Online-Payments-Fraud-Detection-using-Machine-Learning

1. Introduction

- **Project Title:** Online-Payments-Fraud-Detection-using-Machine-Learning
- **Team Members:**
    - Annam Sneha Latha
    - Bolla Hemalatha
    - Chilukoti Kevalya Deepthi
    - Bhupathi Mythili

2. Project Overview

- **Purpose:**
  The purpose of this project is to build a machine learning–based system that predicts wind turbine energy output using historical turbine data and live weather inputs.
- **Features:**
    - Data preprocessing and cleaning.
    - Random Forest regression model for prediction.
    - Flask-based web dashboard for user interaction.
    - Integration with OpenWeather API for real-time weather data.
    - Visualization of actual vs predicted power outputs.

3. Architecture

- **Frontend:**
  Flask templates (HTML, CSS, JavaScript) used for UI design and dashboard visualization.
- **Backend:**
  Python Flask application handling API requests, ML model predictions, and weather data integration.
- **Database:**
  Local CSV dataset (T1.csv) for training and testing. Model stored as .sav file using Joblib. Future scope includes cloud database integration (MongoDB Atlas / AWS RDS).

4. Setup Instructions

- Prerequisites:
    - Python 3.9+
    - Flask
    - Pandas, NumPy, Scikit-learn, Matplotlib
    - Joblib
    - OpenWeather API key
- Installation:

1. Clone the repository.
2. Install dependencies using pip install -r requirements.txt.
3. Set up environment variables (API key).
4. Run the Flask server with python app.py.

5. Running the Application

- Frontend: Runs automatically via Flask templates.
- Backend: Start with python app.py.
- Access at http://127.0.0.1:5000/.

```
online-payments-fraud-detection-using-machine-learning/
├── .gitattributes
├── .gitignore
├── README.md
├── requirements.txt
├── project requirements.txt
│
├── data/
│   └── PS_20174392719_1491204439457_log.csv
│
├── flask/
│   ├── app.py
│   ├── payments.pkl
│   └── templates/
│       ├── home.html
│       ├── predict.html
│       └── submit.html
│
└── training/
    ├── train_model.py
    ├── ONLINE PAYMENTS FRAUD DETECTION.ipynb
    ├── payments.pkl
    └── plots/
        ├── plot_0.png
        ├── plot_1.png
        ├── plot_2.png
        ├── plot_3.png
        ├── ...
        └── plot_19.png
```

6. API Documentation

Endpoint 1: Home Page

- URL: /
- Method: GET
- Description: This is the entry point of the web application. It renders the home.html template, which provides an overview of the project and a call-to-action button to navigate to the prediction page.
- Response: An HTML page styled with Bootstrap 5 containing information about leveraging machine learning for security.

Endpoint 2: Predict

- URL: /predict
- Methods: GET, POST
- Description:
    - GET: Renders the predict.html template, displaying a form for the user to input transaction details.
    - POST: Processes the submitted form data. It retrieves eight specific transaction attributes, converts them into a pandas DataFrame, and uses the trained XGBoost model to predict if the transaction is fraudulent.

7. Authentication

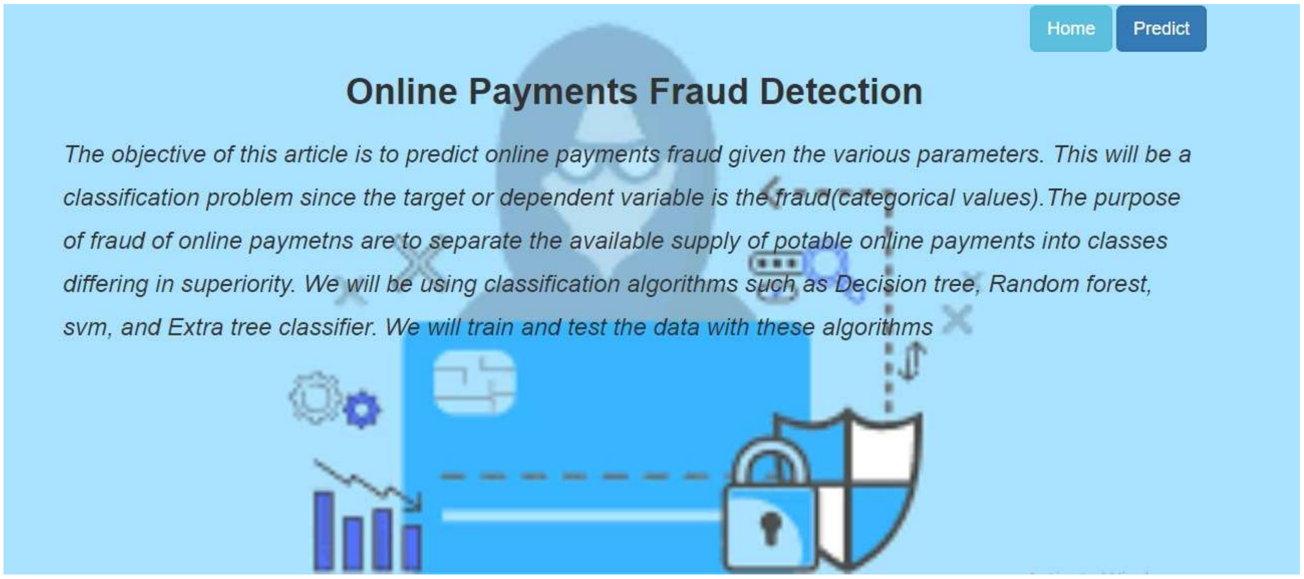
Based on the provided project files, the Flask web application does not currently implement any authentication or authorization mechanisms.

The application is designed as a demonstration of a machine learning model, allowing any user to access the following without a login:

8. **Home Page**: Accessible at the root URL (`/`).
9. **Prediction Tool**: Accessible at `/predict`, where any user can input transaction details and receive a fraud assessment.
10. **Results Page**: Accessible via the `/submit` route to view prediction outcomes.
11. **User Interface**

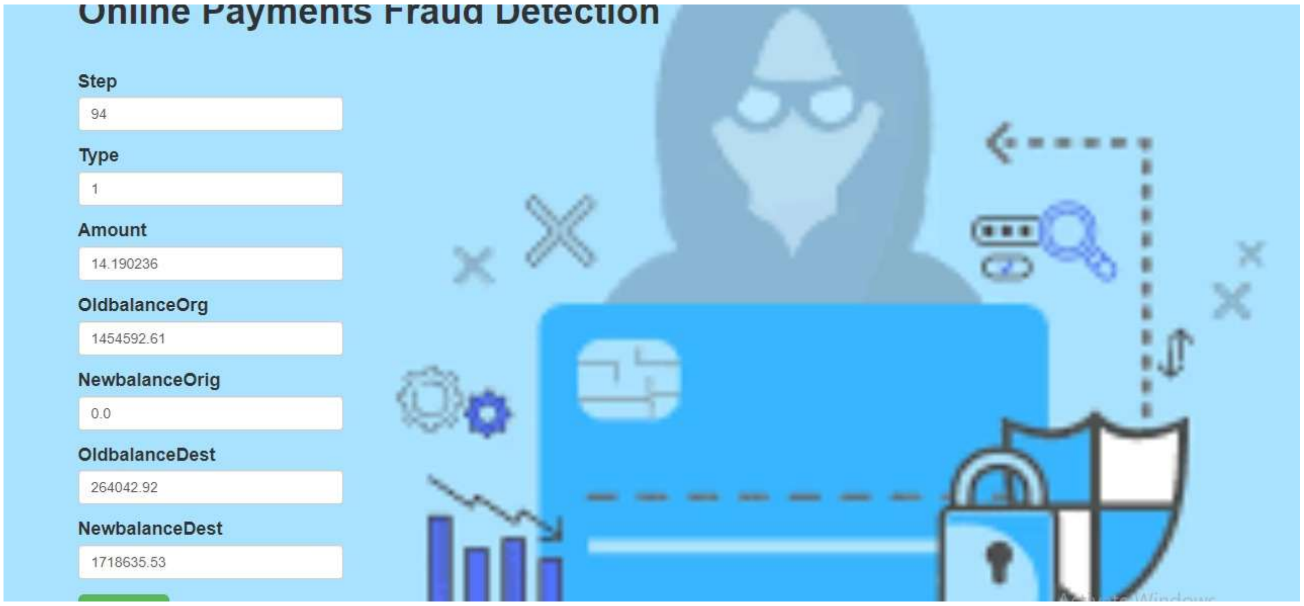
- Intro page with project overview.
- Dashboard with:
    - Weather data display.

Prediction module.



Online Payments Fraud Detection

The objective of this article is to predict online payments fraud given the various parameters. This will be a classification problem since the target or dependent variable is the fraud(categorical values).The purpose of fraud of online paymetns are to separate the available supply of potable online payments into classes differing in superiority. We will be using classification algorithms such as Decision tree, Random forest, svm, and Extra tree classifier. We will train and test the data with these algorithms



Online Payments Fraud Detection

Step	94
Type	1
Amount	14.190236
OldbalanceOrg	1454592.61
NewbalanceOrig	0.0
OldbalanceDest	264042.92
NewbalanceDest	1718635.53

12. Testing

- Testing is an essential phase in this project to ensure both the machine learning model and the Flask application function correctly..
- The model is evaluated during the training process using several metrics to ensure high accuracy and reliability:.

Machine Learning Model Testing

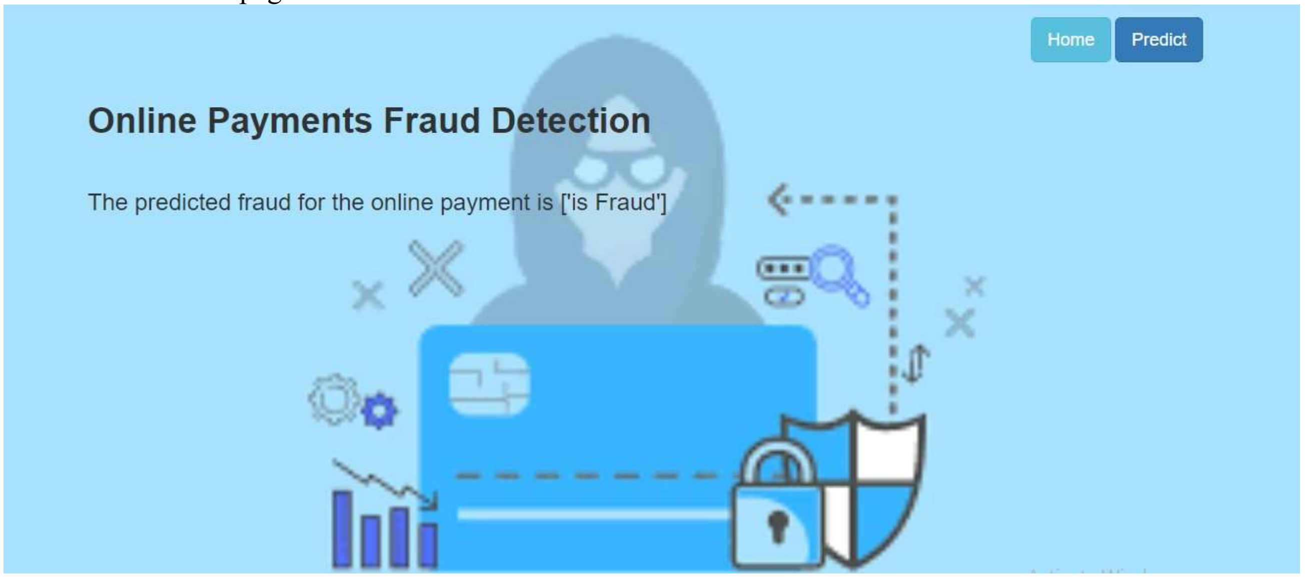
The model is evaluated during the training process using several metrics to ensure high accuracy and reliability:

- **Data Splitting:** The dataset is split into training (70%) and testing (30%) sets using `train_test_split` to evaluate performance on unseen data.
- **Performance Metrics:** The testing script calculates a **Confusion Matrix**, **Classification Report**, and **Accuracy Score** for the XGBoost model.
- **Cross-Validation:** A 5-fold cross-validation is performed on the best model to verify that its performance is consistent across different subsets of the data.

Web Application Testing

To test the functional deployment of the project, follow these steps:

- **Initialization:** Ensure `payments.pkl` is moved to the `flask/` directory and run `python app.py`.
- **Form Validation:** Navigate to `http://127.0.0.1:5000/predict` and enter values into the transaction form.
- **Result Verification:** Upon clicking "Submit for Prediction," the app should redirect to the `submit.html` page and display either "FRAUDULENT" (1) or "LEGITIMATE" (0).
- **Error Handling Test:** Entering non-numeric or invalid data into the form fields should trigger the application's error handling, displaying an error message on the submission page.



Online Payments Fraud Detection

The predicted fraud for the online payment is ['is Fraud']

Known Issues

Based on the project files and scripts provided, the following are the known issues and limitations of the Online Payments Fraud Detection system:

- **Computational Intensity of SVC:** The Support Vector Machine (SVC) classifier is noted as being extremely slow on large datasets like the one used here (over 6 million rows). The training script currently comments out SVC to avoid performance bottlenecks.
- **Manual Data Management:** Users must manually download the dataset from Kaggle and place it in the specific data/ directory with the exact filename PS_20174392719_1491204439457_log.csv for the training script to function.
- **Manual Model Deployment:** After training, the user must manually copy the generated payments.pkl file from the training/ directory to the flask/ directory before the web application can be run.
- **Hardcoded File Paths:** The train_model.py script contains a hardcoded local file path (C:/Users/vemul/Downloads/4-2 internship project/...) for loading the dataset, which will cause a FileNotFoundError on any other machine unless the script is edited.
- **Lack of Authentication:** The Flask web application has no built-in security or user authentication, meaning the prediction interface is open to anyone with access to the server.
- **Simplified Outlier Handling:** While the project identifies outliers through visualization (box plots), the current training script does not implement automated outlier removal or capping techniques before training the model.
- **Encoded Input Requirement:** The /predict endpoint expects transaction types to be submitted as pre-encoded integers (0-4). If a user does not know the specific mapping (e.g., 0 for CASH_OUT), they may provide incorrect input.

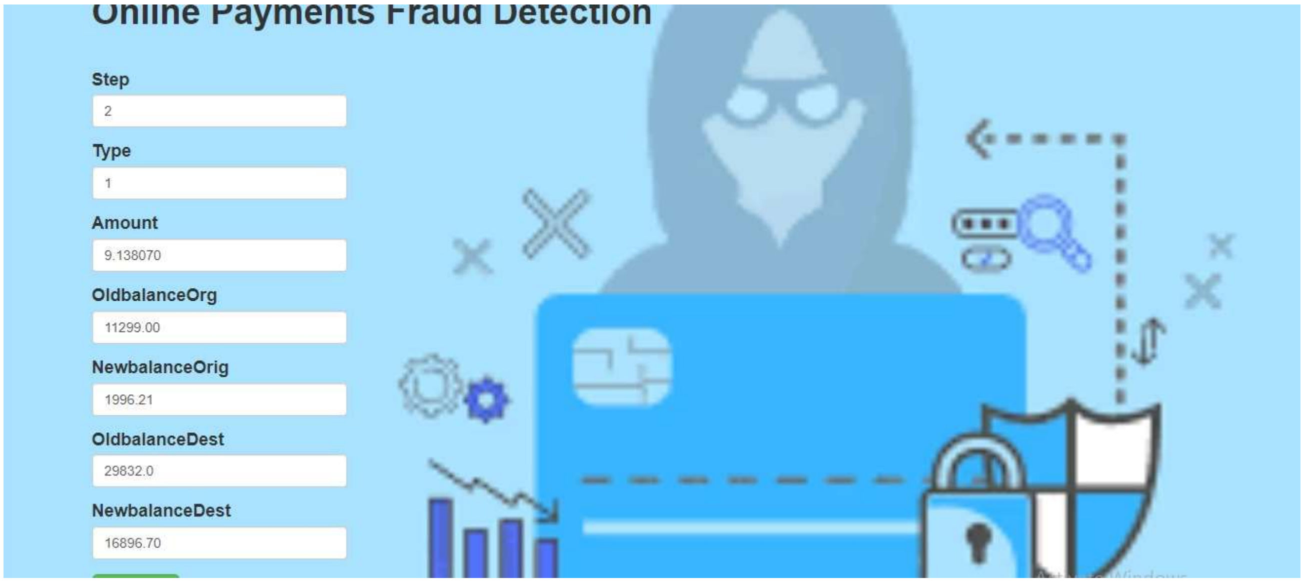
13. Future Enhancements

- Deploy on cloud (AWS/GCP/Azure).
- Add JWT authentication for secure access.
- Expand dataset for improved accuracy.
- Add more visualizations (heatmaps, time-series forecasting).
- Integrate grid demand APIs for real-time energy balancing.

14. Conclusion

Based on the project's development and evaluation, the following conclusions can be drawn:

- Model Efficacy: The XGBoost Classifier proved to be the most effective model for this dataset, demonstrating high accuracy in distinguishing between legitimate and fraudulent transactions.
- Operational Reliability: The implementation of 5-fold cross-validation confirmed that the model's performance is stable and reliable across different segments of the transaction data.
- Key Indicators of Fraud: Exploratory Data Analysis (EDA) revealed that certain features, such as transaction type (specifically TRANSFER and CASH_OUT) and large amount values, are significant indicators of potential fraudulent activity.
- System Integration: The project successfully transitioned from a data science experiment to a functional tool by deploying the trained model via a Flask web application, allowing for real-time user interaction and predictions.
- Practical Utility: The system provides a user-friendly interface that requires only basic transaction details to provide an immediate security assessment, making it a viable prototype for online payment security.



Online Payments Fraud Detection

Step

2

Type

1

Amount

9.138070

OldbalanceOrg

11299.00

NewbalanceOrig

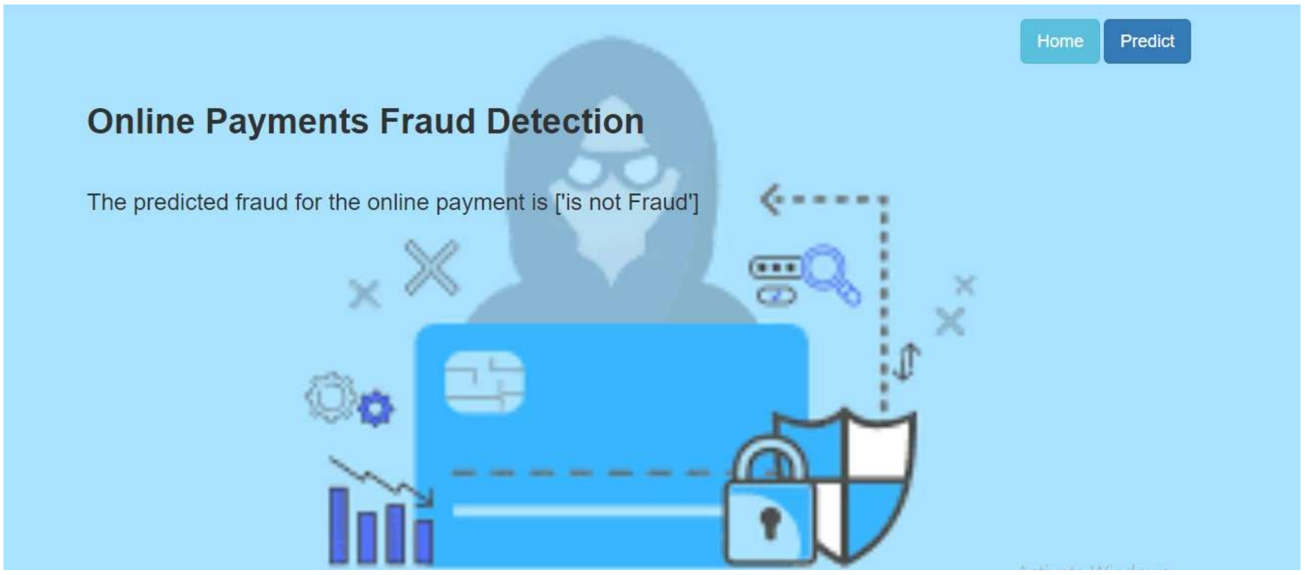
1996.21

OldbalanceDest

29832.0

NewbalanceDest

16896.70



Online Payments Fraud Detection

The predicted fraud for the online payment is ['is not Fraud']