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

Team Members:

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

Lee Joseph Judkins ; <insert your fhsu email>

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

Instructor: Prof. Hieu Vu

**Revision History:**

|  |  |
| --- | --- |
| Part 1 | 23/09/2024 |

Table of Contents

[Section 1:  Customer Problem Statement 3](#_Toc178014454)

[Section 2:  System Requirements 7](#_Toc178014455)

[Functional Requirements: 7](#_Toc178014456)

[Non-Functional Requirements: 8](#_Toc178014457)

[On-Screen Appearance Requirements: 9](#_Toc178014458)

[FURPS Table: 9](#_Toc178014459)

[Project Management 10](#_Toc178014460)

[References 12](#_Toc178014461)

# Section 1:  Customer Problem Statement

Credit fraud remains a substantial concern for financial institutions worldwide, posing significant risks not only to the banks but also to their customers. The primary challenge lies in the sophisticated techniques employed by fraudsters, who continuously innovate and adapt their methods to exploit vulnerabilities in existing detection systems. Current fraud detection models predominantly rely on static, rule-based algorithms that offer limited flexibility and adaptability, leading to several critical issues.

One of the most pressing problems is the **High False Positive Rate** associated with traditional fraud detection systems. These systems often flag legitimate transactions as fraudulent due to their reliance on generalized pattern recognition and heuristics. For instance, when a customer makes a sizable purchase or travels internationally, the system might automatically trigger a fraud alert, causing unnecessary inconvenience. This not only frustrates customers—who may find their accounts unexpectedly frozen—but also inundates financial institutions with an overwhelming volume of false alarms, complicating the investigative process and straining their resources.

Another significant issue is the **Inability to Adapt** rapidly to evolving fraudulent techniques. Fraudsters are agile and continually refine their tactics, exploiting the inflexibility of older systems that require manual intervention for updates. Rule-based systems typically lack the ability to learn from new data; they operate reactively rather than proactively. This failure to keep pace results in many fraud incidents going undetected until significant damage occurs, often leaving banks scrambling to compensate for losses and restore customer trust.

**Data Volume** presents an ongoing challenge in fraud detection as well. Financial transactions are conducted at an unprecedented scale, with millions of credit card transactions processed daily in digital and physical environments. Conventional systems struggle with the sheer volume of data, often leading to delays in fraud detection and response times. In real-time transaction scenarios, such as those occurring at a retail point of sale or during online purchases, the inability to analyze this vast quantity of data dynamically can mean the difference between stopping fraud in its tracks and allowing a substantial loss to occur.

Existing approaches often lack a granular understanding of individual behavior, struggling to recognize deviations from typical patterns. For example, a customer who occasionally makes large purchases as part of their regular spending behavior might be mischaracterized as a fraud risk, resulting in missed opportunities for legitimate transactions. Consequently, there's a significant need for a solution that can accurately model normal transaction behaviors and effectively identify anomalous patterns.

The scenario is further complicated by historical data limitations; many existing systems are built on fixed rules applied to historical data that may not account for evolving consumer behaviors and emerging fraud tactics. As a result, fraudulent transactions slip through undetected, leading to increased losses for banks and decreased confidence among consumers regarding the safety of their financial transactions.

Our project aims to develop a comprehensive fraud detection system utilizing **Hidden Markov Models (HMM).** The proposed HMM-based system will model the sequential nature of credit card transactions, effectively capturing the temporal patterns that characterize normal customer behavior. Recurrent Neural Networks (RNN) will also be used as a second layer to supposedly increase accuracy rates and reduce false positives. Through this approach, the system will be better equipped to recognize deviations indicative of fraudulent activity, resulting in enhanced detection accuracy, reduced false positive rates, and an overall more responsive system that works in real-time.

The CSV Transaction Data Upload function serves as the primary interface for users to input transaction data into the system. This function allows users to upload CSV files that contain historical transaction records, such as transaction amounts, timestamps, merchant information, and user identifiers. Upon uploading a file, the system performs validation checks to ensure that the data adheres to the required format and that all necessary fields are present. This process not only ensures data integrity but also helps to mitigate errors that could arise from improperly formatted files. Once validated, the uploaded data is stored in a secure database, making it accessible for subsequent analysis. This capability enables the analysis of transaction patterns over time, facilitating the detection of anomalies and trends that could indicate potential fraud.

The Manual Checking of Transactions function provides a user-friendly interface for analysts to evaluate individual transactions for potential fraud. Users can easily search or filter transactions based on various criteria such as transaction amount, date, and merchant. The system presents relevant details alongside analysis metrics derived from the Hidden Markov Model (HMM), which indicates whether a transaction is flagged as suspicious based on learned patterns and historical data. Analysts have the ability to review these flagged transactions in depth, examining factors that may contribute to the suspicion, such as deviations from typical spending behavior or anomalies flagged by the algorithm. This functionality empowers users to make informed decisions while allowing them to take a closer look at transactions that require manual verification before any actions are taken.

The User Authentication function is essential for ensuring secure access to the system and protecting sensitive user data. This process requires users to register for an account by providing personal information and creating a secure password. Upon login, the system authenticates users using industry-standard encryption methods to verify credentials, thus preventing unauthorized access and ensuring that only legitimate users can interact with the system. Additionally, the application may implement multi-factor authentication (MFA) as an added layer of security, requiring users to verify their identity through different means, such as an email confirmation or a one-time SMS code. This robust user authentication process not only safeguards user information but also builds trust among users, ensuring that their sensitive financial data remains protected.

The Alerting Function is a critical component of the fraud detection system that notifies users about suspicious transactions in real-time. When the system detects a transaction flagged as anomalous—based on statistical deviations identified by the HMM—it triggers an immediate notification to the affected user. This alert may be delivered via multiple channels, including email, SMS, or in-app notifications, ensuring the user is promptly informed about potential fraudulent activity. The notification includes details such as the transaction amount, merchant name, and timestamp, allowing users to quickly determine whether the transaction is legitimate or fraudulent. This timely alerting mechanism enhances the user's ability to respond rapidly, such as by freezing their account or contesting the charge with their financial institution, thereby minimizing the risk of financial loss.

By allowing the system to learn from data over time, it can dynamically adapt to evolving fraud schemes. Furthermore, offering insights and reporting tools will empower analysts to investigate flagged transactions efficiently and accurately, ultimately bolstering confidence in the financial institutions’ fraud detection capabilities and improving overall customer satisfaction.

Here is a simplified flow of events in the usage of this program: the usage of the Credit Fraud Detection program begins with the user accessing the application, whether through a web interface or a mobile app. Upon arrival at the main page, users are greeted with a straightforward dashboard that provides clear options for uploading transaction data, reviewing historical transaction records, and checking for suspicious activities. Users must first complete the user authentication process, ensuring that their personal information and transaction data are securely protected. This step may involve logging in with established credentials or creating a new account if they are first-time users.

Once authenticated, the user is prompted to upload their CSV file containing transaction data. The CSV Transaction Data Upload function guides users through selecting and uploading their prepared CSV file. As the file is uploaded, the system performs necessary validation checks, ensuring that all required fields are present and correctly formatted. If any issues arise, users will receive specific error messages to assist them in correcting the problems in their data before re-uploading. After successful validation, the transactional data is processed, and the system begins analyzing it for patterns indicative of potential fraud. This process may run for a considerable amount of time considering the use of RNNs and the HMM.

Following the upload, the user can navigate to the Manual Checking of Transactions section, where they can view a list of all processed transactions. This interface allows users to apply filters based on date, transaction amount, and merchant information to easily find specific records. Here, flagged transactions are highlighted, enabling users to take a closer look at those that may be suspicious. By clicking on a flagged transaction, a detailed view appears that includes the transaction's relevant metrics and the reasons for its flagging, which aids users in making informed decisions about the legitimacy of transactions.

While reviewing transactions, users have a straightforward way to address flagged anomalies. If a transaction is identified as fraudulent, the program’s Alerting Function is triggered. The user receives an immediate notification via email or SMS, providing them with all pertinent details of the suspicious transaction. This proactive alerting mechanism empowers users to act quickly by confirming or disputing the transaction, potentially freezing their account if they suspect fraudulent activity.

Additionally, the user interface provides options for generating reports that summarize findings from the analysis, including statistics on detection rates, false positives, and alerts issued. Users can download these reports for further review, which is especially useful for individuals managing multiple account transactions or for financial analysts who need to present findings within their organization. The reporting function serves as an essential tool for ongoing monitoring and assessment of transaction patterns over time.

As they continue using the system, users can revisit the dashboard to run new analyses or check for updates. The program's architecture allows for efficient handling of repeat uploads and regular schedule processing of transactions as new data becomes available. This routine interaction fosters a dynamic and responsive user experience, ensuring users remain informed about their transaction activities and any potential fraud alerts in real time.

Lastly, the Simulated Fraudulent Transaction feature is designed to provide users with a way to generate fraudulent activities in a controlled environment as part of their testing process. This functionality allows users to mimic the actions of a fraudster who attempts to make a purchase while impersonating the legitimate credit card owner, and ultimately test the functionality of this program itself.

This robust and adaptable fraud detection system will enrich the landscape of financial security, ensuring that customers can engage in transactions with greater peace of mind, while financial institutions can mitigate their losses effectively and efficiently.

# Section 2:  System Requirements

## Functional Requirements:

| **Identifier** | **Requirement** |
| --- | --- |
| **REQ1** | The system shall allow the user to upload transaction data in CSV format for analysis. |
| **REQ2** | The system shall provide a user interface with options to visualize transaction patterns over time, categorized by various parameters such as amount, date, and merchant. |
| **REQ3** | The system shall enable users to set thresholds for alerts regarding fraudulent transactions based on statistical anomalies detected by the Hidden Markov Model. |
| **REQ4** | The system shall include an API to fetch real-time transaction data and historical data from external sources to enhance model accuracy. |
| **REQ5** | The system shall have a reporting feature where users can generate and download reports summarizing detected fraudulent activities and system performance. |
| **REQ6** | The system shall include a user-friendly dashboard displaying key metrics such as false positive rates, detection accuracy, and other relevant KPIs. |
| **REQ7** | The system shall facilitate user authentication to ensure secure access to the application and protect sensitive financial data. |

## Non-Functional Requirements:

| **Identifier** | **Requirement** |
| --- | --- |
| **REQ8** | The system shall be designed with a user-friendly interface that minimizes clutter and promotes ease of navigation. |
| **REQ9** | The system shall maintain high availability, ensuring that the service is operational at least 99.5% of the time to cater to users in need of real-time fraud detection. |
| **REQ10** | The system shall provide mobile compatibility, allowing users to access features on smartphones and tablets without losing functionality. |
| **REQ11** | The system shall utilize efficient algorithms to provide timely alerts regarding potential fraudulent activities with minimal latency. |
| **REQ12** | The system shall ensure data privacy and compliance with relevant regulations (e.g., GDPR, PCI DSS) regarding user data handling and storage. *(This is a regulation for financial data use according to our research and I am including this for the sake of treating this as a real shareholder pitch; Question: should we remove this requirement?)* |
| **REQ13** | The system shall provide an admin interface for monitoring system performance, managing user accounts, and configuring detection parameters. |

## On-Screen Appearance Requirements:

| **Identifier** | **Requirement** |
| --- | --- |
| **REQ14** | The system shall visually represent data using dynamic charts and graphs to illustrate patterns and anomalous transactions effectively. |
| **REQ15** | The system shall maintain a consistent and simple graphical user interface (GUI) across both web and mobile applications, providing an intuitive user experience. |

## FURPS Table:

| **Category** | **Description** |
| --- | --- |
| **Functionality** | The system will leverage Hidden Markov Models for detecting fraudulent credit card transactions, providing features such as transaction uploads and real-time monitoring of anomalies. |
| **Usability** | The system will include a straightforward main page with intuitive navigation to all key functionalities, helping users access important features quickly. All input fields will be clearly labelled for improved user experience. |
| **Reliability** | The system will perform input validation to prevent user errors when uploading transaction data. Version control will manage application updates effectively to prevent data loss and maintain integrity. |
| **Performance** | The application will support multiple users concurrently with minimal latency, ensuring that calculations and alerts are generated efficiently to maintain real-time detection capabilities. |
| **Supportability** | The system will be compatible across popular web browsers. The code structure will be modular, allowing for easy updates and feature expansions. |

# Project Management

Both team members will be actively engaged in the development process, utilizing the GitHub repository to track progress and contributions clearly.

Responsibilities will be delegated based on each member's strengths and expertise, with a shared accountability structure ensuring that all aspects of the project are covered. This is because our group consists of only a pair and thus realistically we will both need input and validation from one another in all development areas.

Below is our projected milestones, with week 1 representing the week beginning in 23 Sept 2024.

| **Week** | **Task Description** | **Responsible Team Members** |
| --- | --- | --- |
| Week 1 | Project kick-off meeting, define scope and objectives | Sergio Kun, Joseph Judkins |
| Week 2 | Research HMM algorithms and relevant literature | Sergio Kun |
|  | Familiarization with data sources and datasets | Joseph Judkins |
| Week 3 | Develop initial design and architecture of the application | Sergio Kun, Joseph Judkins |
| Week 4 | Implement CSV transaction data upload feature | Sergio Kun |
|  | Initial development of the user authentication process | Joseph Judkins |
| Week 5 | Develop the manual checking function for transaction validity | Sergio Kun |
| Week 6 | Implement the alerting function for suspicious transactions | Joseph Judkins |
| Week 7 | Integrate the simulated fraudulent transaction feature | Sergio Kun |
| Week 8 | Testing functionality and debugging | Sergio Kun, Joseph Judkins |
| Week 9 | Develop and integrate reporting tools for transaction summaries | Joseph Judkins |
|  | User interface refinement and user experience enhancements | Joseph Judkins |
| Week 10 | Conduct user testing and gather feedback | Sergio Kun, Joseph Judkins |
| Week 11 | Finalize features based on user feedback | Sergio Kun, Joseph Judkins |
| Week 12 | Prepare project presentation and documentation | Sergio Kun, Joseph Judkins |
| Week 13 | Project review and adjustments based on tutor feedback | Sergio Kun, Joseph Judkins |
| Week 14 | Final project deployment and presentation to the class | Sergio Kun, Joseph Judkins |

# References