BY:
ALEJANDRO LOPEZ AZORIN

BANK FRAUD

DETECTION SYSTEM

INDEX

1. Introduction

This project simulates a **bank fraud detection system** using a combination of OOP and machine learning. The system models **users, transactions, banks, and analysts**, where an AI model predicts the likelihood of a transaction being fraudulent.

The main goal is to **demonstrate ML integration into a real-world scenario** and to create a modular, testable Python project.

2. Project Overview

The project consists of:

- Users performing deposits, withdrawals, and transfers.
- Banks coordinating transactions and calling analysts for approval.
- Analysts using a trained **ML model** to assess transaction risk.
- Transactions that log all activities for traceability.

Additionally, the project includes a **FastAPI interface** to expose the ML model for external clients.

3. Features

- **User Management:** Create users, check balances, perform transactions.
- **Fraud Detection:** ML model predicts fraud probability (p_fraud) and fraud status (is_fraud).
- **Manual Review:** Analysts can approve or block suspicious transactions.
- **Logging:** Full traceability of transactions, decisions, and ML predictions.
- **API:** FastAPI endpoint to submit transaction data and receive predictions.
- **Unit & Integration Testing:** Pytest tests covering classes and API endpoints.

4. Project Structure

Bank Fraud Detection System/

- **README.pdf**
- **requirements.txt** # Needed libraries and packages
- **main.py** # Demo program
- **api.py** # API's configuration
- **data/** # Needed data for ML model training
  - **raw_data.csv**
  - **data_cleaner.py** # Cleans raw_data.csv y returns cleaned_data.csv
  - **cleaned_data.csv**
  - **add_concept_to_data.csv** # Adds concepts and Is_fraud columns to cleaned_data.csv and returns final_data.csv
  - **final_data.csv** #Data ready for ML model training
- **ML/**
  - **train_model.py** # Trains ML model and saves the model as model.pkl
  - **model.pkl** # Trained model
  - **predict.py** # Uses model.pkl in order to make predictions
  - **utils.py** # Auxiliar functions of predict.py
- **bank_system/**
  - **user.py** # Class User
  - **bank.py** # Class Bank
  - **analyst.py** # Class Analyst
  - **transaction.py** # Class Transaction
- **tests/** #unit testing for bank_system and API's
  - **test_user.py**
  - **test_bank.py**
  - **test_analyst.py**
  - **test_transaction.py**
  - **test_api.py**

5. Installation

1. Clone the repository:

```
git clone https://github.com/alexlopezazorin/bank_fraud_detection
```

2. Install dependencies:

```
pip install -r requirements.txt
```

6. Running the Project

- **Demo (CLI)**: Run main.py to simulate transactions locally.
- **Train ML Model**: Execute train_model.py to train and save the model.
- **FastAPI**: Run api.py to start the API server: uvicorn api:app --reload
- Access the API documentation at http://localhost:8000/docs

7. Machine Learning Model

- Model: Logistic Regression with TfidfVectorizer for the Concept column and StandardScaler for numeric features.
- Features: Amount, time_sin, time_cos, Is_Night, Concept.
- Target: Is_Fraud (1 = fraud, 0 = safe).
- Dataset: final_data.csv with probabilistic assignment of fraud based on amount, time, and concept.
- Metrics Achieved:
  Accuracy: 0.77
  ROC-AUC: 0.82
  Precision / Recall / F1-scores vary for fraud and safe classes.

8. API Usage

Endpoint: /predict (POST)

Request body(.json):


```
{
"amount": 51000,
"concept": "urgent crypto investment",
"local_time": "22:30:00"
}
```


Response body(.json):


```
{
"is_fraud": 1,
"p_fraud": 0.92
}
```


- The API allows external clients to receive ML predictions without needing the full project.
- local_time is sent as a string in "HH:MM:SS" format.

9. Logging

- **Purpose:** Track all transactions, analyst decisions, and ML predictions.
- **Logging Levels:**
  - INFO: Transaction creation, approvals, and rejections.
  - WARNING: Manual review required.
- **Example:** In transaction.py, logger.info records each new transaction.
- Logs are saved in logs/bank_system.log and include timestamps.

10. Testing

- **Unit Tests:** For User, Bank, Analyst, Transaction.
- **Integration Tests:** Full bank transaction flows.
- **API Tests:** Using fastapi.testclient to validate /predict endpoint.
- Tests are runned with “python -m pytest” on a cmd on the main directory of the project