Automating Fraud Detection with Explainable and Generative AI Solutions

Pablo Fernandez del Amo, Samuel Williams, Maan Patel, and Pranay Shah

InSync Team

The code for the whole solution, both FronEnd, BackEnd and Generative AI can be found in the next url: https://github.com/Pablxfdez/Infosys-Hackaton.git. All LLMs used in this project were imported from HugginFace API.

1 Introduction

Fraud detection in financial transactions is a critical area of focus for financial institutions worldwide. As digital transactions become increasingly prevalent, the sophistication and frequency of fraudulent activities have also escalated. Detecting and preventing fraud is essential to protect both consumers and financial institutions from significant financial losses. Advanced algorithms and machine learning models play a vital role in identifying suspicious activities and preventing fraud before it can cause substantial harm. This document outlines our approach to building a robust fraud detection system, leveraging state-of-the-art techniques and responsible AI practices to ensure effective and transparent fraud management.

2 State of the Art in Fraud Detection Algorithms

The state of the art in fraud detection algorithms for financial data transactions primarily involves the use of advanced machine learning and deep learning techniques. Models such as XGBoost, Random Forests, and Neural Networks are widely adopted due to their ability to handle large datasets and identify complex patterns. More recently, techniques like Graph Neural Networks (GNNs) and Generative Adversarial Networks (GANs) have been explored to detect sophisticated fraud schemes by modeling relationships between entities and generating synthetic fraud patterns for training robust models. These approaches are often complemented by Explainable AI methods, such as SHAP, to provide transparency and interpretability, which are crucial for regulatory compliance and gaining user trust.

3 Solution

3.1 Database

Fraudulent datasets exhibit unique characteristics that distinguish them from other types of datasets. These include a highly skewed class distribution, where

fraudulent transactions are vastly outnumbered by legitimate ones, creating a significant class imbalance challenge for machine learning models. Additionally, fraud patterns are constantly evolving as fraudsters adapt their strategies, making the datasets dynamic and time-dependent. High-cardinality features, such as IP addresses and user identifiers, also require sophisticated feature engineering techniques to be effective in predictive modeling [1].

The Fraud Dataset Benchmark (FDB) includes several publicly available datasets tailored for various fraud detection tasks:

- **IEEE-CIS:** This dataset contains card-not-present transaction data with 561,013 training instances and 67 features, curated from a Kaggle competition [2].
- **ccfraud:** This dataset consists of anonymized credit card transactions by European cardholders over two days, including 492 frauds out of 284,807 transactions [3].
- **fraudecom:** This dataset includes 150,000 e-commerce transactions with features such as sign-up time, purchase time, and IP address, highlighting the importance of account age in fraud detection [4].
- **sparkov:** A simulated dataset covering transactions of 1,000 customers over six months, designed to study the impact of synthetic data on fraud detection techniques [**?**].

These are some of the datasets used to train and test the model, but in a production environment we will conect directly with a SQL database and predict real time the fraud likelihood of each transaction.

3.2 Call Fraud Detection

The fraud detection section of our project integrates two innovative scripts: one for transcribing speech to text and the other for analyzing the transcribed text for fraudulent content. Both scripts can be found in our GitHub repository.

3.3 Speech-to-Text Model

The first script, `speechtotext.py`, utilizes the Google Speech Recognition API to convert audio input into text. The process involves capturing the audio from a file using the `speech_recognition` library and then transcribing it. The key steps in the script include:

- **Loading the Audio:** The audio file is read and prepared for processing.
- **Transcription:** Using the Google Speech Recognition service, the audio is transcribed into text.
- **Error Handling:** The script includes robust error handling to manage cases where the audio is unclear or the transcription service is unavailable.

This transcription step is crucial as it converts raw audio data into text that can be further analyzed for fraudulent content.

3.4 Fraud Detection Model

The second script, `mistral_fraud_detection.py`, uses a model from the Hugging Face Transformers library to analyze the transcribed text. Specifically, it leverages the Mistral-7B-Instruct model to determine whether the conversation in the text is fraudulent or legitimate. The process involves:

- **Model Loading:** The Mistral-7B-Instruct model and its tokenizer are loaded onto the appropriate device (CPU or GPU).
- **Prompt Engineering:** A prompt is created that instructs the model to analyze the conversation for signs of fraud.
- **Text Generation:** The model generates a response based on the prompt, indicating whether the conversation is fraudulent or legitimate.
- **Result Parsing:** The script parses the model's response to extract the prediction and the justification for the decision.

By combining these two scripts, our solution can process audio data end-to-end: from transcription to fraud analysis. This integration allows us to handle various types of input data and apply advanced machine learning models to detect fraudulent activities efficiently.

For detailed code and further information, please refer to our GitHub repository.

3.5 Transaction Fraud Detection

The Transaction Fraud Detection section focuses on identifying fraudulent transactions within tabular data, a crucial task in financial services. Our solution employs the winning model from the IEEE-CIS Fraud Detection competition on Kaggle, which utilized XGBoost due to its outstanding performance in this domain.

3.6 IEEE-CIS Fraud Detection Competition

The IEEE-CIS Fraud Detection competition on Kaggle aimed to identify fraudulent transactions from a dataset provided by Vesta Corporation. Participants were tasked with developing machine learning models to detect fraud in online transactions, utilizing a wide array of features extracted from transaction data. The competition attracted numerous data scientists and machine learning enthusiasts, leading to innovative solutions and significant advancements in fraud detection techniques.

The dataset included a variety of features such as transaction amount, time, device information, and anonymized categorical features, posing a significant challenge due to the high dimensionality and class imbalance typical in fraud detection tasks. The goal was to maximize the Area Under the Receiver Operating Characteristic Curve (AUC-ROC), a metric that measures the ability of the model to distinguish between fraudulent and legitimate transactions.

3.7 XGBoost Model Implementation

We implemented the XGBoost model, which achieved exceptional results in the IEEE-CIS Fraud Detection competition. XGBoost, or Extreme Gradient Boosting, is a scalable and efficient implementation of gradient boosting that has been widely recognized for its performance and speed in various machine learning tasks. The model's ability to handle large datasets and perform feature selection automatically makes it particularly suitable for fraud detection.

In the competition, the XGBoost model attained an AUC of 0.9677 on the public leaderboard and 0.96 on the private leaderboard, securing the first position. This remarkable performance underscores the model's effectiveness in identifying fraudulent transactions with high accuracy. The key to this success was the sophisticated feature engineering and hyperparameter tuning, which allowed the model to capture intricate patterns in the data.

For a detailed explanation of the model implementation, including feature engineering techniques and hyperparameter optimization strategies, please refer to the Kaggle notebook titled *XGB Fraud with Magic* by Chris Deotte. This notebook provides comprehensive insights into the steps taken to achieve the top performance in the competition. More details can be found at https://www.kaggle.com/code/cdeotte/xgb-fraud-with-magic-0-9600.

The use of this proven model in our solution ensures robust and accurate transaction fraud detection, leveraging state-of-the-art techniques and best practices in the field.

XAI Model In our implementation of the transaction fraud detection model, we have integrated Explainable AI (XAI) techniques using SHAP (SHapley Additive exPlanations) to interpret the model's predictions. Additionally, we employed a smaller, more efficient language model to process and summarize these explanations, making the system more sustainable, faster, and cost-effective.

3.8 SHAP for Model Explanation

SHAP is a powerful tool used to explain the output of machine learning models. It provides a unified measure of feature importance, allowing us to understand how each feature contributes to the model's predictions. In our implementation, SHAP is used to generate explanations for the predictions made by the XGBoost model.

The script loads the pre-trained XGBoost model and uses SHAP to explain its predictions:

- **Model Prediction**: The XGBoost model predicts the probabilities of fraudulent transactions.
- **SHAP Explanation**: SHAP values are computed to understand the contribution of each feature to the prediction. These values indicate the importance and impact of each feature on the model's decision.

3.9 Language Model for Explanation Summarization

To make the SHAP explanations more accessible and less technical, we utilize a smaller language model, `EleutherAI/gpt-neo-125M`, for summarization. By processing the SHAP values through this language model, we can produce user-friendly explanations that highlight which features have the most significant impact on the model's prediction and why. This helps in bridging the gap between the technical model outputs and the end-users who may not have a deep understanding of machine learning concepts.

The implementation involves the following high-level steps:

1. **Prediction and SHAP Values Calculation**: The XGBoost model is used to predict the likelihood of fraud, and SHAP values are calculated to explain these predictions.
2. **Generating Explanations**: The SHAP values are processed using the `EleutherAI/gpt-neo-125M` model to generate a concise, human-readable explanation of the most important features affecting the prediction.
3. **Visualization**: The SHAP values are visualized using force plots and summary plots, which are saved as images for further analysis.

For the full implementation details, including the code, please refer to the `prediction.py` script available in our repository.

This approach ensures that our fraud detection model is not only accurate but also interpretable, providing clear insights into the factors driving each prediction.

3.10 Automated Client Fraudulent Transaction Checking

Our system for automated client fraudulent transaction checking enhances the efficiency and effectiveness of fraud detection by automating the communication process with clients. This allows professionals to focus on genuine cases of fraud, thereby improving overall productivity and accuracy. The process is designed to verify flagged transactions by directly engaging with the client through automated emails.

3.11 Process Overview

1. **Transaction Flagging**: When a transaction is flagged as potentially fraudulent by our detection model, it triggers an automated process.
2. **Automated Email**: An automatic email is sent to the client, asking them to confirm whether the transaction was legitimate. The email includes details of the transaction and provides a simple way for the client to respond.
3. **Client Response Handling**: If the client confirms that the transaction is legitimate, the system unflags the transaction. If the client denies the transaction or does not respond within a certain timeframe, the transaction is reported to the fraud team for further investigation.

3.12 High-Level Implementation

The implementation of this automated process involves the following steps:

1. **Email Sending Mechanism**:
   - The system uses the `smtplib` library to send emails. A predefined email template is used to ensure consistency and clarity in the communication.
   - The `send_email_with_template` function is responsible for creating and sending the email. It connects to the SMTP server, authenticates using the sender's email credentials, and sends the email to the client's email address.
2. **Email Content and Template**:
   - The email template includes the transaction details and asks the client to confirm if the transaction was made by them. This template is designed to be clear and straightforward to facilitate quick responses.
3. **Client Response Processing**:
   - Upon receiving a response from the client, the system updates the transaction status based on the client's input.
   - If the client confirms the transaction, it is unflagged in the system. If the client denies the transaction or does not respond within a set period, the transaction is escalated to the fraud team for manual review.

For the full implementation details, including the code, please refer to the `automatic_response.py` script available in our repository. This approach automates the initial verification of potentially fraudulent transactions, reducing the workload on fraud analysts and improving the speed of the fraud detection process.

On Loop Technician One physical implementation is to have an on-loop technician periodically reviewing random samples from the transaction data to verify whether they have been correctly flagged as fraudulent or legitimate. This ongoing manual check would help identify any failures or inaccuracies in the model, especially in the face of new and evolving fraud tactics, ensuring the system remains robust and effective against emerging threats.

3.13 Feedback and Improvement of the Model

To ensure our fraud detection model remains effective and adaptable to new fraud techniques, we plan to incorporate a feedback loop using the new labels provided by client responses and the continually incoming transaction data. By retraining the model with this updated data, we can improve its accuracy and robustness against evolving fraudulent activities. This approach not only enhances the model's performance but also ensures that resources are allocated responsibly, maintaining efficiency and cost-effectiveness. This retraining process will be part of a future implementation, allowing our system to adapt dynamically to the ever-changing landscape of fraud detection. “‘

3.14 Frontend

The frontend of our system is designed to be an easy-to-use UI that allows users to seamlessly navigate through the data processed by our Machine Learning Model. We used Node.js and React for building the frontend, aiming to provide a smooth and responsive user experience. For connecting the frontend to the Python backend, we initially attempted to use Flask, though there are still aspects that need refinement.

3.15 Sidebar and Navigation

The sidebar is a crucial component, providing quick access to various pages:

- **Overview Page**: This acts as a dashboard for admins to review all accumulated data. Future enhancements will include predictions indicating sectors that might need closer monitoring for fraud based on the current dataset. It also features two bar graphs displaying email fraud occurrences and fraud detected throughout the year, helping predict critical periods for intensified monitoring.
- **Projects Tab**: This section allows admins to monitor employee efforts in identifying new scamming and fraud techniques. It facilitates communication and accountability among team members.
- **Transactions Tab**: This tab displays the most recent transactions. While we did not have time to implement advanced search query functionalities, this feature aims to allow thorough monitoring of all transaction data, ensuring nothing slips past the initial AI model screening.
- **Expenses and Goals Tabs**: These tabs help admins track spending on fraud monitoring efforts. While the AI model effectively detects early fraud, keeping an eye on resource expenditure is crucial for maintaining transparency with clients. Although incomplete, these features are intended to enhance transparency.
- **Settings Tab**: This tab allows users to change usernames, passwords, and manage permissions. Despite being unfinished, the goal is to make it user-friendly for our customers.

3.16 Future Enhancements

Future enhancements will focus on refining the Flask integration for smoother backend connectivity, implementing advanced search functionalities in the Transactions tab, and completing the Expenses and Goals tabs for better financial transparency. Additionally, the Settings tab will be developed to offer more robust user management capabilities.

4 Concerns and Next Steps

There are several concerns we need to address as we move forward with the deployment of our fraud detection model. Firstly, the model has been trained

using an artificial dataset, which may not fully represent the complexities and nuances of real-world data. As a result, the model's performance in real-world scenarios may necessitate adjustments or even a transition to a more powerful model trained on actual transaction data.

Additionally, for a large-scale implementation across an enterprise or multiple banks, significant GPU resources will be required to handle the computational demands of the generative AI components in this ensemble model. While a cloud-based environment could provide the necessary infrastructure, it also entails considerable expenses that need to be carefully evaluated.

In the next steps, we plan to:

- Evaluate the model's performance on real-world data and make necessary adjustments.
- Explore the feasibility and costs associated with a cloud-based hosting environment to ensure scalability and performance.
- Continue to refine and enhance the model based on feedback and new data, ensuring it remains robust and effective in detecting fraudulent transactions.

In the next section we will focus on one of the main concerns we have not explained here, and how to tackle it.

4.1 Privacy and Integration: Federated Learning

An AI model like ours, or any other similar fraud detection model, would significantly benefit from access to transaction data across different banks. This broader data pool is crucial as it allows for the detection of fraudulent patterns that may involve multiple financial institutions, given that a single individual can commit fraud across various accounts.

This integration can be achieved through the Open Banking API, which provides a standardized way for banks to share financial data with authorized third parties securely. The Open Banking initiative aims to increase competition and innovation in the financial services sector by enabling secure and seamless data sharing, thereby allowing more comprehensive fraud detection capabilities.

To maintain the privacy of account data, we propose using a federated learning approach. This method allows each bank to update its local models with new transaction data without sharing sensitive data. Instead, only the updated model parameters are sent to a central server, which aggregates these updates to refine the overall model. This approach ensures that individual transaction data remains private while still benefiting from the collective learning of all participating banks. Federated learning is a strategy employed by leading firms like Meta and Google to train their models while preserving user privacy.

This approach not only enhances the model's effectiveness by leveraging a more extensive dataset but also aligns with stringent data privacy regulations, ensuring that customer information remains secure.

5 Conclusion

In conclusion, our comprehensive fraud detection system integrates cutting-edge AI techniques, leveraging models like XGBoost and SHAP for transaction fraud detection and utilizing innovative automation for client verification. The integration of Explainable AI ensures transparency and interpretability, while our frontend, built with Node.js and React, provides a user-friendly interface for monitoring and managing fraud detection efforts.

We implement responsible AI by ensuring data privacy through federated learning and maintaining transparency with Explainable AI techniques. Our approach is feasible and can be seamlessly integrated within the generative AI and financial structures of Infosys, aligning with their robust technological ecosystem. This solution can be commercialized to banks and other financial institutions, offering a scalable, efficient, and secure method for detecting and managing fraud. By continuously refining the model with real-world data and exploring federated learning for privacy-preserving data integration, we aim to create a scalable and effective solution for financial institutions. Future work will focus on addressing the identified concerns, improving resource management, and maintaining high standards of client transparency and data security.

References

1. Grover, P., Xu, J., Tittelfitz, J., Cheng, A., Li, Z., Zablocki, J., Liu, J., & Zhou, H. *Fraud Dataset Benchmark and Applications.*
2. IEEE-CIS Fraud Detection. Available at: https://www.kaggle.com/c/ieee-fraud-detection/overview.
3. Credit Card Fraud Detection. Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.
4. Fraud ecommerce. Available at: https://www.kaggle.com/datasets/vbinh002/fraud-ecommerce.