Credit Card Fraud Detection System Abstract

Abstract: Credit Card Fraud Detection System Using Machine Learning and Rule-Based Verification

This project presents a hybrid Credit Card Fraud Detection System that leverages both Machine Learning (ML) and rule-based verification techniques to identify potentially fraudulent transactions. Developed using Python and implemented with a GUI using Tkinter, the system is designed to analyze anonymized transaction data and classify transactions as legitimate or fraudulent, enhancing the security of credit card systems while preserving user privacy.

System Overview

The system uses a Random Forest Classifier, a robust ensemble learning method, trained on the popular credit card fraud detection dataset which includes anonymized features (V1–V28) generated through Principal Component Analysis (PCA). These transformed features represent underlying transaction behaviors while concealing sensitive information. Alongside these, the system also considers the transaction amount and a class label to signify whether a transaction is fraudulent.

How It Works

The detection system operates in multiple stages:

1. Data Preprocessing:  
 - Loads transaction data from a CSV file.  
 - Removes unnecessary columns (e.g., Time).  
 - Scales numerical features using StandardScaler from sklearn.preprocessing.

2. Model Training & Evaluation:  
 - Splits the dataset into training and testing sets using train\_test\_split.  
 - Trains a RandomForestClassifier on scaled data.  
 - Evaluates model performance with confusion matrix, classification report, and accuracy score.

3. Fraud Prediction:  
 - Accepts individual transaction data input (V1–V28, Amount).  
 - Predicts the fraud probability and outputs the result using the trained model.

4. Rule-Based Verification:  
 - Applies simple yet effective custom rules:  
 - Detects unusually large amounts.  
 - Flags atypical transaction locations.  
 - Identifies multiple rapid transactions.  
 - Enhances detection by checking against the user's transaction history.

5. Graphical User Interface (GUI):  
 - Built using Tkinter for user-friendly interaction.  
 - Features tabbed navigation to:  
 - Train and save/load models  
 - Test individual transactions  
 - Perform rule-based checks

6. Visualization:  
 - Displays confusion matrix using matplotlib and seaborn embedded in Tkinter via FigureCanvasTkAgg.

Key Features

- 💻 Model Training Interface: Load datasets, train and evaluate models with real-time metric output.  
- 🧠 ML-Powered Predictions: Analyze new transactions using a trained Random Forest model.  
- 🧾 Rule-Based Checking Module: Verify suspicious patterns based on domain-specific rules.  
- 💾 Model Persistence: Save and load models using pickle for reuse without retraining.  
- 📊 Visualization: Heatmap of confusion matrix for intuitive model performance insight.  
- 🧑‍💻 Custom GUI: Fully interactive desktop interface with structured tabs and input validation.

Packages and Their Roles

| Package | Purpose |  
|--------|---------|  
| pandas, numpy | Data handling, transformation, and numerical operations |  
| scikit-learn | Machine learning model training, preprocessing, evaluation |  
| matplotlib, seaborn | Visualization, especially for confusion matrix |  
| tkinter, ttk | GUI development, user input, layout design |  
| pickle | Saving and loading trained models for persistence |  
| io, sys | Capturing and displaying stdout in GUI text boxes |

This system exemplifies a multi-layered fraud detection architecture, combining statistical learning with heuristic rules. It is ideal for educational purposes, demonstrating how AI and conventional logic can work in tandem to solve real-world problems like financial fraud in a privacy-conscious manner.