**Credit Fraud Detection Using the Hidden Markov Model**

Team Members:

Sergio Gabriel Jiawei Kun ; s\_kun@mail.fhsu.edu

Lee Joseph Judkins ; ljjudkins@mail.fhsu.edu

Project Github: <https://github.com/kunsergio117/CreditFraudDetectionHMM.git>

#### **Customer Statement of Requirements**

Credit fraud is an ever-evolving threat to financial institutions and their customers. Current fraud detection systems, which often rely on static, rule-based algorithms, struggle with high false-positive rates, inflexibility, and inefficiency when handling massive transaction datasets. False positives frustrate customers, who might find their accounts frozen during legitimate transactions, and burden financial institutions with unnecessary alerts. These outdated systems also lack the adaptability needed to counter the rapidly evolving techniques of fraudsters, leaving banks exposed to significant losses.

To address these challenges, our project proposes a robust fraud detection system that leverages **Hidden Markov Models (HMMs)** and **Recurrent Neural Networks (RNNs)** to analyze transaction patterns dynamically. This hybrid approach aims to enhance fraud detection accuracy, reduce false positives, and provide real-time responsiveness. Core features include CSV transaction data uploads, manual transaction verification, user authentication, real-time alerts, and advanced reporting tools. Additionally, the system will allow the simulation of fraudulent transactions to test its capabilities, ensuring a user-friendly and efficient interface for analysts and customers alike.

#### **Glossary of Terms**

* **HMM (Hidden Markov Model):** A statistical model used to capture temporal patterns and sequential data, aiding in the detection of transaction anomalies.
* **RNN (Recurrent Neural Network):** A type of neural network designed for sequential data analysis, complementing the HMM for enhanced accuracy.
* **CSV (Comma-Separated Values):** A file format used for uploading transaction data into the system.
* **False Positive Rate:** The percentage of legitimate transactions incorrectly flagged as fraudulent.
* **Anomaly Detection:** The process of identifying unusual patterns in data that deviate from the norm.

#### **System Requirements**

**Functional Requirements:**

* REQ1: Enable CSV data uploads for transaction analysis.
* REQ2: Provide visual tools for analyzing transaction patterns.
* REQ3: Allow users to configure fraud detection thresholds.
* REQ4: Integrate APIs for real-time data fetching.
* REQ5: Generate and download fraud detection reports.
* REQ6: Display KPIs such as false positive rates and accuracy.
* REQ7: Ensure secure user authentication.

**Non-Functional Requirements:**

* REQ8: Intuitive and uncluttered user interface.
* REQ9: Maintain 99.5% system availability.
* REQ10: Ensure mobile compatibility.
* REQ11: Provide low-latency alerts for fraudulent activity.
* REQ12: Ensure compliance with financial data regulations (e.g., GDPR, PCI DSS).
* REQ13: Include an admin interface for system management.

**On-Screen Appearance Requirements:**

* REQ14: Use dynamic charts and graphs for data visualization.
* REQ15: Ensure consistency in GUI design across platforms.

#### **Functional Requirements Specification and Use Cases**

**Use Cases for Final Demo:**

1. **CSV Upload:** Users can upload transaction datasets in CSV format for analysis. The system validates and preprocesses the data to ensure integrity.
2. **Manual Transaction Verification:** Analysts can review flagged transactions, examining their attributes and determining whether they are fraudulent.
3. **Real-Time Alerting:** Suspicious transactions trigger immediate alerts, providing details such as transaction amount and merchant.
4. **User Authentication:** Secure login ensures access only for authorized users.

**Future Use Cases:**

* **Threshold Configuration:** Users could set custom alert thresholds to tailor fraud detection sensitivity.
* **API Integration:** Real-time data fetching and integration from external systems could enhance system adaptability.
* **Admin Dashboard:** Admins could monitor system performance and configure detection parameters.

#### **System Sequence Diagrams (SSDs)**

Updated SSDs incorporate the final demo's use cases, such as the CSV upload and alerting features, demonstrating interaction flows between users, the database, and the HMM-RNN system.

#### **System Architecture and Design**

The system follows a modular architecture with separate layers for data preprocessing, fraud detection, and user interaction. Design patterns, including the **Model-View-Controller (MVC)** and **Observer Pattern**, enhance maintainability and real-time responsiveness. The MVC pattern separates business logic from user interface concerns, while the Observer Pattern facilitates timely alert notifications.

#### **Algorithms and Data Structures**

The system uses HMMs for temporal sequence modeling, complemented by RNNs for deep learning of complex patterns. Data structures include hash maps for quick lookup of transaction attributes and graphs for visualizing user-specific transaction behaviors.

#### **History of Work, Current Status, and Future Work**

#### **History of Work**

The project milestones evolved over the weeks as we worked to implement a credit fraud detection system using a Hidden Markov Model (HMM). Below is the timeline of our tasks, team responsibilities, and progress:

| **Week** | **Task Description** | **Responsible Team Members** |
| --- | --- | --- |
| Week 1 | Project kick-off meeting, defined scope and objectives | Sergio Kun, Joseph Judkins |
| Week 2 | Researched HMM algorithms and relevant literature | Sergio Kun |
|  | Familiarized with data sources and datasets | Joseph Judkins |
| Week 3 | Developed initial design and architecture of the application | Sergio Kun, Joseph Judkins |
| Week 4 | Implemented CSV transaction data upload feature | Sergio Kun |
|  | Initiated development of the user authentication process | Joseph Judkins |
| Week 5 | Developed the manual checking function for transaction validity | Sergio Kun |
| Week 6 | Implemented the alerting function for suspicious transactions | Joseph Judkins |
| Week 7 | Integrated simulated fraudulent transaction features | Sergio Kun |
| Week 8 | Tested functionality and debugged | Sergio Kun, Joseph Judkins |
| Week 9 | Developed and integrated reporting tools for transaction summaries | Joseph Judkins |
|  | Refined user interface and enhanced user experience | Joseph Judkins |
| Week 10 | Conducted user testing and gathered feedback | Sergio Kun, Joseph Judkins |
| Week 11 | Finalized features based on user feedback | Sergio Kun, Joseph Judkins |
| Week 12 | Prepared project presentation and documentation | Sergio Kun, Joseph Judkins |
| Week 13 | Reviewed project and made adjustments based on tutor feedback | Sergio Kun, Joseph Judkins |
| Week 14 | Deployed the final project and presented it to the class | Sergio Kun, Joseph Judkins |

Toward the end of the project, we encountered a significant setback when Joseph accidentally spilled a drink on his old computer, resulting in a total loss of progress. This necessitated a complete restart, which compressed the timeline for final deliverables and testing. Despite this, we successfully recovered before the demo.

**Key Accomplishments:**

* Developed and validated the CSV upload function.
* Built the manual transaction verification interface.
* Integrated the HMM-based fraud detection model.
* Implemented real-time alerting functionality.

**Future Work:**

* API integration for real-time data fetching.
* User-configurable fraud detection thresholds.
* Advanced reporting and admin dashboards.

**Comparison to Initial Plan:**

* All planned milestones, such as CSV uploads and manual transaction verification, have been met.
* Threshold configuration and API integration, initially slated for completion, are now deferred for future enhancements.

#### **Traceability Matrix**

The traceability matrix maps use cases to system requirements, ensuring all functionalities are addressed.

| **Use Case** | **REQ1** | **REQ2** | **REQ3** | **REQ4** | **REQ5** | **REQ6** | **REQ7** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CSV Upload | ✅ |  |  |  |  |  |  |
| Manual Transaction Check |  | ✅ |  |  |  |  |  |
| Alert Notifications |  |  |  |  | ✅ | ✅ |  |

#### **Effort Estimation Using Use Case Points**

**Calculation:**

* Total Use Case Points = 15
* PF = 28 hours/use case point
* Total Effort = 15 × 28 = **420 hours**

#### **Domain Analysis**

The domain model emphasizes entities such as **Transactions**, **Users**, **Alerts**, and **Reports**, showcasing relationships between them. For instance, a "User" uploads "Transactions," which the system analyzes to generate "Alerts" and "Reports."

#### **Interaction Diagrams and Design Patterns**

Updated interaction diagrams implement design patterns such as:

* **Strategy Pattern** for selecting fraud detection methods (HMM or RNN).
* **Observer Pattern** for real-time alert notifications.

These patterns enhance system flexibility, scalability, and responsiveness, ensuring efficient fraud detection and alerting.

#### **Conclusion**

Our fraud detection system addresses pressing challenges in financial security with innovative algorithms and user-centric design. By implementing core use cases and leveraging advanced design patterns, the project lays a solid foundation for future enhancements, such as real-time API integration and customizable detection thresholds. Our approach ensures robust fraud detection capabilities while providing an intuitive and secure user experience.