Universidad Carlos III de Madrid

Grado en Ingeniería Informática
2022-2023

Trabajo Fin de Grado

Desarrollo de sistemas de detección de fraude con técnicas de inteligencia artificial

Luis Docampo Mateos Pardo

Tutor
Ruben Majadas Sanz

Madrid

Introduction

In the new era of technology, the need to detect and prevent fraud has become increasingly important. Fraud in banking transactions has been a longstanding issue, and its detection is crucial to mitigate financial losses and protect individuals and entities. This work focuses on two types of fraud: the detection of fraudulent accounts in the Ethereum network and the detection of fraudulent transactions. By identifying fraudulent accounts and transactions before any funds are transferred, we can prevent financial losses and ensure the security of transactions.

Fraud detection is a constant challenge in various industries, including finance, insurance, and telecommunications. Machine learning techniques have proven to be effective in detecting fraudulent activities. However, fraudsters continually adapt and devise new strategies to evade detection systems, making fraud detection a complex and evolving problem. To address this challenge, innovative approaches and technologies are needed to stay ahead of fraudsters and protect financial systems.

Motivation

The motivation for this study is rooted in a deep interest in financial systems coupled with the potential of data science. By utilizing machine learning and artificial intelligence in fraud detection, there is an opportunity to bolster the security within financial systems. The digitalization and global integration of financial services have created opportunities for complex fraud schemes. Developing adaptable and robust fraud detection mechanisms is vital to keep ahead and secure transactions. Additionally, tackling fraud is not just about preventing financial loss; it is about building a trustworthy financial environment, which has far-reaching implications on a personal and societal level.

Objectives

To clearly outline the objectives, the following list is provided:

1. Study the state-of-the-art in fraud detection systems.
2. Perform an analysis and preprocessing of the Ethereum fraudulent account dataset.
3. Create a model using machine learning algorithms with the Ethereum fraudulent account dataset.
4. Conduct an analysis and preprocessing of the J.P. Morgan fraudulent transaction dataset.
5. Build a model using machine learning algorithms with the J.P. Morgan fraudulent transaction dataset.
6. Examine the results obtained from both models.
7. Define the implementation approach for the model in the Ethereum blockchain network.

Project Development

The development of the analysis of the data and the results of the model obtained with both data sets is presented in this section.

Model Evaluation Metrics

To address the problem of detecting fraudulent accounts, it is important to focus on specific metrics:

- Recall (Sensitivity or True Positive Rate): This metric measures the proportion of actual fraudulent transactions that the model correctly identifies.
- Precision: Precision quantifies the proportion of predicted fraudulent transactions that are actually true positives.
- F1-Score: The F1-Score combines recall and precision into a single metric using the harmonic mean. It provides an overall measure of the model's ability to detect fraud while minimizing false alarms.

The choice of the primary metric depends on the relative importance of detecting all fraudulent accounts. In this case, the F1-Score is considered essential as the cost of missing fraudulent transactions can be significantly higher than falsely identifying legitimate accounts as fraud. However, if precision is relatively acceptable, focus can be placed on recall. If the recall of fraudulent accounts is high while precision is low, it suggests that the model is labeling a large number of legitimate accounts as fraudulent, making it unreliable.

Ethereum account data analysis

The dataset used in this project was obtained from Kaggle, utilizing Ethereum Etherscan API, etherscamdb API, and web scraping techniques. It consists of 9,840 entries and 50 attributes. Initially, the index column, which provides no relevant information, was removed. Following that, columns with zero variance were eliminated, as they do not contribute any useful information to the model.

The distribution of the FLAG column, which indicates whether an account has fraudulent transactions, is as follows: 7,662 entries are marked as 0 (non-fraudulent), while the remaining 2,179 rows are considered fraudulent. To address bias in the model, techniques were employed to balance the fraudulent and non-fraudulent entries. It is worth noting that the majority of entries are non-fraudulent, which aligns with the nature of the problem.

Columns related to ERC20 data typically have fewer distinct values, as most accounts primarily use the Ethereum main network rather than the secondary ERC20 network. However, the total number of transactions sent and received by each account, whether through Ethereum or ERC20, were summed and added as two new columns: Total unique received and Total unique sent.

After removing duplicate rows, the dataset was reduced from 9,841 to 9,012 entries. Attributes with higher correlations are often ERC20 metrics, such as mean, maximum, or minimum values of transactions sent or received. Some Ether-related attributes, such as avg val received and max value received, also exhibit correlations, although around 0.6 compared to the near-perfect correlations of ERC20 values. Additionally, a strong correlation exists between attributes related to contracts, including the value sent to contracts and the number of contracts created. After this and after studying the distributions of the attributes, they have been discretized, for the model training it has been used the SMOTE technique to balance the classes.

Results obtained with the Ethereum data set

The results obtained are the following ones:

Algorithm	Class	Precision	Recall	F1-score	Macro Average Precisión	Macro Average Recall	Macro Average F1-Score
Logistic Regression	0	0.99	0.94	0.97			
	1	0.74	0.92	0.81	0.84	0.93	0.89
KNN (Neighbour=4)	0	0.98	0.94	0.96			
	1	0.69	0.88	0.78	0.84	0.91	0.87
Random Forest	0	0.97	0.99	0.98			
	1	0.92	0.79	0.85	0.94	0.89	0.91
Naives Bayes (Bernouilli)	0	0.98	0.92	0.94			
	1	0.62	0.85	0.72	0.80	0.89	0.83
Decision tree (Max Depth=4)	0	0.98	0.89	0.93			
	1	0.55	0.87	0.80	0.76	0.88	0.80
Support Vector Machine	0	0.98	0.98	0.98			
	1	0.88	0.86	0.87	0.93	0.92	0.92
XGBoost	0	0.97	0.99	0.98			
	1	0.92	0.84	0.88	0.95	0.92	0.93

Tabla 1: Results obtained with the Ethereum data set.

The best result has been obtained with XGBoost, parameter optimization have been done to find the best configuration. The results obtained with XGBoost are highly satisfactory, with the model demonstrating effective performance in detecting fraudulent accounts. It achieves a sensitivity of 84.5 % in identifying fraudulent accounts, and when classifying an account as fraudulent, it is accurate 95 % of the time. The precision for non-fraudulent accounts is 97.5 %, while the precision for fraudulent accounts is 92.4 %. These metrics indicate the model's ability to correctly classify both types of accounts. Overall, the model exhibits an f1-score of 93 %.

Transactional data analysis

These data contain a wide variety of transaction types representing normal activities as well as abnormal/fraudulent activities introduced with predefined probabilities. The data was generated by executing an AI planning-execution simulator and translating the output planning traces into tabular format. Initially, the dataset consists of 13 attributes and 1,498,178 rows. The dataset is considerably imbalanced, with 97.94 % of transactions labeled as not fraud and 2.06 % labeled as fraud. The majority of transactions consist of normal payments, followed by fast payments, fund movements, and the least frequent type being salary transfers. This distribution can be attributed to individuals receiving monthly salaries but conducting various types of movements and payments throughout the month. Regarding the countries of origin and destination for transfers, it is confirmed that the majority of transfers originate from or are directed to accounts in the United States, which is expected as it is a U.S.-based bank. For the purpose of later using the One-Hot Encoding technique, most countries have been grouped by continents, except for the United States, Germany, and Canada, as they have the highest number of transfers and can provide valuable information. This approach avoids creating a new column for each country, making the model computation more efficient.

Feature Engineering

The feauteres obtained from the originals are the following ones:

- **Bene ClienteJP:** Binary variable indicating if the beneficiary is a client of J.P. Morgan Chase.
- **Sen ClienteJP:** Binary variable indicating if the sender is a client of J.P.

Morgan Chase.

- **National Transaction:** Binary variable indicating if the transaction is domestic.
- **Time Diff Last Trans Sen:** Time difference between current and last transaction by the same sender account.
- **Time Diff Last Trans Ben:** Time difference between current and last transaction with the same beneficiary account.
- **Average Diff Time Rec:** Cumulative average time difference between consecutive transactions received by each account.
- **Average Diff Time Sen:** Cumulative average time difference between consecutive transactions sent by each account.
- **Average USD Sent:** Cumulative average amount in USD sent by each sender account.
- **Average USD Rec:** Cumulative average amount in USD received by each beneficiary account.
- **Max USD Sent:** Maximum amount in USD sent by each sender account.
- **Max USD Rec:** Maximum amount in USD received by each beneficiary account.
- **Min USD Sent:** Minimum amount in USD sent by each sender account.
- **Min USD Rec:** Minimum amount in USD received by each beneficiary account.
- **Time step:** Column converted to date and time format.

- **Sender Transactions Last 1/7/30 Days:** Number of transactions sent by the sender account in the last 1, 7, and 30 days, respectively.
- **Bene Transactions Last 1/7/30 Days:** Number of transactions received by the beneficiary account in the last 1, 7, and 30 days, respectively.

Results obtained with the transactional J.P. Morgan data set

The results obtained are the following ones:

Algorithm	Class	Precision	Recall	F1-score	Macro Average Precision	Macro Average Recall	Macro Average F1-Score
Logistic Regression	0	0.99	0.59	0.74			
	1	0.04	0.73	0.07	0.51	0.66	0.40
KNN (Neighbour=4)	0	0.98	0.89	0.93			
	1	0.05	0.27	0.08	0.52	0.58	0.51
Random Forest (Estimators=50)	0	0.98	0.93	0.95			
	1	0.06	0.20	0.09	0.52	0.57	0.52
Naives Bayes (Bernouilli)	0	1.00	0.21	0.34			
	1	0.03	0.98	0.05	0.51	0.59	0.20
Decision Tree (Max Depth=5)	0	1.00	0.24	0.39			
	1	0.03	1.00	0.05	0.51	0.62	0.22
XGBoost	0	0.98	0.88	0.92			
	1	0.03	0.31	0.09	0.51	0.60	0.51
XGBoost & Logistic Regression	0	0.98	0.94	0.96			
	1	0.06	0.19	0.10	0.52	0.60	0.53

Tabla 2: Results of J.P Morgan data set experiments

The ensemble model, consisting of Logistic Regression and XGBoost using a Voting Classifier, showed mixed results in detecting fraudulent bank transactions. While it achieved high precision for non-fraudulent transactions (98 %), the recall for fraudulent transactions was low (28 %). The F1-Score for fraudulent transactions was also low (11 %), indicating the need for significant improvements in detecting fraud. Although the overall accuracy was 90 %, the model's performance was influenced by the class imbalance in the dataset. To enhance the model's effectiveness, further improvements such as resampling techniques, exploring different algorithms, or incorporating additional attributes should be considered.

Implementation of the Ethereum model in a smart contract

To detect fraudulent accounts in Ethereum, an off-chain machine learning model is employed. The model resides on an external server, which is crucial as executing it directly on the blockchain would be impractical and costly. A smart contract on Ethereum gathers necessary data, which can be obtained from blockchain events and transactions, and potentially enriched with historical or additional information through APIs. An oracle serves as a bridge for communication between the smart contract and the external machine learning model. The smart contract sends a request with input data to the oracle, which then passes the data to the model. The model processes the data and sends back its prediction (whether an account is fraudulent or not) through the oracle to the smart contract.

The smart contract plays a vital role; it not only collects data but also communicates with the external model via the oracle. Once the smart contract receives the prediction, it processes the information and, if the account is detected as potentially fraudulent, records this information on the blockchain. It is essential to have the smart contract optimized regarding resource consumption to ensure economic viability on the Ethereum blockchain. Besides, extensive testing and ensuring the security of the smart contract are paramount, especially considering the serious implications of labeling accounts as fraudulent. Lastly, the machine learning model may require updates and maintenance over time. It is crucial to have strategies for managing these updates and ensuring that the oracle functions correctly and that the system's integrity is maintained.

Conclusions

This project has been a fulfilling learning experience in data science techniques, such as data mining and machine learning. The steep learning curve has offered invaluable insights into the complexity of fraud detection. However, the endeavor faced challenges, particularly with computational times and mixed results in the J.P. Morgan dataset. Real-world data can be messy, necessitating adaptability and revised strategies. Comparing models between Ethereum and J.P. Morgan highlighted the importance of contextual understanding and custom approaches. This project was not only about achieving results but also about honing critical thinking and problem-solving skills in data science, which are essential for future projects. The experience has underscored the importance of continuous learning in this ever-evolving field, and the insights gained will be invaluable for future innovation. Overall, this project marks a significant milestone in my development as a data scientist.

Conclusions for the Ethereum model

In summary, the XGBoost model gave good results, though there's room for improvement, possibly through ensemble techniques. However, due to the focus on the J.P. Morgan dataset, this project did not allocate much time to fine-tune the Ethereum model. The Ethereum dataset may not be very realistic as it's outdated by two years, and the Ethereum blockchain has since evolved. Caution is advised if using the model for classification as its reliability is questionable. Comparing with results from Kaggle, it's evident that preprocessing plays a significant role. One notable Kaggle work had better XGBoost performance, possibly due to not discretizing the data, which can cause information loss affecting complex models like XGBoost more than simpler ones like Logistic Regression.

Conclusions for the J.P. Morgan model

The results obtained using this dataset did not meet expectations. One of the key issues was the large size of the dataset, which made it computationally demanding to work with. For instance, running the KNN algorithm took a substantial amount of time (8 hours and 38 minutes), and even then, the results were unsatisfactory. Other methods such as ensemble techniques and neural networks were considered, but they either did not offer a significant improvement or were too resource-intensive to be practical with the computing power available.

Additionally, it is believed that the dataset was not comprehensive enough. Specifically, it lacked historical account data, which is often essential for effective fraud detection. The absence of historical data deprived the model of important contextual information. For example, knowing an account's typical transaction patterns would help identify deviations that might indicate fraud.

The historical data is also important for recognizing how fraud patterns change over time (temporal trends), and for grouping similar accounts together to better understand typical behaviors (account grouping).

Given these challenges, there is still room for improvement. It's possible that an alternative approach to the problem or incorporating more comprehensive data could yield better results. The combination of transaction data with historical account data might enable the model to discern more complex patterns and behaviors, which is critical for effective fraud detection.