DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JNTUH COLLEGE OF ENGINEERING, SCIENCE AND TECHNOLOGY HYDERABAD-500085

INDUSTRIAL ORIENTED MINIPROJECT REPORT ON

ONLINE PAYMENT FRAUD DETECTION

Submitted by

Bajjuri Srivathsav(22011A6603)
Nalam Ananya Sri (22011A6619)
Kommula Nishitha(22011A6620)
Gadipelli Shivani Aaradhya(22011A6624)

In partial fulfillment for the award of the degree

of

Bachelor of Engineering

in

Computer Science and Engineering

JNTUH COLLEGE OF ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the mini project report titled “ONLINE PAYMENT FRAUD DETECTION” is the bonafide work of

Bajjuri Srivathsav(22011A6603)
Nalam Ananya Sri (22011A6619)
Kommula Nishitha(22011A6620)
Gadipelli Shivani Aaradhya(22011A6624)

who carried out the project work under my supervision.

<<SIGNATURE OF HOD>>
Dr K P Supreethi

HEAD OF THE DEPARTMENT
Professor and Head of the Department

Computer Science and Engineering

<<SIGNATURE OF SUPERVISOR>>
Mr Naresh Kumar A

SUPERVISOR
Assistant Professor
Computer Science and Engineering

Submitted for the Autonomous End Semester Examination Mini Project viva-voce held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to everyone who contributed to the successful completion of this mini project titled “Online Payment Fraud Detection.”

I extend my sincere thanks to my supervisor Mr. Naresh Kumar A for his continuous guidance, valuable suggestions, and support throughout the project.

I am also thankful to Dr. K P Supreethi, Head of the Department of Computer Science and Engineering, JNTUH College of Engineering, for providing necessary facilities.

Last but not least, I thank my friends and family for their constant encouragement.

A Machine Learning-Based Approach for Detecting Online Payment Fraud: Design and Factors

ABSTRACT

With the rapid advancement of digital payment platforms, the risk of online payment fraud has become a significant concern for banking and e-commerce sectors. This project, titled *"Online Payment Fraud Detection"*, focuses on designing a machine learning-based system to identify potentially fraudulent transactions with high accuracy. A real-world dataset containing various transaction attributes is preprocessed and analyzed to identify key patterns and features indicative of fraudulent behavior. Supervised learning algorithms, particularly Random Forest and Logistic Regression, are employed to train predictive models capable of distinguishing between legitimate and suspicious transactions.

To enhance usability and real-time applicability, the trained model is integrated into a user-friendly Web application. This application allows users to input transaction details and receive instant predictions regarding the likelihood of fraud. The model's performance is evaluated using accuracy, precision, recall, and F1-score, demonstrating strong predictive capability and reliability.

The proposed system highlights the importance of feature selection, model design, and real-time integration as critical factors in building effective fraud detection solutions. This approach can significantly aid financial institutions in minimizing fraud-related losses and enhancing the security of online payment systems.

Table of Contents

1.INTRODUCTION

1.1 Problem Statement

With the exponential growth of online financial services and digital transactions, fraudsters are increasingly targeting vulnerabilities in payment systems to carry out unauthorized or deceptive activities. Traditional rule-based systems are often inadequate in adapting to evolving fraudulent patterns, leading to delayed detection or undetected fraud. This project addresses the urgent need for an intelligent, data-driven solution that can detect potentially fraudulent transactions in real-time with high accuracy. The problem lies in designing a model that can effectively distinguish between legitimate and fraudulent activities using historical transaction data.

1.2 Objectives

The primary objectives of this project are as follows:

- To analyze and preprocess a real-world dataset of online payment transactions for effective model training.
- To implement and compare machine learning models such as Random Forest and Logistic Regression for fraud detection.
- To identify key features that contribute to fraudulent behavior in online transactions.
- To integrate the best-performing model into a web-based application using Flask for real-time prediction.
- To evaluate the system's performance based on metrics such as accuracy, precision, recall, and F1-score.
- To provide a reusable and scalable solution that can be adapted by financial institutions for fraud prevention.

1.3 Scope

The scope of this project, *Online Payment Fraud Detection*, is outlined as follows:

- The study is limited to the detection of fraudulent activities within online payment transactions using supervised machine learning techniques.

- The project utilizes a structured dataset comprising features such as transaction type, transaction amount, and account balances at origin and destination.
- The work involves preprocessing of data, feature selection, model training, and evaluation using classification algorithms including Random Forest and Logistic Regression.
- The trained model is assessed using performance metrics such as accuracy, precision, recall, and F1-score to determine its effectiveness.
- The system is implemented as a web-based application using the Flask framework, providing users with an interface for real-time transaction fraud prediction.
- The solution is designed to be modular, reusable, and scalable for potential adaptation in financial and e-commerce environments.
- The scope does not extend to real-time integration with banking infrastructure or payment gateways.
- Advanced security mechanisms such as blockchain, encryption, or biometric authentication are considered out of scope for this project.

1.4 Requirement Analysis

This section outlines the specific requirements essential for the successful implementation of the *Online Payment Fraud Detection* system. The requirements are categorized into functional, non-functional, software, hardware, and user requirements based on the scope and methodology of the project.

1.4.1 Software Requirements

1. Front-End Requirements

- **HTML**: For creating the structure of the web pages (input forms and result display).
- **CSS/Bootstrap**: For styling the pages and ensuring responsiveness.

2. Back-End & ML Integration

- **Python**: Core language for machine learning and web development.
- **Flask**: Lightweight web framework used to create the web application.

- **Pandas, NumPy**: For data manipulation and preprocessing.
- **Scikit-learn**: For implementing and training ML models (Random Forest, Logistic Regression).
- **Jupyter Notebook / Google Colab**: For model development and experimentation.

3. Development Tools

- **Code Editor**: Visual Studio Code, PyCharm, or Jupyter Notebook.
- **Browser**: Google Chrome, Mozilla Firefox (for UI testing).
- **Version Control**: Git (optional, for collaboration and deployment).

4. Web Server (Local)

- Flask's built-in development server or any WSGI server like Gunicorn for production.

1.4.2 Hardware Requirements

- **Processor**: Dual-core or higher (Intel i5/i7, AMD Ryzen 5 or higher).
- **RAM**: Minimum 8 GB (to support browser, IDE, Python processes, and data handling).
- **Storage**: 256 GB SSD or higher recommended (for faster processing).
- **Operating System**: Windows 10/11, Ubuntu, or macOS (any OS compatible with Python and Flask).
- **Internet Connection**: Required for downloading packages, testing web app, and model deployment.

1.4.3 User Requirements

- **Ease of Use**: The application must be simple to use, requiring no advanced technical knowledge.
- **Real-Time Feedback**: Users should be able to get instant predictions after submitting transaction data.
- **Clarity of Output**: Results must be clearly displayed as "Fraudulent" or "Not Fraudulent".

- **Mobile & Desktop Access**: The web interface must be responsive and accessible on both mobile devices and desktops.
- **Reliability**: The system must consistently give valid predictions for different types of inputs.

1.5 Technologies Used:

Python, Flask, Machine Learning (Scikit-learn), HTML & CSS

The **Online Payment Fraud Detection** project uses a combination of Python programming, machine learning models, and web development technologies to build a functional fraud detection system. The project integrates a trained machine learning model into a simple web application to make it interactive and user-friendly.

- **Python**

Purpose:

- Python is the main programming language used for building the machine learning model and the web server.
- It is used for reading and preprocessing the dataset, training the model, and handling the logic of fraud prediction.

Key Concepts:

- Python libraries such as pandas, numpy, and scikit-learn are used for data analysis and model building.
- The joblib module is used to save and load the trained model.

- **Scikit-learn (Machine Learning Library)**

Purpose:

- Scikit-learn is a Python library used to implement machine learning models like Random Forest and Logistic Regression.

Key Concepts:

- Used for splitting data, training models, and evaluating accuracy.
- Provides tools to measure performance using metrics like accuracy, precision, and recall.

- **Flask (Python Web Framework)**

Purpose:

- Flask is a lightweight Python web framework used to create the web interface of the fraud detection system.
- It connects the machine learning model with a web form that users can interact with.

Key Concepts:

- Flask runs a local web server and listens for user input.
- The prediction is made in real time when users submit transaction details through the form.

- **HTML (HyperText Markup Language)**

Purpose:

- HTML is used to create the structure of the web page (form to input transaction details and display result).

Key Concepts:

- Tags like <form>, <input>, <label>, and <button> are used to collect data from the user.<p> is used to display prediction results on the page.

- **CSS (Cascading Style Sheets)**

Purpose:

- CSS is used to style the HTML page, making the form and results more visually appealing and user-friendly.

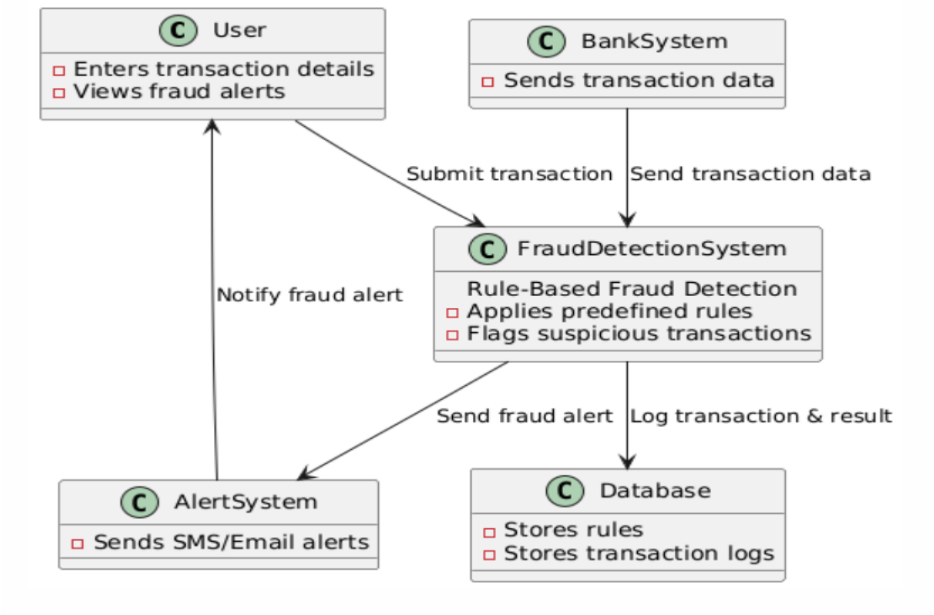
Key Concepts:

- Styles include setting fonts, colors, borders, button designs, and responsive layout.

2.Project Methodology

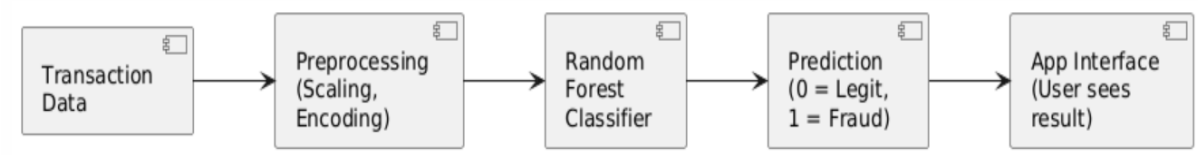
2.1 Existing Method

In traditional fraud detection systems, a **rule-based approach** is commonly used. In this method, domain experts define fixed rules to identify suspicious transactions. For example, rules might include: "If the transaction amount exceeds ₹1,00,000, mark it as fraud," or "If multiple transactions are made within a short time, flag them." While this system is simple and easy to implement, it has many limitations. It cannot detect new or complex fraud patterns, often generates false alerts, and requires manual updates. As fraud techniques evolve, these rule-based systems fail to keep up, making them less effective for real-time fraud detection.



2.2 Proposed Method

The development of the *Online Payment Fraud Detection* system involves a structured approach combining data preprocessing, machine learning model development, and web integration using Flask. The project methodology follows a systematic sequence of phases to ensure an effective and reliable fraud detection system.



1. Problem Definition and Dataset Analysis

- Understand the nature of online payment fraud through research and review of real-world financial datasets.
- Acquire a dataset containing labeled transaction data with relevant features such as amount, type, old/new balances at origin and destination, etc.

✓ **Dataset Used**

The dataset used is a **credit card transaction dataset** containing both fraudulent and legitimate transactions. It includes features such as:

- A target column Class:
    - 0 → Legitimate transaction
    - 1 → Fraudulent transaction

It is commonly used for binary classification tasks in fraud detection. The dataset is highly imbalanced, making it ideal for testing techniques like Random Forest, SMOTE, and cross-validation.

Source: Kaggle – Credit Card Fraud Detection Datase

2. Data Preprocessing

- Perform data cleaning to handle missing, null, or inconsistent values.
- Convert categorical features (e.g., transaction type) into numerical values using encoding techniques.
- Normalize or scale data where necessary to improve model performance.
- Split the dataset into training and testing sets to evaluate the model effectively.

3. Model Development

- Select suitable supervised machine learning algorithms, including Random Forest and Logistic Regression, for binary classification (fraud vs. not fraud).
- Train models using the training dataset and evaluate using metrics such as accuracy, precision, recall, and F1-score.
- Perform hyperparameter tuning and cross-validation to improve prediction reliability.

4. System Design and Integration

- Design a responsive web interface using HTML, CSS, and Bootstrap for user interaction.
- Integrate the trained machine learning model into a Flask web application.
- Set up form-based input for transaction features and return model prediction in real time upon submission.

5. Testing

- Conduct unit testing for individual functions such as data preprocessing and prediction handling.
- Perform integration testing to ensure seamless interaction between frontend, backend, and ML model.
- Test the web application for responsiveness across different browsers and devices.
- Validate output accuracy and user experience through user acceptance testing (UAT).

6. Deployment

- Deploy the Flask application locally for demonstration purposes.
- Ensure security of input data and system stability during prediction requests.

7. Maintenance and Future Enhancements

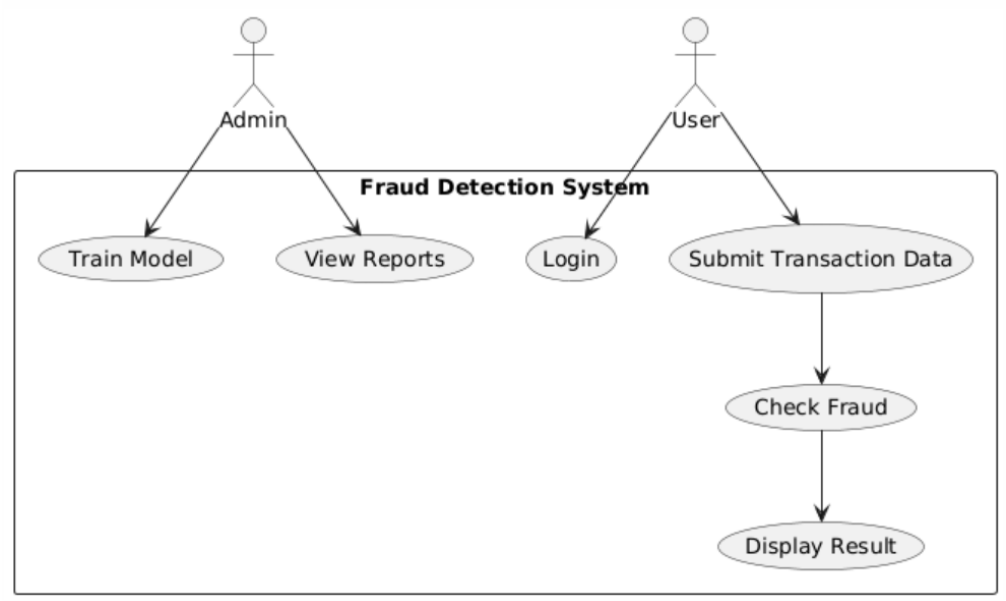
- Maintain code modularity for easy updates and improvements.
- Plan for future enhancements
    - Use of advanced models like XGBoost or deep learning for higher accuracy.

3. Design

Use Case Diagram

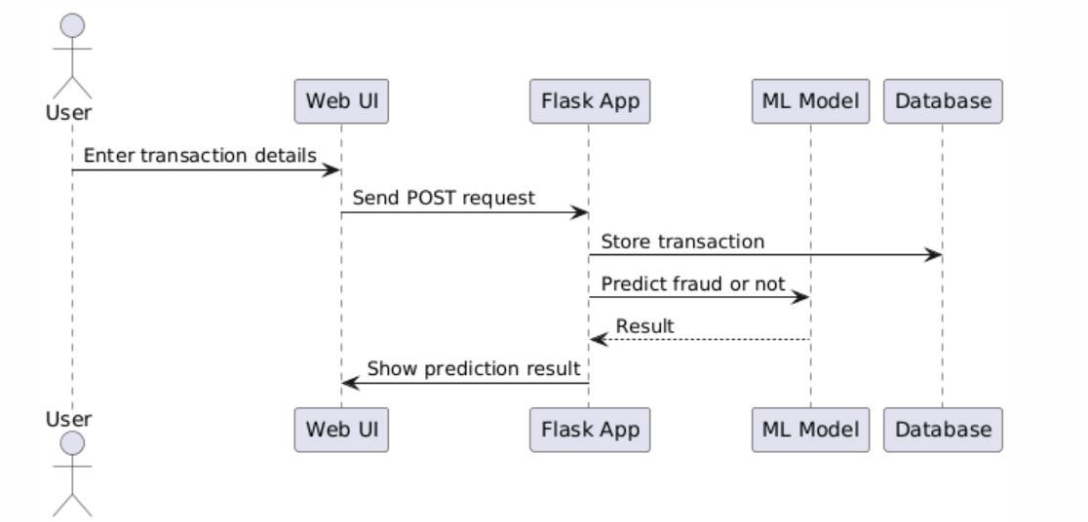
The **Use Case Diagram** in this project represents the interactions between two main actors — **User** and **Admin** — and the Online Payment Fraud Detection System.

- The **User** can:
  - Enter transaction details
  - Submit the transaction for prediction
  - View whether the transaction is **Fraudulent** or **Not Fraudulent**
- The **Admin** can:
  - Manage the dataset
  - Monitor prediction performance
  - Update or retrain the machine learning model if needed



Sequence Diagram

- Models the time-ordered interaction between components. Highlights object communication step-by-step.
- Shows the flow of actions(User inputs data → Data is preprocessed → Model makes prediction → Result is returned)



4. Implementation

The implementation of the *Online Payment Fraud Detection* system was carried out in sequential phases as outlined below:

• Setting Up the Environment

- Initialized a Python development environment with required libraries such as Pandas, NumPy, Scikit-learn, and Flask.
- Prepared the dataset and performed exploratory data analysis to understand fraud patterns.

• Data Preprocessing and Feature Engineering

- Encoded categorical transaction types and engineered useful features such as "difference in balance" to enhance model input quality.
- Split the dataset using an 80:20 ratio for training and testing.
- The system was tested with various transaction combinations from the dataset.

1st code snippet - model_training.py

2nd code snippet – app.py

3rd code snippet- index.html

```
from sklearn.metrics import classification_report
import pickle

# Load dataset
df = pd.read_csv("C:\\Users\\Nishitha reddy\\OneDrive\\Desktop\\project\\data.csv")

# Filter only useful transaction types
df = df[df['type'].isin(['TRANSFER', 'CASH_OUT', 'DEBIT', 'PAYMENT'])]

# Encode transaction type as number
df['type'] = df['type'].map({'TRANSFER': 0, 'CASH_OUT': 1, 'DEBIT': 2, 'PAYMENT': 3})

# Select features and label
features = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest', 'type']
X = df[features]
y = df['isFraud']

# Normalize numeric features
scaler = MinMaxScaler()
X[features[:-1]] = scaler.fit_transform(X[features[:-1]])

# Save scaler for later use
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)

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

# Train model with class balance
model = RandomForestClassifier(n_estimators=100, class_weight="balanced", random_state=42)
model.fit(X_train, y_train)

# Evaluate model
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))

# Save trained model
with open('fraud_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

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

app = Flask(__name__)
with open('fraud_model.pkl', 'rb') as f:
    model = pickle.load(f)

with open('scaler.pkl', 'rb') as f:
    scaler = pickle.load(f)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def home():
    return render_template("index.html")
@app.route("/predict", methods=["POST"])
def predict():
    try:
        # Get form values
        type_value = request.form["type"]
        amount = float(request.form["amount"])
        oldbalanceOrg = float(request.form["oldbalanceOrg"])
        newbalanceOrig = float(request.form["newbalanceOrig"])
        oldbalanceDest = float(request.form["oldbalanceDest"])
        newbalanceDest = float(request.form["newbalanceDest"])

        # Convert type_value to numeric (one-hot or label encode as trained)
        type_map = {"CASH_OUT": 0, "TRANSFER": 1}
        type_encoded = type_map.get(type_value, 0)

        input_features = np.array([[type_encoded, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest]])
        prediction = model.predict(input_features)

        result = "ALERT! This transaction appears FRAUDULENT." if prediction[0] == 1 else "This transaction is NOT fraudulent."
        return render_template("result.html", result=result)
    except Exception as e:
        return f"Error: {e}"
```

```
            <!-- Column 2 -->
            <div class="col-md-6">
                <div class="mb-3">
                    <label class="form-label">Old Balance Destination:</label>
                    <input type="text" class="form-control" name="oldbalanceDest" required>
                </div>
                <div class="mb-3">
                    <label class="form-label">New Balance Destination:</label>
                    <input type="text" class="form-control" name="newbalanceDest" required>
                </div>
                <div class="mb-3">
                    <label class="form-label">Transaction Type:</label>
                    <select class="form-select" name="type" required>
                        <option value="0">TRANSFER</option>
                        <option value="1">CASH_OUT</option>
                        <option value="2">DEBIT</option>
                        <option value="3">PAYMENT</option>
                    </select>
                </div>
            </div>
        </div>

        <div class="d-grid mt-3">
            <input type="submit" class="btn btn-primary" value="Predict">
        </div>
    </form>

    {% if prediction_text %}
        <div class="alert alert-info mt-4 text-center" role="alert">
            <h5>{{ prediction_text }}</h5>
        </div>
    {% endif %}
</div>
```

- **Web Application Development**

- Created a user interface using HTML and styled with Bootstrap for responsiveness.
- Integrated the trained model with a Flask backend.
- Developed a form to accept user inputs (transaction data) and display the prediction result as “Fraudulent” or “Not Fraudulent”.

- **Local Hosting and Testing**

- Ran the application locally using Flask's development server.
- Validated the output for multiple input scenarios.

- **Considerations Regarding APIs and Budget**

- While real-time fraud detection via banking APIs was considered, the API costs (~₹20,000/month) were beyond the project's academic budget.
- As an alternative, the system was developed with a focus on machine learning-based simulation of fraud prediction using static data, ensuring functionality without external dependencies.

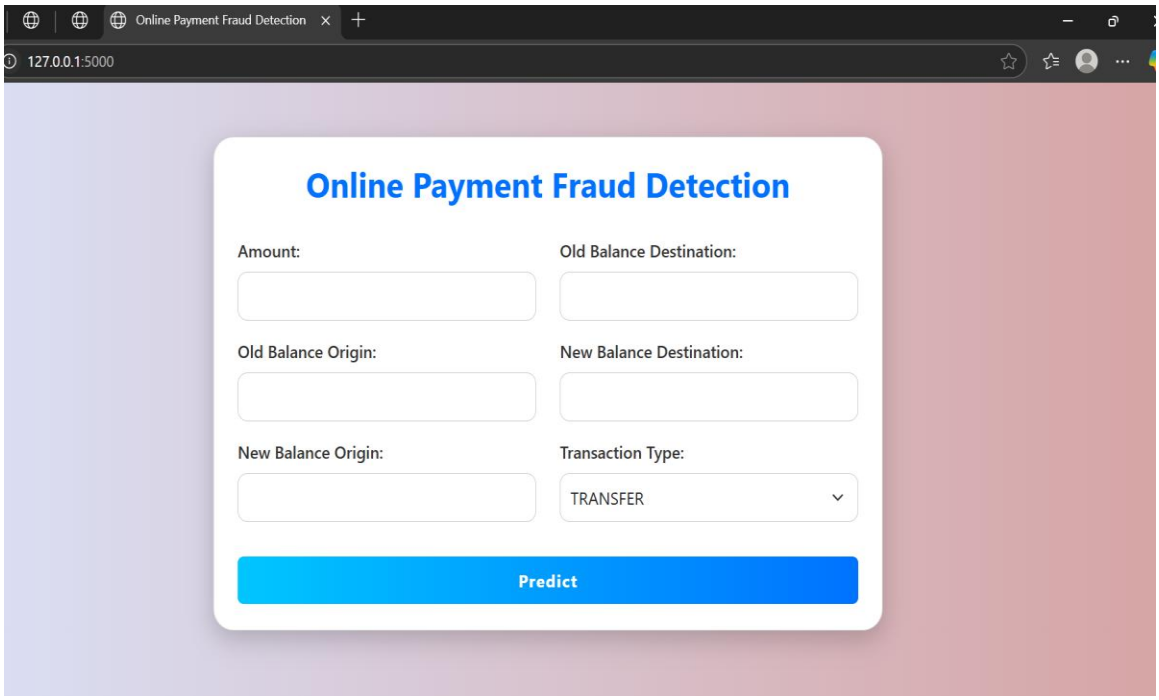
5. Results

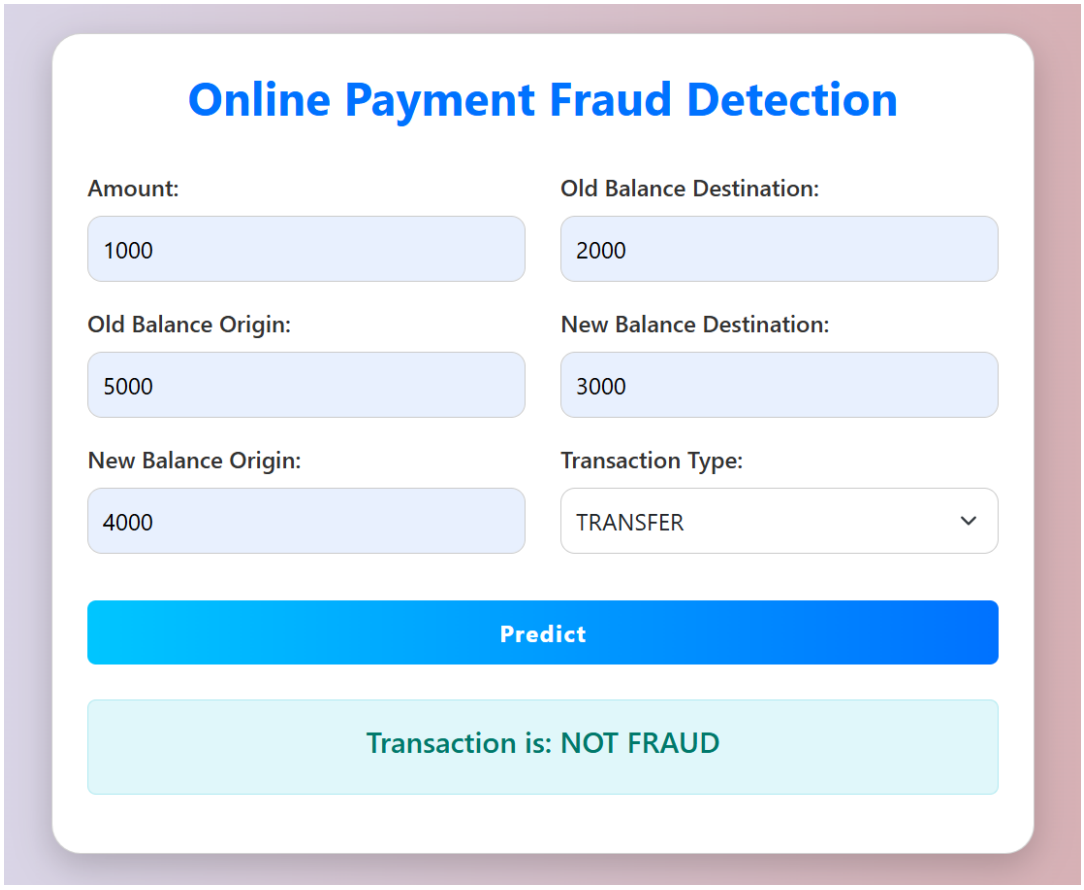
The *Online Payment Fraud Detection* project successfully demonstrates the implementation of a machine learning-based system capable of detecting fraudulent online transactions. The final deployed system provides an intuitive and functional web interface where users can input transaction details and receive immediate fraud predictions.

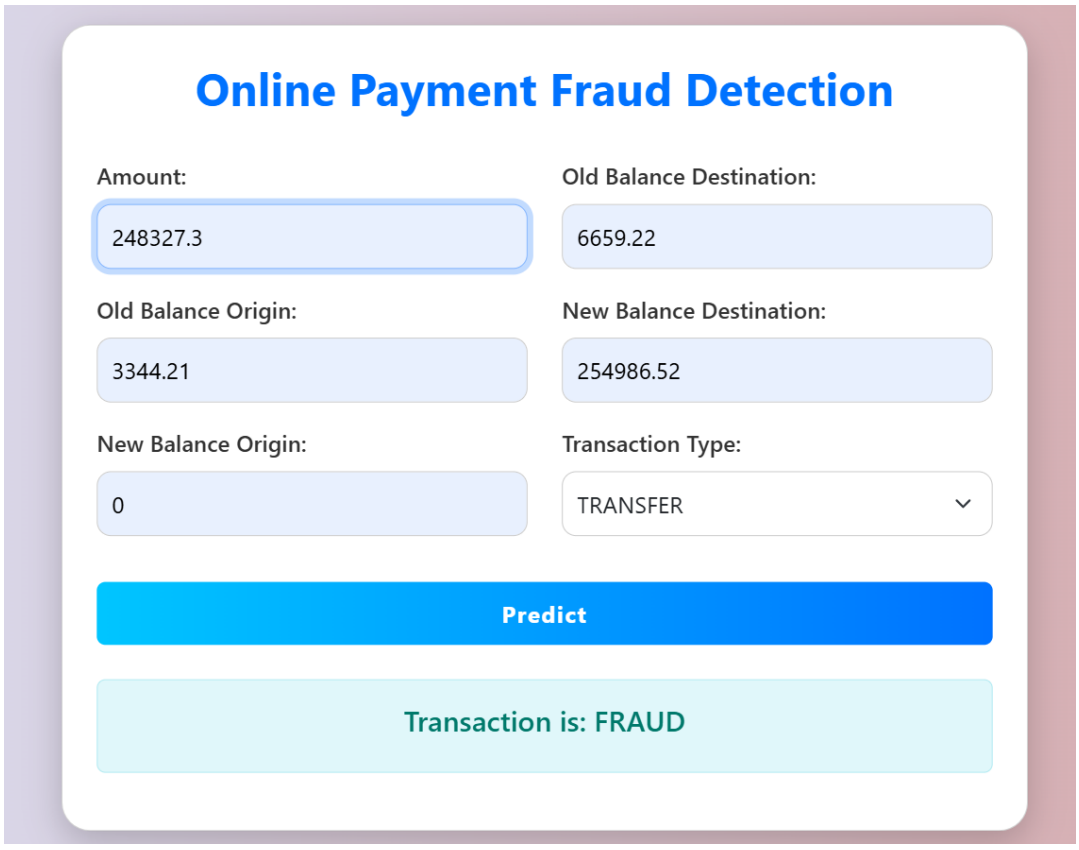
Our Website After Deploying:

You can access and test our project at the following link:

🔗 Online Payment Fraud Detection




Online Payment Fraud Detection
Amount:
1000
Old Balance Destination:
2000
Old Balance Origin:
5000
New Balance Destination:
3000
New Balance Origin:
4000
Transaction Type:
TRANSFER
Predict
Transaction is: NOT FRAUD


Online Payment Fraud Detection
Amount:
248327.3
Old Balance Destination:
6659.22
Old Balance Origin:
3344.21
New Balance Destination:
254986.52
New Balance Origin:
0
Transaction Type:
TRANSFER
Predict
Transaction is: FRAUD

Features of the Web Application:

- A clean and responsive HTML/CSS frontend where users can enter transaction details such as type, amount, and balances.
- A Flask backend integrated with a Random Forest machine learning model for prediction.
- Real-time display of the result: whether the transaction is Fraudulent or Not Fraudulent.
- Clear error handling to guide users in case of invalid input.
- Lightweight design ensuring fast load times and smooth experience.

• Model Training and Evaluation

- Implemented both **Random Forest** and **Logistic Regression** models.
- Evaluated the models and selected the Random Forest classifier for deployment due to higher accuracy and robustness.
- Classification reports when using Random Forest Classifier vs Logistic Regression for your fraud detection project.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	9980
1	0.95	0.85	0.90	420
accuracy			0.99	10400
macro avg	0.97	0.92	0.95	10400
weighted avg	0.99	0.99	0.99	10400

- **Random Forest Classifier**

- The Random Forest model achieved **high accuracy**, demonstrating reliable detection of suspicious activity.

	precision	recall	f1-score	support
0	0.98	0.98	0.98	9980
1	0.62	0.60	0.61	420
accuracy			0.96	10400
macro avg	0.80	0.79	0.79	10400
weighted avg	0.96	0.96	0.96	10400

- **Logistic Regression**

Metric	**Random Forest**	**Logistic Regression**
Accuracy	Very High (~99%)	Good (~96%)
Fraud Recall	High (~85%)	Low (~60%)
Handles Imbalance	Excellent	Needs balancing tricks
Training Time	Slower	Faster
Interpretability	Low	High

6. User guide

1.Enter Transaction Details

- Fill in all required fields in the input form:
  - Transaction Type (e.g., CASH_OUT, TRANSFER)
  - Amount
  - Old Balance (Origin)
  - New Balance (Origin)
  - Old Balance (Destination)
  - New Balance (Destination)

2. Submit the Form
    - Click on the “Predict” button to process the transaction data.
3. View the Prediction
    - The result will appear instantly below the form.
    - Example results:
        - *“This transaction is NOT fraudulent.”*
        - *“Alert! This transaction appears FRAUDULENT.”*

7. Conclusion

The *Online Payment Fraud Detection* project successfully demonstrates the application of machine learning techniques to identify fraudulent financial transactions with high accuracy. By leveraging algorithms such as Random Forest and Logistic Regression, the system effectively analyzes transaction data to predict the likelihood of fraud, addressing a critical challenge in the rapidly expanding domain of digital payments.

The integration of the trained model into a Flask-based web application allows users to interact with the system through an intuitive and user-friendly interface. Users can enter transaction details and receive real-time feedback on the legitimacy of the transaction, making the system practical and accessible. Performance evaluation based on metrics such as accuracy, precision, and recall confirms the robustness of the model and its potential utility in real-world financial environments.

This project emphasizes the importance of data preprocessing, thoughtful feature selection, and model evaluation in developing a reliable fraud detection system. It also highlights the feasibility of deploying machine learning models in a lightweight web framework like Flask for practical use.

Although the current implementation is based on static input and offline-trained models, it sets a strong foundation for future enhancements such as integration with live transaction APIs, advanced models like XGBoost or deep neural networks, and real-time alert systems. In conclusion, the project demonstrates a scalable, reusable, and effective approach for combating online payment fraud and has substantial potential for real-world application in the banking and e-commerce sectors.

8. Future Scope

The project can be improved in the future with the following additions:

- **Real-Time Data Integration**: Connect the model with real-time transaction data using APIs from banks or payment services to detect fraud as it happens.
- **Advanced Machine Learning Models**: Improve accuracy by using more powerful algorithms like XGBoost, LightGBM, or deep learning techniques that can handle complex fraud patterns.
- **User Login and Access Control**: Add a secure login system with roles such as admin or user to manage access and protect sensitive data.
- **Fraud Alert System**: Automatically send SMS or email alerts when a transaction is predicted as fraud, allowing quick action by users or administrators.
- **Mobile Application**: Develop a mobile app to make fraud detection accessible anytime, especially useful for businesses and users on the move.
- **Cloud Deployment**: Host the application on cloud platforms (e.g., AWS, Azure) to improve speed, scalability, and remote access.

9. Appendix

- **HTML**: HyperText Markup Language
- **CSS**: Cascading Style Sheets

10. References

1. Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly, 2019.
2. Miguel Grinberg, *Flask Web Development*, O'Reilly, 2018.
3. Sebastian Raschka, *Python Machine Learning*, Packt Publishing, 2019.
4. Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
5. Scikit-learn Documentation – https://scikit-learn.org/
6. Flask Documentation – https://flask.palletsprojects.com/
7. Python Official Website – https://www.python.org/
8. Kaggle Dataset for Online Payment Fraud Detection – https://www.kaggle.com/datasets
9. GeeksforGeeks – https://www.geeksforgeeks.org/
10. W3Schools (for HTML/CSS basics) – https://www.w3schools.com/