Project Design Phase – Online Payments Fraud Detection

1. Problem Statement:

Online payment systems are increasingly targeted by **fraudulent transactions**, leading to financial losses for users and institutions. Detecting these frauds manually is **time-consuming, error-prone, and inefficient** due to the massive volume of transactions processed daily.

The challenge is to **build an automated system** that can detect fraudulent transactions in real-time, with high accuracy, and alert users or institutions to prevent financial loss.

Real-Time Scenario:

- A customer makes an unusually large transfer. The system analyzes transaction type, amount, and account behavior and predicts whether it is **fraudulent (1)** or **legitimate (0)**.
- This prevents financial loss and enhances **trust in digital payment platforms**.

2. Selected Algorithm:

Random Forest Classifier

3. Description and Reason for Selection

Random Forest is an ensemble learning algorithm that combines multiple **decision trees** to improve predictive performance and reduce overfitting.

Reasons for Choosing Random Forest:

1. **High Accuracy:** Performs well on complex datasets with categorical and numerical features.
2. **Handles Imbalanced Data:** Can deal with rare fraudulent transactions without requiring heavy preprocessing.
3. **Feature Importance:** Helps identify **which transaction features contribute most to fraud detection**.
4. **Robustness:** Less prone to overfitting compared to a single decision tree.
5. **Ease of Implementation:** Available in **Scikit-learn** with straightforward integration into Python projects.

Other algorithms like **SVM, Extra Trees, and Decision Trees** were also considered, but Random Forest provided the **best balance of accuracy, interpretability, and computational efficiency** for this dataset.

4. Proposed Solution

The proposed solution is a **machine learning-based fraud detection system** integrated into a **Flask web application**.

Solution Workflow:

1. **User Input:**
    - Users enter transaction details via a web interface.
2. **Data Preprocessing:**
    - Encode categorical features (e.g., transaction type) using **LabelEncoder**.
    - Scale numerical features (e.g., transaction amount, account balances) using **StandardScaler**.
    - Handle outliers if necessary.
3. **Machine Learning Prediction:**
    - The preprocessed transaction data is passed to the **Random Forest model**.
    - The model predicts the **isFraud** label:
        - 0 → Legitimate transaction
        - 1 → Fraudulent transaction
4. **Result Display:**
    - Flask displays the prediction on a **result page** for the user.

Optional Features:

- Visualization of transaction patterns.
- Logging predictions for future model retraining and improvement.

5. About the Selected Algorithm (Random Forest)

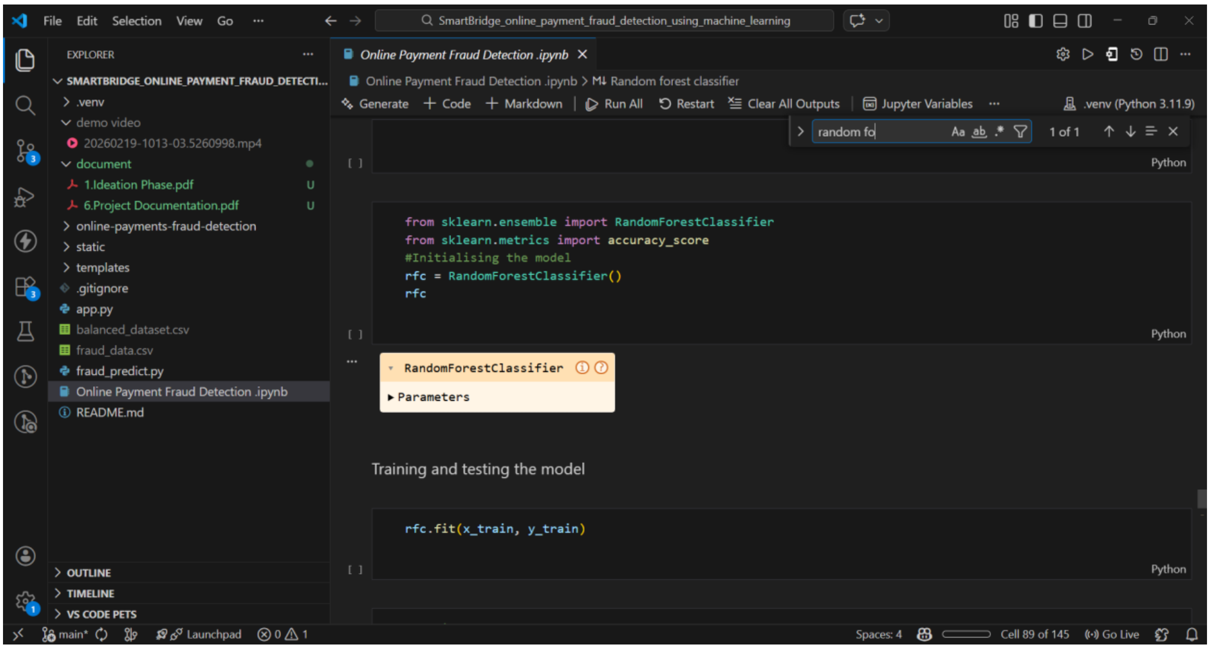
- **Type:** Ensemble learning algorithm (Bagging method)
- **Components:** Collection of multiple decision trees. Each tree is trained on a **random subset of the data**.
- **Prediction:**
    - For classification, each tree votes for a class.
    - The class with the majority votes is selected as the final prediction.

Advantages for Fraud Detection:

- Reduces the **variance** of single decision trees.

- Handles **high-dimensional data** with both numerical and categorical features.
- Provides **feature importance scores** to understand the most critical predictors of fraud.

Python Implementation Example:


File Edit Selection View Go
SmartBridge_online_payment_fraud_detection_using_machine_learning
EXPLORER
SMARTBRIDGE_ONLINE_PAYMENT_FRAUD_DETECTI...
.venv
demo video
20260219-1013-03.5260998.mp4
document
1.Ideation Phase.pdf
6.Project Documentation.pdf
online-payments-fraud-detection
static
templates
.gitignore
app.py
balanced_dataset.csv
fraud_data.csv
fraud_predict.py
Online Payment Fraud Detection .ipynb
README.md
OUTLINE
TIMELINE
VS CODE PETS
Online Payment Fraud Detection .ipynb
Online Payment Fraud Detection .ipynb > Random forest classifier
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables
.venv (Python 3.11.9)
random fo
1 of 1
Python
main* Launchpad 0 1
Spaces: 4 Cell 89 of 145 Go Live

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
#Initialising the model
rfc = RandomForestClassifier()
rfc
```

```
RandomForestClassifier
▸ Parameters
```

Training and testing the model

```
rfc.fit(x_train, y_train)
```

6. Conclusion

- The **Project Design Phase** outlines a **clear plan** for implementing the fraud detection system.
- Random Forest is chosen for its **accuracy, robustness, and interpretability**.
- The proposed solution combines **machine learning and web deployment**, enabling **real-time fraud prediction** and **user interaction**.