Data Quality and Anomaly Detection using AI Agents in Banking/Finance

Index:

Problem Overview

Large organizations, especially in sectors like **banking and finance**, often face the challenge of **data anomalies**. These anomalies can manifest in many ways, such as:

- **Missing entries** (incomplete data records),
- **Fraud patterns** (unusual transactions or suspicious behavior),
- **Duplicate transactions** (repeated entries due to errors),
- **Inconsistent data** (e.g., unrealistic values for credit scores or card opening years).

These issues can degrade data quality, making it difficult to make reliable decisions. **Manual review** of transactions is time-consuming and prone to human error, making it crucial for organizations to implement automated anomaly detection systems using Agent AI.

Solution Overview

The solution proposed is the use of **AI Agents** for **Data Monitoring** to handle anomaly detection in an automated, scalable, and efficient manner. These agents are categorized as follows:

1. **Detection Agent**:
    - **Purpose**: Monitors streaming or batch data for any discrepancies or irregular patterns.
    - **Functionality**: Uses **AI models** (like **Mistral LLM**) to automatically detect anomalies such as missing entries, duplicate transactions, or fraud.

2. **Validation Agent**:
    - **Purpose**: Flags anomalies for further review or validation.
    - **Functionality**: Ensures that flagged anomalies meet certain criteria (e.g., transactions outside expected ranges) before they are passed for resolution.

3. **Resolution Agent**:
    - **Purpose**: Suggests fixes or triggers workflows to address detected anomalies.
    - **Functionality**: Once the anomaly is validated, this agent could automatically suggest fixes (e.g., correcting missing data, flagging duplicate transactions for manual review, or alerting compliance teams for potential fraud).

The system consists of multiple agents, each specialized in a different aspect of anomaly detection, validation, and resolution. Here's a breakdown of the components:

1. Environment Setup

- The script first loads environment variables (such as the OpenRouter API key) from a `.env` file. If the `python-dotenv` package isn't installed, it tries to use system environment variables directly.

- LangChain libraries are imported and installed if not present. LangChain facilitates communication with LLMs (large language models) like OpenAI models and allows for fine-tuned control over AI-driven processes.

2. Configuration

- The script initializes **OpenRouter** (a platform for accessing AI models) using an API key (`OPENROUTER_API_KEY`).

- The `mistralai/mistral-large` model from Mistral is selected as the language model for AI-driven tasks.

3. Data Models

- Several **data models** (using `pydantic`) are defined to handle anomalies, validation, and resolution actions. These include:

    - **AnomalyType**: Lists types of anomalies like missing data, fraud patterns, duplicates, etc.

    - **DetectedAnomaly**: Defines the structure for anomalies detected in transactions.

    - **ValidationResult**: Used to store results of validating whether an anomaly is real.

    - **ResolutionAction**: Describes actions to resolve anomalies (e.g., flag, alert, fix).

4. Anomaly Detection Agent

- The **DetectionAgent** uses the Mistral LLM model to detect anomalies in transaction data. It is provided a prompt to detect:

    - Missing data
    - Fraud patterns
    - Duplicate transactions

    - Inconsistent data (e.g., unrealistic credit scores or amounts)

- It processes the transactions and returns a list of detected anomalies.

5. Anomaly Validation Agent

- The **ValidationAgent** validates the detected anomalies. It reviews the anomalies and decides if they are genuine, what the reason for validation is, whether human review is needed, and the priority level for addressing the anomaly.

- It uses a prompt to validate anomalies by considering business rules, historical patterns, risk levels, and regulatory requirements.

6. Resolution Agent

- The **ResolutionAgent** suggests actions to resolve validated anomalies. These actions can include:

    - **Fix**: Automatically correctable anomalies.
    - **Flag**: No immediate action but should be flagged for review.
    - **Alert**: Immediate notification of an issue to a compliance or fraud team. **Investigate**: Requires detailed investigation by an analyst.

- It suggests specific fixes and triggers workflows where appropriate.

7. Orchestrator

- The **AnomalyDetectionOrchestrator** coordinates the workflow by:

    1. Calling the **DetectionAgent** to find anomalies.
    2. Passing those anomalies to the **ValidationAgent** for validation.
    3. If the anomalies are valid, passing them to the **ResolutionAgent** to suggest remediation actions.

- It provides a **summary report** on the number of anomalies detected, validated, and resolved, including the number of automated and manual fixes.

8. Example Usage

- The script contains **sample transaction data** with various types of anomalies:

    - Transaction with a missing date.

    - Transaction with an unusually high amount.
    - Duplicate transaction.
    - Transaction with an impossible credit score.
- It initializes the **AnomalyDetectionOrchestrator**, processes the transactions, and saves the results as a JSON file.

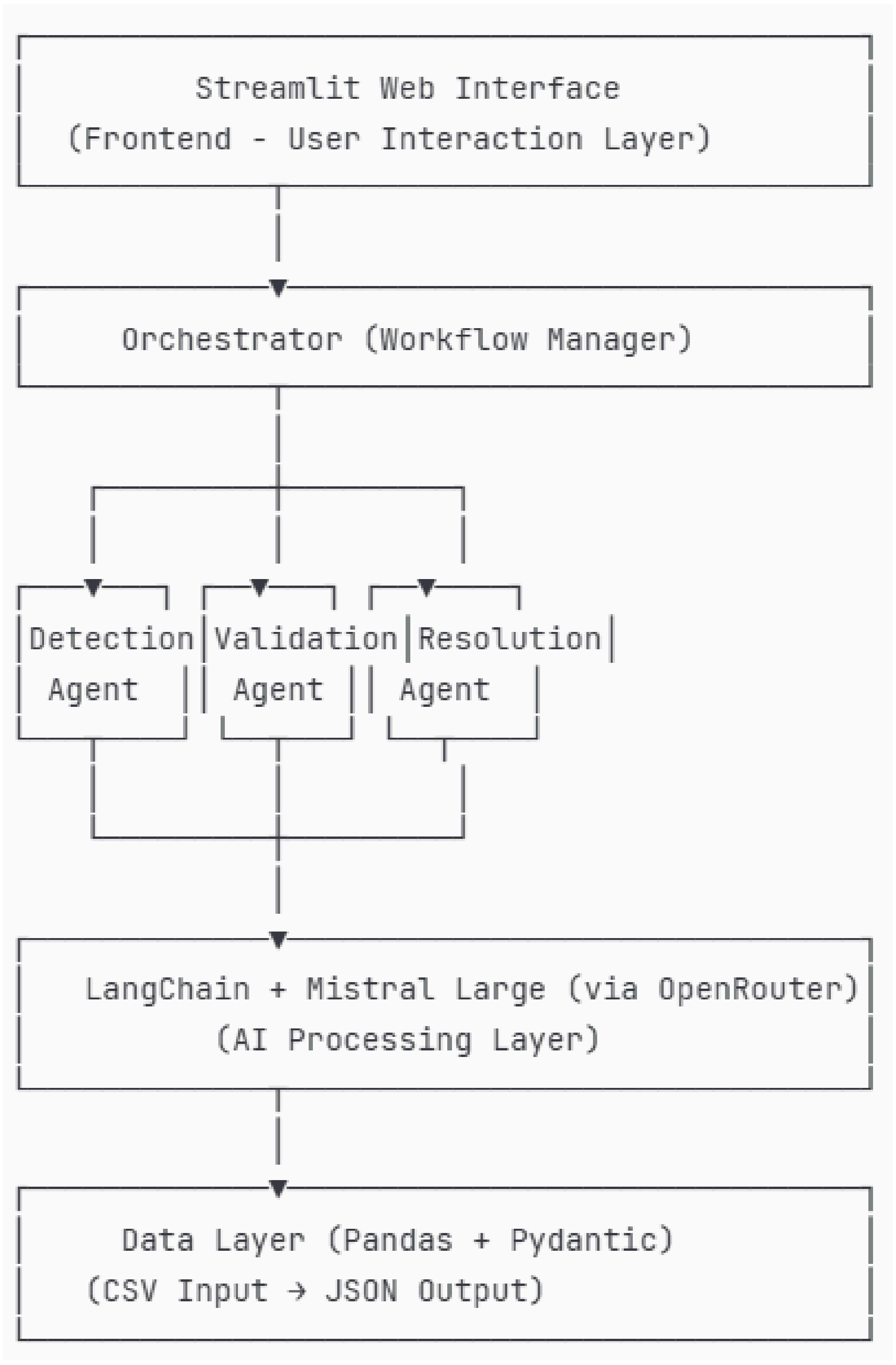
What Happens When the Script Runs:

- **Step 1**: The script processes the sample transaction data through the detection agent to identify anomalies.
- **Step 2**: It validates each anomaly, checking if they require human review or if they're false positives.
- **Step 3**: For valid anomalies, it generates resolution actions (e.g., fixing, flagging, or alerting).
- Finally, it **prints a summary** of the detection results and saves them to a file.

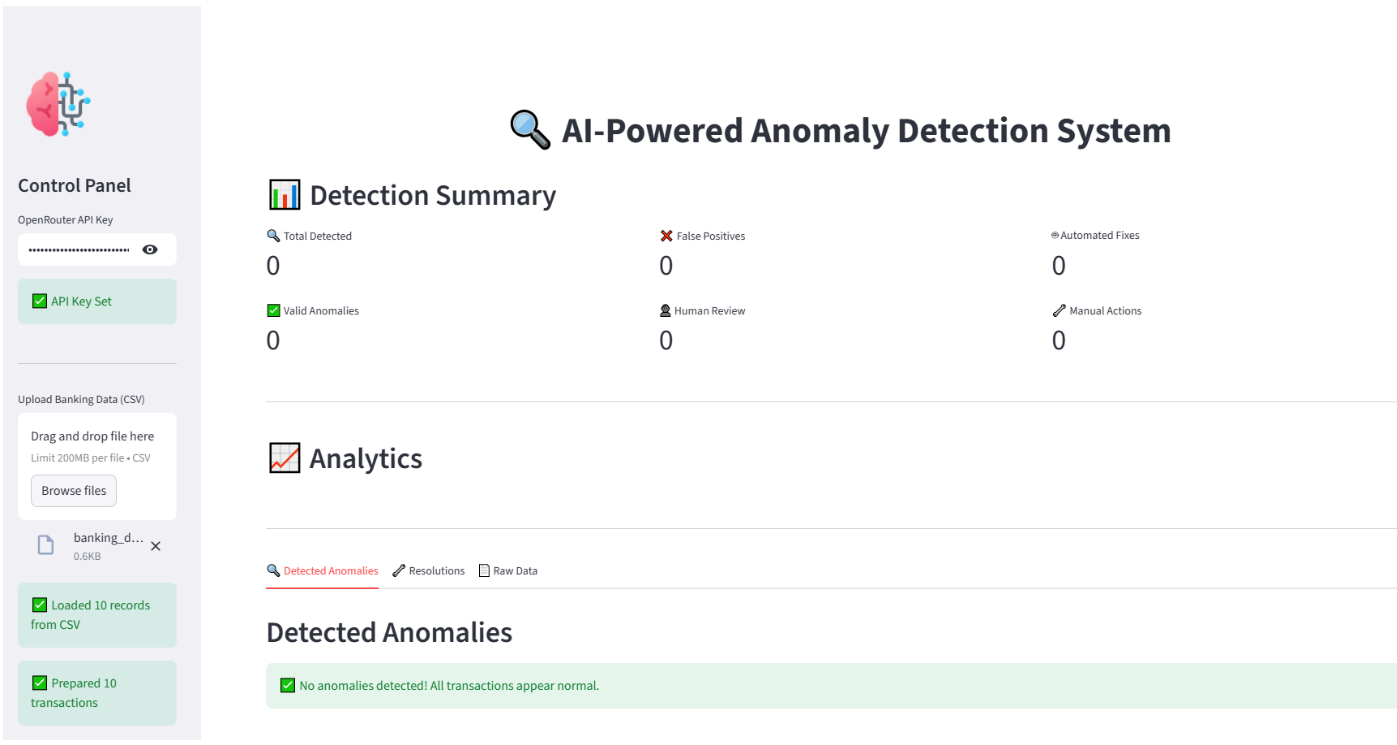
Why This is Useful:

- **Automation**: The system automatically detects, validates, and resolves anomalies without manual intervention.
- **AI Integration**: By leveraging language models (like Mistral), the system can understand complex data patterns and make decisions in a way that traditional rule-based systems can't.
- **Real-Time Monitoring**: The system could be used in production for real-time anomaly detection in banking and finance, enabling quicker responses to potential fraud or errors.

Workflow of Agents:


Streamlit Web Interface
(Frontend - User Interaction Layer)
Orchestrator (Workflow Manager)
Detection Agent
Validation Agent
Resolution Agent
LangChain + Mistral Large (via OpenRouter)
(AI Processing Layer)
Data Layer (Pandas + Pydantic)
(CSV Input → JSON Output)

Page snippets:



Control Panel

OpenRouter API Key

✅ API Key Set

Upload Banking Data (CSV)

Drag and drop file here

Limit 200MB per file • CSV

Browse files

banking_d... 0.6KB

✅ Loaded 10 records from CSV

✅ Prepared 10 transactions

🔍 AI-Powered Anomaly Detection System

📊 Detection Summary

🔍 Total Detected	❌ False Positives	Automated Fixes
0	0	0
✅ Valid Anomalies	👤 Human Review	Manual Actions
0	0	0

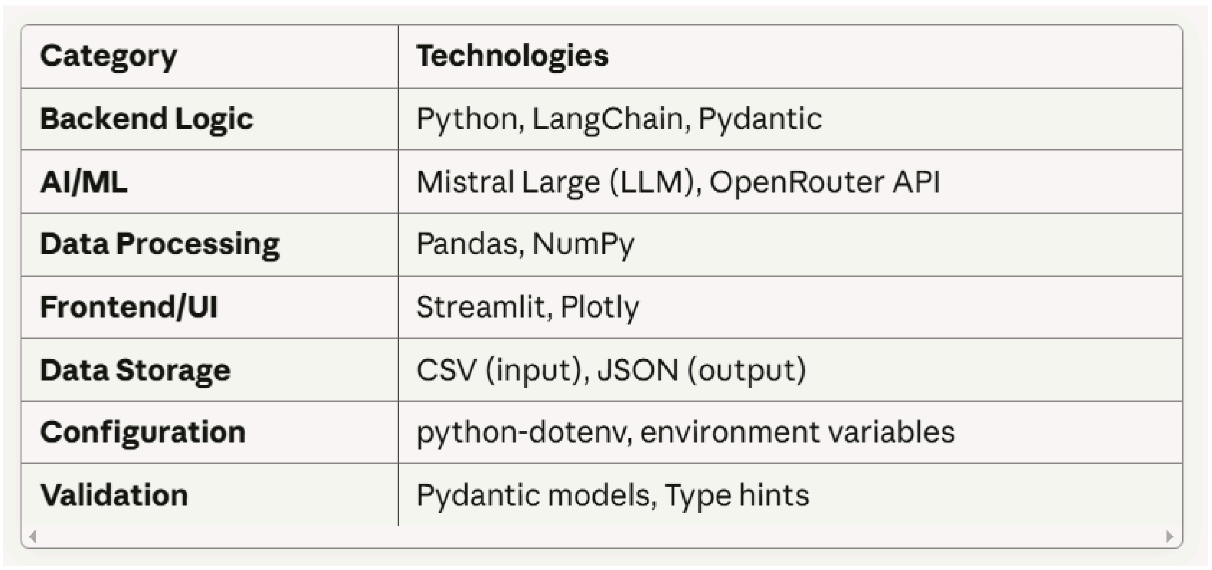
📈 Analytics

🔍 Detected Anomalies | Resolutions | Raw Data

Detected Anomalies

✅ No anomalies detected! All transactions appear normal.

Tech Stacks Used:



Category	Technologies
Backend Logic	Python, LangChain, Pydantic
AI/ML	Mistral Large (LLM), OpenRouter API
Data Processing	Pandas, NumPy
Frontend/UI	Streamlit, Plotly
Data Storage	CSV (input), JSON (output)
Configuration	python-dotenv, environment variables
Validation	Pydantic models, Type hints

Challenges Faced

- **Module Import Issues**: Initially, you encountered issues with missing modules (like `langchain.prompts`), which were resolved by updating packages.
- **API Configuration**: The process of configuring the **OpenRouter API** and securely managing the **API keys** in the `.env` file was challenging but successfully resolved.
- **Parsing Model Outputs**: The challenge of parsing raw model output into a structured JSON format was handled using error-catching techniques to ensure robust results.

Future Extensions

- **Real-time Monitoring**: Integrate with streaming data sources for **real-time anomaly detection**.

- **Fraud Detection Integration**: Enhance the model's fraud detection capabilities by integrating additional fraud-detection models or external data sources.

- **Full Workflow Automation**: Implement a complete system with **detection, validation, and resolution** agents to automatically flag, validate, and resolve anomalies without manual intervention.

10. Resources and Links

- **GitHub Repository**:
  Access the complete source code and documentation for the project:
  GitHub Repository Link

- **Published Page**:
  Read more about the project on the official page:
  Project Published Page