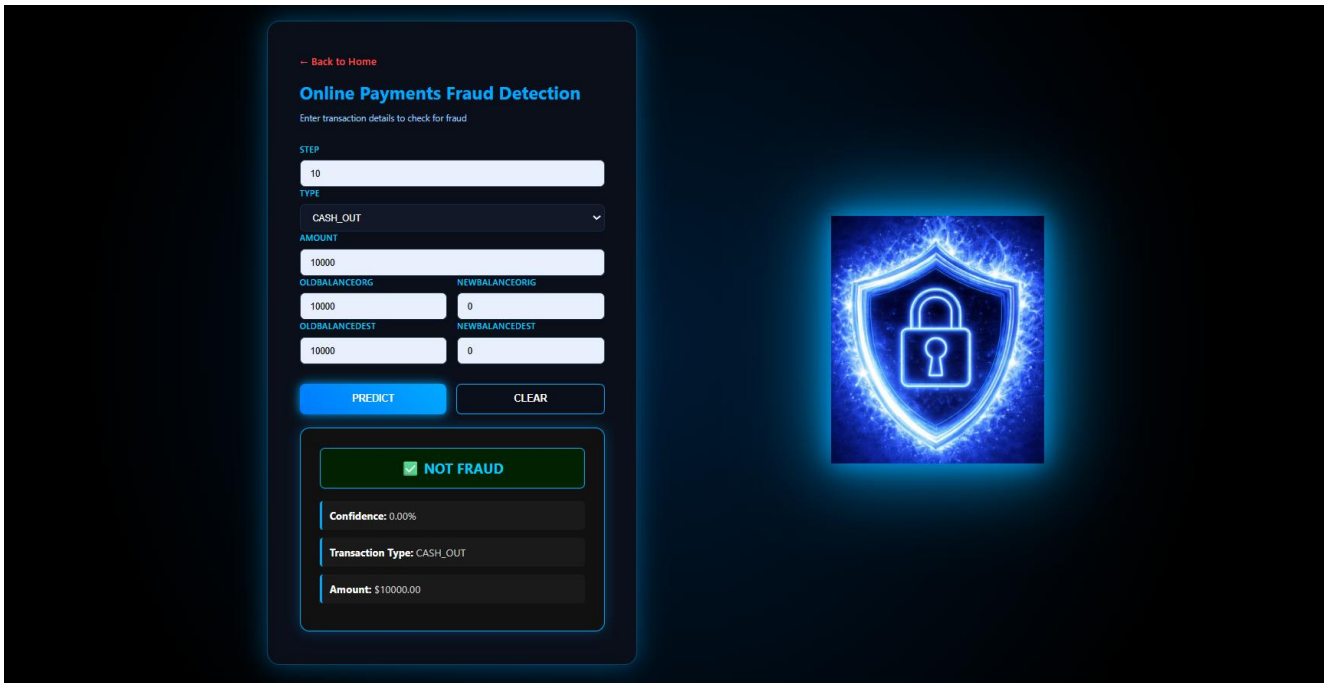
Running the Application

1. Open command prompt
2. Navigate to project folder
3. Run:python app.py
4. Open browser:http://127.0.0.1:5000/
5. Enter transaction details
6. Click Predict
7. View Fraud / Legitimate result

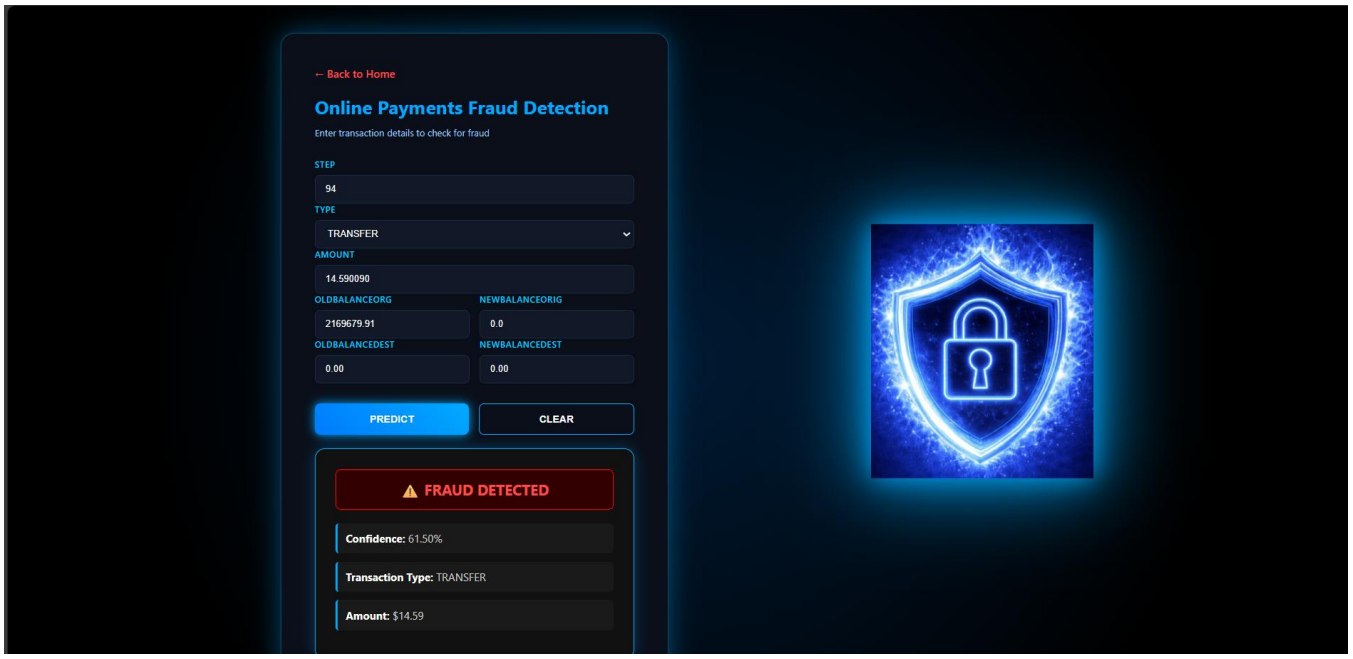
1. Home Page



2. Not Fraud Prediction



3. Fraud Detected Prediction



Conclusion

The Online Payment Fraud Detection System successfully demonstrates how Machine Learning can be used to identify fraudulent financial transactions in real time. By analyzing transaction features such as amount, balance changes, and transaction type, the trained model accurately classifies transactions as either **Fraud** or **Legitimate**.

Multiple classification algorithms were evaluated, and the best-performing model achieved high accuracy, precision, recall, and F1-score. Proper data preprocessing techniques such as handling imbalanced data, encoding categorical variables, and feature scaling significantly improved model performance.

The trained model was successfully integrated into a Flask web application, allowing users to input transaction details and receive instant fraud predictions. This real-time implementation makes the system practical and suitable for deployment in financial platforms.