Universidade do Minho
Escola de Engenharia

Ana Paula Fernandes Martins

Fraud Detection and Anti-Money Laundering applying Machine Learning Techniques in Cryptocurrencies Transactional Graphs

July 2023

Fraud Detection and Anti-Money Laundering applying Machine Learning Techniques in Cryptocurrencies Transactional Graphs

Ana Paula Martins

UMinho | 2023

Universidade do Minho
Escola de Engenharia

Ana Paula Fernandes Martins

Fraud Detection and Anti-Money Laundering applying Machine Learning Techniques in Cryptocurrencies Transactional Graphs

Master's Dissertation
Integrated Master's in Engineering and Management of Information Systems

Work performed under the supervision of
Miguel Abrunhosa de Brito

July 2023

COPYRIGHT AND TERMS OF USE OF THE WORK BY THIRD PARTIES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my heartfelt gratitude to all those who have supported and guided me throughout the process of completing this dissertation.

First and foremost, I am immensely grateful to my advisor, Miguel Abrunhosa de Brito, for his unwavering support, invaluable insights, and guidance. His expertise, encouragement, and dedication have played a crucial role in shaping this research and ensuring its success.

I would also like to extend my sincere appreciation to Uphold, the company that provided me with the opportunity to conduct this research. I am grateful to Diogo Pinto and Paulo Marques, my supervisors at Uphold, for their continuous support, valuable input, and guidance throughout the research process. Their expertise and encouragement have been instrumental in enriching my understanding and ensuring the relevance of my work.

Furthermore, I would like to express my deepest gratitude to my family for their unwavering support, understanding, and encouragement throughout this journey.

Lastly, I would like to offer a very special thanks to my boyfriend for his unwavering support, patience, and companionship throughout the entire process. His belief in me, understanding of the challenges I faced, and unwavering support were invaluable and played a pivotal role in my success.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Assinado por: **Ana Paula Fernandes Martins**
Data: 2023.07.29 15:55:20 +0100

ABSTRACT

Fraud Detection and Anti-Money Laundering applying Machine Learning Techniques in Cryptocurrencies Transactional Graphs

Cryptocurrencies have revolutionized financial transactions, but they have also introduced new opportunities for fraudulent activities. To combat this, anti-money laundering regulations and machine learning techniques are crucial for detecting suspicious transactions. Machine learning facilitates in-depth analysis, reduces false positives, and strengthens fraud defenses. This study explores using graph-based machine learning models to detect fraud in cryptocurrency transactions. By analyzing networks and using graph-based models, hidden patterns can be uncovered, fraudulent behavior can be identified, and anti-money laundering measures can be improved. Graph-based models are effective in detecting complex relationships and anomalies, making them valuable for uncovering fraudulent activity in transactional graphs. The research aims to compare traditional machine learning methods with graph-oriented approaches to determine which approach performs better in identifying illegal transactions, supporting regulatory compliance, and enhancing risk management.

Therefore, in this research, three traditional approaches were implemented: Random Forest, Logistic Regression, and Multilayer Perceptron. These methods have been widely used and provide a solid baseline for comparison. Additionally, more advanced graph-based models were implemented, specifically the Graph Convolutional Network and a variation of it called the Skip-Graph Convolutional Network. Node embedding techniques were also employed to investigate whether they could enhance the performance of baseline methods.

The results of this study revealed that the Random Forest algorithm outperformed all other methods, including the graph-based machine learning models, GCN, and Skip-GCN. Additionally, the application of node embedding techniques did not significantly improve the performance of the baseline machine learning methods. Despite the potential of node embeddings to capture latent features and enhance predictive capabilities, their implementation did not yield substantial improvements in this study.

Keywords: Anti-Money Laundering; Cryptocurrencies; Machine Learning; Graphs; Embeddings

RESUMO

Deteção de Fraude e Anti Lavagem de Dinheiro aplicando Técnicas de Aprendizagem de Máquina em Grafos Transacionais de Criptomoedas

As criptomoedas revolucionaram as transações financeiras, mas também introduziram novas oportunidades para atividades fraudulentas. Para combater isso, regulamentos para combate à lavagem de dinheiro e técnicas de aprendizado de máquina, que facilita a análise aprofundada, reduz os falsos positivos e fortalece as defesas contra fraudes, são cruciais para detetar transações suspeitas.

Este estudo investiga o uso de modelos de aprendizado de máquina baseados em grafos para detetar fraudes em transações de criptomoedas. Esses modelos analisam as redes de transações e identificam padrões ocultos e comportamentos fraudulentos, melhorando as medidas de combate à lavagem de dinheiro. Comparando com métodos tradicionais de aprendizado de máquina, as abordagens baseadas em grafos são eficazes na deteção de relacionamentos complexos e anomalias, sendo valiosas para descobrir atividades fraudulentas em redes transacionais. O objetivo da pesquisa é determinar qual abordagem apresenta o melhor desempenho na identificação de transações ilegais, no suporte à conformidade regulatória e no aprimoramento da gestão de riscos.

Portanto, nesta pesquisa, foram comparadas três abordagens tradicionais: *Random Forest*, *Logistic Regression* e *Multilayer Perceptron*, com dois modelos baseados em grafos: *Graph Convolutional Network* e uma variação chamada *Skip-Graph Convolutional Network*. Também foram exploradas técnicas de incorporação de nós para melhorar o desempenho dos métodos tradicionais.

No entanto, os resultados mostraram que o algoritmo *Random Forest* teve um desempenho superior a todos os outros métodos, incluindo os modelos baseados em grafos. Além disso, a aplicação das técnicas de incorporação de nós não trouxe melhorias significativas.

Palavras-chave: Anti Lavagem de Dinheiro; Criptomoedas; Aprendizagem de Máquina; Grafos; Incorporação de nós

INDEX

List of Abbreviations and Acronyms

AD	Anomaly Detection
AML	Anti-Money Laundering
ANN	Artificial Neural Network
DSR	Design Science Research
FD	Fraud Detection
FP	False Positive
FN	False Negative
FPR	False Positive Rate
FNR	False Negative Rate
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NRL	Network Representation Learning
ReLU	Rectified Linear Unit
RF	Random Forest
SAR	Suspicious Activity Report
SVM	Support Vector Machine
Skip-GCN	Skip-Graph Convolutional Network
sklearn	Scikit-Learn
TN	True Negative
TP	True Positive
TPR	True Positive Rate
t-SNE	t-Distributed Stochastic Neighbor Embedding
UTXO	Unspent Transaction Output

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Cryptocurrencies have become increasingly popular in recent years, with some experts arguing that they may be among the most promising fintech innovations of the decade. Their use has been expanding and has been applied to a wide range of activities. Now, major companies are beginning to accept payments in cryptocurrencies as an alternative to traditional fiat currencies. This suggests that cryptocurrencies have the potential to fundamentally change market mechanisms and financial services (Eigelshoven et al., 2021).
As a result, research in this area is becoming increasingly important and has been receiving significant attention in recent years.
The problem and motivation that gave rise to this work, as well as the established objectives, will be discussed subsequently.

1.1 Problem

As cryptocurrencies gain widespread acceptance and usage, the decentralized nature of blockchain technology provides inherent security. However, the unregulated nature of the cryptocurrency market creates opportunities for criminals to exploit vulnerabilities and engage in illicit activities such as fraud, and money laundering. This poses a significant challenge for organizations and authorities to detect and prevent such illegal activities effectively, specifically in the area of Fraud Detection (FD).
The evolving problem lies in the need for robust measures and strategies to combat money laundering and other illicit activities within the cryptocurrency ecosystem. Traditional Anti-Money Laundering (AML) controls, which have been established in the traditional financial system, are now essential to be implemented in cryptocurrency-oriented businesses. However, distinguishing fraudulent transactions from legitimate ones is not always straightforward, as criminals often mimic normal transaction behavior to evade detection. Moreover, financial criminals are continually evolving their methods to exploit the anonymity and decentralized nature of cryptocurrencies. This necessitates the constant development and implementation of innovative techniques and tools to stay ahead of these criminals and protect the integrity of the cryptocurrency market.

1.2 Motivation

This dissertation is sponsored by Uphold, a company that provides financial services through an online platform for the transaction of digital assets such as cryptocurrencies. This makes it a complex business with many players, where identifying anomalies and understanding user interactions is of great value. As a company that handles cryptocurrencies on a daily basis, Uphold is at risk of being used to commit illegal transactions which makes it vulnerable to illegal transactions that can compromise the trust of its users and put the company in a vulnerable position regarding compliance with laws, regulations, and industry standards.

To address this challenge, integrating Machine Learning (ML) algorithms into traditional AML presents a promising avenue for research. ML techniques can analyze large volumes of transactional data, extract meaningful patterns, and identify suspicious activities more effectively. These algorithms can adapt to changing market trends and improve the effectiveness of AML efforts.

Further, the utilization of transactional data on blockchain technology provides new prospects for transaction network analysis. This can be achieved through the creation of visual representations of the transaction networks using graphs. These graphs highlight valuable insights into the relationships between nodes, making them ideal for ML on graph data.

However, traditional ML methods may have difficulty in effectively handling the complex and non-linear connections present in graph data. Thus, utilizing deep learning methods, such as neural networks, in combination with graph analysis can potentially identify complex relationships within the data and detect fraudulent activities in the cryptocurrency market. The adaptability and evolution of deep learning methods make them particularly suitable for detecting fraud in the dynamic and ever-changing cryptocurrency market. In short, by focusing on the inherent structure and interconnections within the transaction network, graph-oriented approaches have the potential to provide novel insights and more accurate detection of illicit activities. Examining the effectiveness of these methods will shed light on their potential to enhance the detection and prevention of money laundering in the cryptocurrency domain.

1.3 Dissertation Objectives

Considering the aforementioned problems and motivations, this dissertation focuses on assessing the feasibility of utilizing ML and graph analysis techniques to detect fraudulent transactions within the cryptocurrency network. The objective is to evaluate the practical significance of uncovering fraudulent activities through ML methods that leverage the graph structure for classification purposes. Furthermore, the study aims to gain valuable insights into the potential of deep learning graph analysis methods, which could be effectively applied to Uphold's proprietary datasets. By efficiently identifying illegal transactions, these methods would support regulatory compliance and enhance risk management activities.

To achieve these objectives, the following goals will be pursued:

1. Conduct comprehensive research on key concepts such as fraud and money laundering, as well as the measures that companies must implement to effectively combat these issues.
2. Perform a literature review to explore the utilization of ML and graph analysis for detecting fraudulent transactions. This review should encompass recent studies, including the techniques, datasets, and findings related to fraudulent behavior, such as commonly observed anomalies and patterns.
3. Obtain a representative dataset of cryptocurrency transactional data to serve as the basis for the analysis.
4. From the literature review, select a study that provides a suitable framework to evaluate the feasibility of using ML and graph analysis in the context of AML.
5. Select a set of algorithms to be implemented and gain a thorough understanding of their functionalities and applicability within the experiments.
6. Implement a baseline by utilizing several supervised traditional ML models to compare and evaluate their performance against graph-oriented approaches. This will serve as a benchmark for assessing the effectiveness of using graph analysis techniques in detecting fraudulent transactions within the cryptocurrency network.
7. Explore the role of feature engineering in relation to graph-related features and analyze its impact on the performance of the models.
8. Evaluate the impact of using graph-oriented deep learning methods in order to understand the effectiveness of detecting fraud in cryptocurrency transactions, considering the unique characteristics of graph data.

9. Evaluate the feasibility of employing node embeddings, a graph-oriented approach, to detect fraudulent transactions within the cryptocurrency market. Analyze the performance of node embedding techniques in capturing important graph properties and assess their effectiveness in accurately identifying illegal activities in cryptocurrency networks.
10. Discuss the applicability and effectiveness of ML and graph analysis models on Uphold's dataset for detecting fraudulent transactions within the Uphold cryptocurrency network. Analyze their practical significance in supporting regulatory compliance and enhancing risk management, outlining the required resources, data preprocessing steps, and model training and evaluation processes.

1.4 Research Methodology

In this dissertation, the investigation methodology employed is rooted in the principles of DSR. DSR is an appropriate methodology for addressing practical issues and generating new knowledge and artifacts that can be utilized to enhance the design of existing systems or develop new ones. The ultimate outcome of DSR is frequently a new artifact, such as a tool, system, or method, that can be applied to improve the well-being of individuals or organizations. The specific problem addressed in this dissertation pertains to FD in cryptocurrencies using graph techniques.

The DSR methodology includes the following phases:

1. **Problem identification and formulation:** The initial phase of this research project involves the identification of a problem or opportunity that can be addressed through systematic investigation. This phase includes the definition of the problem, the comprehension of its context, and the formulation of clear and specific research objectives.
2. **Literature review:** In this phase, the researcher conducts a literature review to gain an understanding of existing solutions, identify gaps in current knowledge, and ensure that the proposed solution is novel. The literature review helps to establish the relevance and feasibility of the research and serves as a foundation for the design and execution of the research, providing a basis for comparison with the results obtained. It is an important step in the research process that helps to ensure the research is grounded in existing knowledge and contributes to the advancement of the field.

3. **Design and development:** This phase of the research process involves designing and developing a solution to the problem at hand. This includes the creation of a prototype or proof of concept, as well as the definition of the methods and techniques that will be utilized to evaluate the solution. The design and development of the solution is a critical step in the research process, as it helps to ensure that the proposed solution is practical, feasible, and effective.
4. **Evaluation:** This phase of the research process is dedicated to testing and evaluating the solution developed in the previous phase. The objective of this phase is to determine the effectiveness and potential impact of the solution, this can be achieved by conducting user testing, experiments, or other forms of evaluation. The results of these tests and evaluations will provide valuable insight into the performance of the solution and will inform any necessary adjustments or improvements to the design.
5. **Dissemination and Implication:** This phase involves communicating the results of the research and its implications to the relevant stakeholders. This can include publishing the research in academic journals, presenting it at conferences, or creating a user manual.

Following there's a description of this work's objectives that are related to each one of the DSR phases.

1. **Problem identification and formulation:** To ensure that the research is comprehensive and yields valuable and relevant outcomes, it is imperative to commence by clearly defining the problem that initiated the research and outlining the motivation for addressing it. Following that, is necessary to establish an objective, and a plan to solve the problem, which includes specific and measurable steps, should be laid out to achieve that objective. Furthermore, it is essential to conduct in-depth research on the context and related concepts of the topic to fully comprehend the field of exploration and be aware of the current state of the art. This involves understanding the concepts of cryptocurrencies, fraud, AML, and graph analysis.
2. **Literature review:** This phase implicates a thorough review of existing literature in the same field of experimentation to gain insight into previous solutions proposed and issues identified about the application of ML techniques and graph analysis in the field of AML. This includes identifying and describing relevant and related works that have been

conducted in the past, as well as identifying areas that warrant further exploration in the current study.

3. **Design and development:** In this phase of the research, after conducting a thorough literature review, a methodology is proposed to attain the objectives of the study and to add to the existing body of knowledge on the utilization of ML techniques and graph analysis in the field of AML. This methodology includes an examination of the specific ML and graph analysis techniques that will be employed, the selection of an appropriate dataset, and the establishment of a pipeline for implementation. Following the establishment of the methodology, it will be implemented.
4. **Evaluation:** The proposed methodology will undergo a thorough evaluation process, in which its results will be compared with those of a previous study that employed similar techniques. The comparison will be based on metrics that objectively measure the performance of the techniques on the identification of fraudulent transactions and will be used to determine the effectiveness of the proposed solution in comparison to the previous studies.
5. **Dissemination and Implication:** In conclusion, the findings of the study will be presented in the form of a master's dissertation document at the university. Furthermore, an article will be written and submitted for publication.

1.5 Document Structure

This dissertation is structured into nine chapters, each focusing on different aspects of the research.

Chapter number two lays the foundation with a Theoretical Framework, which offers an overview of key concepts relevant to the dissertation. It covers topics such as cryptocurrencies, fraud, AML, and graphs.

Moving forward, in chapter number three, a comprehensive literature review is conducted to explore the utilization of ML in AML and the application of graph analysis and learning within the same domain.

Chapter number four provides an in-depth explanation of the tools and materials employed in the research. This includes a detailed description of the tools used, the dataset utilized, and its corresponding features.

In chapter number five, the focus shifts to the ML algorithms used in the research. Each algorithm is thoroughly explained, along with its functioning principles.

Chapter number six offers a detailed account of the conducted experiments. It covers the preprocessing procedures and provides an explanation of the experimental setup. Additionally, the implementation details of each model, including the parameters used, are discussed.

The findings of the experiments and their implications are presented in chapter number seven. This chapter includes a comprehensive discussion of the results obtained.

Chapter number eight explores the practical application of the developed methodology on Uphold's dataset, shedding light on its effectiveness and potential impact.

Lastly, chapter number nine concludes the dissertation by summarizing the key findings, highlighting their significance, and discussing potential avenues for future research.

2. THEORETICAL FRAMEWORK

2.1 Cryptocurrencies

Cryptocurrencies are defined as a digital representation of value, issued by a private developer instead of a central bank, credit institute, or e-money institute and can be used as an "alternative form of money" (Bank, 2015).

Blockchain is the underlying technology behind cryptocurrencies. It is a decentralized, peer-to-peer network that utilizes various cryptographic techniques to create a secure, append-only database. The consensus protocol allows for the addition of verified and confirmed information to the blocks, which are then distributed to all participant nodes in the network (Tang et al., 2023). In this way, cryptocurrencies do not require a central authority to process and settle transactions, unlike traditional transactions. Instead, transactions are verified within a decentralized network through cryptographic techniques and the transaction data are then stored immutably on a distributed ledger (Gandal et al., 2018). This happens as the input of a signed transaction contains both the hash value of a previous transaction and an index to locate the output since it is a reference to an output from a previous transaction. This guarantees that transactions are not easily changed or damaged (Tang et al., 2023).

Cryptocurrencies, due to their decentralized and peer-to-peer structure, offer several advantages over traditional monetary systems. These include lower transaction costs, increased transactional security, and privacy, and higher transaction efficiency (Rejeb et al., 2021). The transparency of cryptocurrency transactions is one of its key features, as all transactions are recorded on the publicly available blockchain (Trozze et al., 2022). Additionally, they offer a high degree of privacy as the transactions are not directly linked to a specific user or organization (Tang et al., 2023) being pseudo-anonymous in the sense that one might know the address of a user but cannot know exactly who he or she is (Yuan et al., 2018). Blockchain technology, by serving as a public ledger, enables the utilization of analytical tools for the purpose of searching for transactions associated with a specific address. This includes the capability to search and validate related transactions stored within a specific wallet address (Tang et al., 2023).

2.2 Fraud

Khrestina et al. (2017) define fraud as attempts to launder proceeds obtained illegally or first-party fraud or even fraudulent debit card charges. Zhou et al. (2021), on the other hand, define fraud generally as an illegal or criminal deception aimed at obtaining financial or personal benefits. As the adoption and usage of cryptocurrencies increase, it becomes crucial to comprehend and investigate the different types of fraud that occur within this system to combat and prevent it effectively.

While the immutability of transactions on the blockchain has the potential to greatly reduce fraud (Tang et al., 2023), the cryptocurrency market remains vulnerable to fraud and scams. This is due to its attractiveness to financial crime because of its lack of controls (Astrakhantseva et al., 2021) resulting from the pseudo-anonymity (Sabry et al., 2020), not existing a central third party to manage the cryptocurrency and regulate the market (Eigelshoven et al., 2021), and neither existing any jurisdiction to enforce the law (Astrakhantseva et al., 2021).

As well, according to Trautman (2014), criminals prefer virtual currencies because they offer the greatest degree of anonymity for both users and transactions, the ability to move illicit proceeds quickly and confidently from one country to another, widespread adoption, and trustworthiness.

Brenig et al. (2015) disagree with the notion that criminals prefer cryptocurrencies solely for their technical design features and instead argue that it is the contextual and transactional factors that make them attractive for money laundering. They found that instantaneous transactions, decentralization, and pseudo-anonymity are positive incentives for criminals, while limited acceptance and high price volatility are negative incentives. However, it is worth noting that as cryptocurrencies continue to gain wider adoption, the limited acceptance were viewed by the author as a negative incentive for criminals may gradually transform into a positive incentive, potentially amplifying their attractiveness for illicit activities.

The study by Trozze et al. (2022) highlights that the cryptocurrency markets are vulnerable to various types of fraud and scams, which is a growing concern worldwide. The authors conducted a literature review to understand the types of fraud that have already been identified and those that might occur in the future. They identified 29 kinds of fraud schemes that might occur, such as Ponzi Schemes, phishing scams, and money laundering, and grouped

them into four categories, including those considered harmful by experts, those in the public sector, those in the private sector, and those in the academic sector. The authors emphasized the need for further research to address the problems in the future.

2.2.1 Money Laundering

It is important to highlight that many identified and studied cases of fraud involve the use of cryptocurrencies as a means to carry out illegal activities (Brenig et al., 2015). Cybercriminals exploit the vulnerabilities of cryptocurrencies, particularly in money laundering, which involves the conversion of illicit funds obtained through criminal activities. These funds are then mixed with legitimate money to create the appearance of legality (Korejo et al., 2021).

Law enforcement agencies globally recognize the use of cryptocurrencies in various criminal activities, mostly correlated with economic and financial spheres, in particular money laundering (Dyntu & Dykyi, 2018). The use of cryptocurrencies in money laundering activities allows criminal organizations to seamlessly transfer funds while evading detection by authorities, enabling them to conduct their illegal activities with greater success and less likelihood of being caught (Albrecht et al., 2019). This makes it increasingly difficult for law enforcement agencies to trace and investigate these illegal transactions (Mabunda, 2018).

Even though transactions made using cryptocurrencies are not directly linked to a specific user or organization, they can be traced through their associated addresses (Trozze et al., 2022). As a result, some users turn to alternatives such as Monero, which offers more privacy or use "mixers" or "tumblers" to obscure the origin of their funds. These methods can form the basis for money-laundering schemes on the blockchain.

The largest case of money laundering to date is that of Liberty Reserve, which in 2013 was accused of laundering a staggering 6 billion US dollars. The organization conducted 55 million transactions using their virtual currency, "Liberty Dollars" but converted and stored the funds in fiat currency (i.e., USD) at the beginning and end of each transaction. This allowed them to evade detection and launder the funds using a virtual currency while still ultimately storing the funds in a traditional, government-backed currency (Dyntu & Dykyi, 2018).

2.2.2 Handling Fraud

Given the growing threat of fraud, organizations, particularly those handling cryptocurrencies like Uphold, must take action to detect and prevent these deceitful activities. This can be accomplished by utilizing existing data and implementing effective controls, which will help to safeguard the organization and its customers against financial losses and other negative impacts of fraud.

Anomaly detection (AD) is a key technique for addressing fraud, as it involves identifying unusual items, events, or observations that raise suspicions by deviating significantly from most of the data. FD is based on identifying anomalies and suspicious behavior in transactions and trade history.

However, criminals often attempt to conceal their fraudulent activities by mimicking normal transactional behavior (Lorenz et al., 2020), making it difficult to detect fraudulent activity as an anomaly. As fraudsters continue to develop new methods for committing fraud, it is essential to regularly improve these techniques to keep pace with their evolving tactics (Sabry et al., 2020). Additionally, due to the decentralized and transparent nature of blockchain technology, detecting fraudulent activity by rogue users is a challenging task to analyze, as it is hard to trace the generation of money and profits in such markets (Sureshbhai et al., 2020).

2.3 Anti-Money Laundering

Chen et al. (2018) emphasize that given the significant financial losses that result from money laundering, organizations must implement effective AML systems, controls, and practices to mitigate the risk of money laundering activities.

AML is a set of activities that involve the examination of customers' transactions by analysts and automated systems. The goal is to identify suspicious behaviors that may be linked to money laundering (Labanca et al., 2022).

AML solutions are a component of overall fraud control measures, and they help to automate and reduce the manual workload of the screening and checking process. AML solutions typically include a combination of software, databases, and analytical tools that can be used to monitor and analyze financial transactions for suspicious activity (Chen et al., 2018). This includes monitoring transaction patterns, performing customer due diligence, and implementing procedures to detect and report suspicious activities. Such measures can help

organizations identify and prevent money laundering and to comply with legal and regulatory requirements.

Traditionally, AML regulations require banks and other financial institutions to perform due diligence to monitor and prevent customers from performing transactions that may be involved in money laundering (Mabunda, 2018). This helps to ensure that banks are proactive in the fight against money laundering.

However, the decentralized and transparent nature of cryptocurrencies and blockchain technology makes it much harder to implement traditional AML practices in this case. The pseudonymous and decentralized nature of blockchain transactions makes it difficult to track the origin of the funds, making it harder to identify and prevent money laundering activities.

Traditional AML systems are designed to monitor unusual behaviors and detect potential money laundering activities. However, as money laundering techniques and financial crime are always evolving, the rules and algorithms used by these systems must be updated to ensure that they are able to capture these changes (Labanca et al., 2022). Additionally, these systems are based on pre-determined rules and are only able to detect known unusual behavior, which can result in False Negatives (FN) if unknown strategies for performing money laundering are used (Labanca et al., 2022). Furthermore, the use of specific static thresholds in these systems can result in a high number of False Positives (FP) (Labanca et al., 2022).

Therefore, to effectively detect money laundering activities and comply with AML regulations, there is a need to develop new methods and technologies that can adapt to the ever-evolving nature of money laundering and reduce the number of FP and FN.

2.4 Machine Learning

ML plays a crucial role in the field of AML by helping to identify and prevent financial crimes. Overall, ML models can be used to identify unusual correlations and assess the effectiveness of money laundering detection algorithms, making them a valuable tool in the fight against financial crime. But it is crucial to understand the strengths and limitations of each algorithm when applied to the problem of detecting money laundering. This can help ensure that the appropriate model is used for a given task and that the results are interpreted correctly.

There are three main types of ML models: supervised, unsupervised, and semi-supervised learning. Supervised ML is trained on labeled data and is used to make predictions on new,

unseen data. Unsupervised ML is trained on unlabeled data and is used to find patterns or structures in the data without any prior knowledge of the labels. Supervised learning algorithms can also be used to classify cryptocurrency transactions as suspicious or non-suspicious based on historical data. Unsupervised learning algorithms can be used to detect patterns in the data that may indicate money laundering activity. Semi-supervised combines both labeled and unlabeled data during training. It utilizes a small amount of labeled data and a larger amount of unlabeled data to learn patterns and structures in the data. Semi-supervised learning can be beneficial in cases where obtaining labeled data is costly or time-consuming, as it can leverage the unlabeled data to improve the model's performance.

Further, Han et al. (2020) suggest that using AI and graph learning can help enhance current AML solutions. These techniques can analyze financial data, identify patterns and behaviors that may indicate money laundering activities, and help to improve the accuracy and efficiency of AML systems (Sabry et al., 2020). ML can decrease the False Positive Rate (FPR), while keeping the False Negative Rate (FNR) relatively low, to overcome the drawbacks of traditional AML systems (Vassallo et al., 2021). This can help to detect and prevent money laundering activities involving cryptocurrencies, which pose privacy and security threats.

The blockchain is an effective resource for this study as it allows for the public accessibility of transaction data. This facilitates the verification of the correlation between account operations and the identification of fraudulent activities (Ostapowicz & Żbikowski, 2019). Utilizing the publicly accessible data within the blockchain, researchers can gain valuable insights into the issue of money laundering.

When evaluating the performance of an ML model for AML, it is important to take into consideration metrics such as Precision, Recall, specificity, and F1-Score. These metrics can provide insight into the model's ability to accurately identify money laundering. The choice of metrics used for evaluation depends on the model and whether it is in the training or testing phase. Accuracy is often the most highlighted metric by authors when evaluating the performance of their models. However, Chen et al. (2018) pointed out that there is an extremely large volume of raw data that is highly imbalanced. This naturally happens because fraudulent activities are rare events and anomalous transactions represent only a small fraction of the total number of transactions (Y. Zhang & Trubey, 2019). This presents a significant issue as it affects the accuracy of the model. A high accuracy does not necessarily indicate a useful model, as it could incorrectly predict everything as legitimate, resulting in

apparent correctness without true effectiveness. Therefore, in this case, it's more useful to analyze the trade-offs between Recall and FNR.

In summary, AI and graph learning can improve AML systems by analyzing financial data and identifying money laundering patterns. ML can also decrease FP and FN to overcome the limitations of traditional AML systems and help detect and prevent money laundering activities involving cryptocurrencies.

In the following section, a comprehensive analysis of graph-based techniques and their potential applications in combating money laundering is presented.

2.5 Transactional Graphs

Graphs are commonly used data structures, and they are able to model several real-world systems, such us economic networks or social networks. They provide a level of abstraction suitable for describing complex data belonging to different fields, and to capture intrinsic relationships present within the data. Any real-world network can be modeled as a graph G defined as a tuple of two sets: the first set V consists of vertices (nodes), and the second set E consists of pairs of vertices called edges (links), representing the connection among two vertices, $G = (V, E)$. Graph structures represent valuable information as they incorporate existing relationships among vertices.

According to a study by Song et al. (2023) it was found that analyzing the information contained within the cryptocurrency transaction network can be an effective method for detecting illegal financial activities involving cryptocurrencies. This can include graph analysis and graph learning.

While graph analysis refers to the process of studying and interpreting the relationships and connections between entities within a graph data structure, graph learning, on the other hand, refers to the application of ML techniques to graph data. Graph learning can learn the deep representation of node features directly in the topology of the network graph, which avoids time-consuming feature engineering and enhances the feature representation of nodes, making it beneficial for FD in financial transactions (Li et al., 2022).

Graph Anomaly Detection (GAD) is a subfield of graph learning, as it utilizes graph representations and graph algorithms to identify anomalies in graph-structured data and aims to identify anomalous nodes, edges, or sub-graphs, expanding the concept of AD to graphs

(Ma et al., 2021). Although, traditional AD techniques are not suitable for detecting anomalies in graph data due to the complex, high-dimensional nature of the data. As a result, researchers have turned to deep learning techniques, which have been shown to be effective in representing and recognizing patterns in graph data (Ma et al., 2021).

Graph learning can include various facets such as Node Classification, Graph Classification, Node Clustering, Link Prediction, and Influence Maximization (Daigavane et al., 2021). Graph learning and analysis can provide a deeper understanding of relationships and interactions within the cryptocurrency ecosystem.

Utilizing graph-based methods allows for the identification of patterns and connections that may not be apparent with traditional ML algorithms. These methods are typically based on or extend deep learning methods, as they can effectively transform graph data into vector representations for analysis and manipulation (Xia et al., 2021).

One type of graph used in this context is a transaction network graph, which is a vectorial space representation of the nodes and connections (edges) between them (Song et al., 2023). In the Unspent Transaction Output (UTXO) blockchain, such as with Bitcoin, these graphs can be divided into three types: address networks, transaction networks, and user networks. The graph-based representation enriches the data by incorporating the relations between nodes and facilitates data analysis related to link prediction, node classification, forensic investigation, and the relationship between entities (Tharani et al., 2021).

Graph neural networks (GNNs) are a family of neural networks that can operate on graph-structured data, allowing for the extraction of high-level representations of graphs explicitly (Daigavane et al., 2021). This enables various graph analytic tasks such as classification, recommendation, and clustering to be performed effectively (Kim et al., 2023). One of the most explored technique in this context is the Graph Convolutional Network (GCN), a neural network that operates on local graph neighborhoods, involving a learnable kernel to output the node embeddings (Alarab et al., 2020b).

2.6 Representation Learning for Graph Analysis

Representation learning is a fundamental concept in ML that aims to automatically discover and extract meaningful representations or features from raw data. Unlike traditional

approaches that rely on handcrafted features, representation learning empowers models to learn these representations directly from the data itself (Bengio et al., 2013).

The primary goal of representation learning is to transform data into a more informative and compact representation space, where important patterns and relationships can be more easily discerned. By capturing relevant features automatically, representation learning enhances the performance of downstream tasks (D. Zhang et al., 2018).

In the context of graphs, Network Representation Learning (NRL) techniques focus on uncovering hidden structures and characteristics present within the graph data. This enables a more comprehensive understanding of the underlying graph properties and facilitates more effective analysis and reasoning.

2.6.1 Traditional Strategy

When working with traditional ML methods on graph data, the input is typically a graph, and the goal is to extract features that describe the topological structure of the network around a specific node. This process is known as feature engineering. These topological features, such as the node's degree and centrality, can be combined with attribute-based information to train a classical ML model.

Figure 1 represents the traditional strategy when applying traditional ML strategies to graphs.

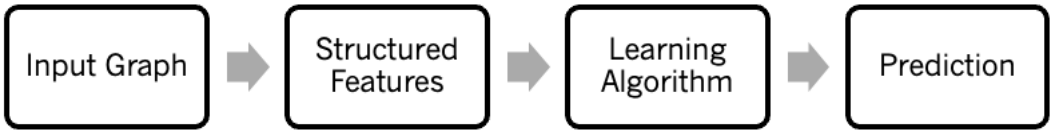

Figure 1: Traditional ML Strategy for Graphs

While the traditional strategy for graph analysis has been widely employed, it has certain limitations. The process of manual feature engineering can be time-consuming, labor-intensive, and heavily reliant on expert knowledge. Additionally, these handcrafted features may not fully capture the complex and nuanced patterns present in the graph data, limiting the performance of traditional ML models.

2.6.2 Network Representation Learning

To overcome these limitations, NRL emerges as a solution by automating the feature extraction process. Instead of relying on explicitly designed features, NRL algorithms automatically learn to extract informative features directly from the graph data. This enables the models to capture the underlying patterns and relationships present in the graph, leading to more effective and accurate analysis.

2.6.3 Node Embeddings

Node embedding, a specific type of NRL, plays a pivotal role in graph analysis. It involves mapping each node in a graph to a dense, low-dimensional vector known as an embedding.
By representing nodes as continuous vectors, node embeddings enable the exploration of complex graph structures and semantic relationships. They facilitate the integration of both topological and attribute-based information, leading to more effective analysis tasks such as node classification, link prediction, and graph clustering.
Graph embedding techniques, such as DeepWalk or Node2Vec, have grown in popularity as a means for learning latent representations of vertices on large-scale networks. These techniques effectively reduce the data dimension of the transaction network and transform high-dimensional one-hot node vectors into dense low-dimensional node vectors (Lin et al., 2020).
Despite the success of these techniques, they have some limitations. For example, DeepWalk and Node2Vec only consider the structural information of the graph and do not incorporate temporal information, such as changes and evolution over time (Lin et al., 2020). This could be crucial for cryptocurrency applications, making it a crucial aspect to consider in graph representation studies.
The GCNs are also used for node embeddings. GCNs use graph convolution operations to aggregate information from neighboring nodes and update the node representations. This enables effective learning of node embeddings that capture both the structural and attribute-based information in the graph.

3. LITERATURE REVIEW

This literature review focuses on the use of ML algorithms for detecting fraud and money laundering. It also examines the application of graph learning and analysis in the same field. The concepts discussed in the previous section are used to support and clarify the information presented in this review.

3.1 Machine Learning for Fraud Detection and Anti-Money Laundering

This chapter focuses on the traditional approaches of ML used for FD and AML.

Chen et al. (2018) surveyed the literature on ML techniques for AML solutions in the suspicious transaction detection area and found that efficient data preparation, data transformation, and data analytics techniques are necessary for an AML solution to offer data quality assurance, high detection accuracy, scalability, and fast performance in suspicious transaction detection. They also found that various ML models, including fuzzy logic, Support Vector Machine (SVM), clustering, outlier detection, neural networks, genetic algorithms, Bayesian network, and sequence matching have been tested to better evaluate suspicious transactions in the banking sector.

Considering the security concerns surrounding cryptocurrencies, Choithani et al. (2022) conducted a study to evaluate the effectiveness of AI and ML algorithms in risk management within the banking sector and predict cryptocurrency behavior. Through their research, the authors found that utilizing SVM, Long Short-Term Memory (LSTM), and Artificial Neural Networks (ANN) yielded superior results compared to other techniques examined.

Bhowmik et al. (2021) studied various ML models for FD in the blockchain and found that Logistic Regression (LR), Multilayer Perceptron (MLP), SVM, and Random Forest (RF) yield very good detection rates.

Kamišalić et al. (2021) reviewed studies that examine the potential benefits of combining blockchain technology with data mining techniques for anomaly and FD. They found that the most used methods were Gradient Boosting, SVM, and RF. Additionally, they identified a recent trend toward the use of Deep Learning (Neural Networks, and LSTM) for these applications, which corroborates the graph learning approach explored in this work.

Most of the research in this field utilizes ML models on the Elliptic Dataset, which has been a valuable asset to the AML and research communities (Alarab & Prakoonwit, 2022) and was created by Weber et al. (2019). The dataset, provided by cryptocurrency intelligence company Elliptic, is a graph network of Bitcoin transactions and is one of the largest labeled datasets available for any cryptocurrency.

Ostapowicz & Żbikowski (2019) and Fawaz Ibrahim et al. (2021) proposed models to detect fraudulent accounts on the Ethereum network and classify them as "Fraud" or "Non-Fraud" using ML. Both studies found that the RF algorithm had the best results in terms of Recall and false-positive rate. However, Ostapowicz & Żbikowski (2019) noted that the Recall of 84.92% was not suitable for use in a real-world anti-fraud system. Additionally, Alarab et al. (2020a) note that it is not possible to optimize Recall and Precision simultaneously, as this can lead to an increase in FP.

The supervised methods have been widely explored in this research field as many authors have been proving that these have good results, particularly RF. However, the use of these methods is limited by the availability of large-scale labeled datasets (Labanca et al., 2022). i.e., when applying supervised methods, it's important to have historical data with the events accurately assigned, this means that the datasets need to be properly labeled (Y. Zhang & Trubey, 2019).

The labeling process is costly and time-consuming, as it requires manual annotation of historical transactions, and the complexity of money laundering schemes makes it difficult to identify all the entities involved (Lorenz et al., 2020). Even the Elliptic Dataset, which is considered one of the largest labeled datasets available, has only 2% of transactions classified as illicit and 21% as licit.

In light of these limitations, Lorenz et al. (2020) proposed using unsupervised methods and active learning to detect money laundering in datasets with few labels. By using a training dataset with just a few hundred labels (5% of the total), the authors obtained similar performance as the best-supervised methods.

Zhang & Trubey (2019) used ML and sampling schemes (to deal with unbalanced data) to detect money laundering activities in financial transactions and observed that ANNs are the ones that perform better.

Due to the increasing amount of available data in the blockchain, also increases the difficulty in searching for preferent data and datasets (Tang et al., 2023).

Chen et al. (2018) also pointed out that there is an extremely large volume of raw data that is highly imbalanced. To address this issue, the authors suggested a clustering process to handle the large amounts of available data or use semi-supervised techniques. Yang et al. (2019) used a heuristic user address clustering method to cluster user pseudonyms and propose a user address update algorithm. They mined the user characteristics, classified different types of user features, and distinguished between normal users and abnormal user features. They used the Gaussian Mixture Model to cluster users and identify users with suspected anomalies based on the characteristics of the user classification. Therefore, the authors were able to verify the suspicious user and prove that there are abnormal users among suspicious users of the experiment.

As well, Monamo et al. (2017) used k-means clustering and a trimmed version of the algorithm to detect fraudulent activity in the Bitcoin transactions network. Both algorithms achieved optimal clustering. The trimmed algorithm was able to detect five of thirty well-known anomalies. Using k-d trees, they were able to detect two more anomalies. Based on the clustering labels for outliers, the authors employed some supervised classification models to understand the relation between the labels and predictor variables. RF achieved the best Precision.

In conclusion, the literature reviewed suggests that traditional ML algorithms such as SVM, RF, and ANNs have been extensively tested for FD in the financial sector, especially for cryptocurrency transactions. However, these methods have limitations in the form of high costs for manual labeling, the difficulty of detecting rare events, and the challenge of balancing Recall and FPR. On the other hand, graph learning has been proposed as an alternative approach. These approaches show promising results, as demonstrated by recent studies. Nevertheless, more research is needed to explore the full potential of these methods in real-world anti-fraud systems. The following section explores the use of graph learning in the context of AML.

3.2 Graph Learning and Analysis for Detecting Fraud and Money Laundering

In recent years, several techniques such as GCNs, DeepWalk, and Node2Vec have emerged as potential solutions for encoding topological structures from graphs into dense

representations. These techniques aim to place nodes with similar neighborhoods close to the embedding space (Khazane et al., 2019).

Motamed & Bahrak (2019) explored the properties of five different cryptocurrencies. They aimed to propose algorithms for extracting transaction-related features from Bitcoin-core and Ethereum. The authors generated three different types of graphs, including the Transaction graph, the Address graph, and the Money flow graph, to visualize the data. They found that the address graph and the transaction graph of Bitcoin, as well as the money flow transaction graph of Ethereum, were most suitable for visualizing anomalous transactions. They emphasized the importance of these visualizations for the future application of ML algorithms in detecting fraudulent transactions. Sait Canbaz et al. (2020) also acknowledged the significance of data visualization in FD and proposed a web-based software tool for detecting financial fraud through graph representation.

Alarab et al. (2020b) proposed a novel approach that combined GCN with linear layers to perform node classification on the transaction graph. The approach utilized two sets of features, with the first set being node embeddings obtained from GCN and the second set obtained from the latent representation of a linearly transformed hidden layer from the original features. The results of the study indicated that the combination of features derived from GCN, and the latent representation of a linear layer improved the performance of the model in comparison to using graph convolutions alone.

Alarab & Prakoonwit (2022) addressed the limitations of previous studies that used GCN for detecting illicit transactions in the financial industry. These studies did not consider the temporal aspect of transactions, leading to limitations in the model's accuracy. The authors proposed a new model that combined LSTM and GCN and used active learning methods to label unlabeled data. The results showed that the new model outperformed previous studies, with an accuracy of 97.77% and an F1-Score of 80%. The integration of LSTM was a key innovation in the study, as it considers the temporal sequence of transactions and improves the performance of the model.

Singh et al. (2021) aimed to tackle the issue of temporal biasing in the Elliptic Dataset by proposing a GNN-based adversarial loss architecture. The results showed that the GNN-based adversarial loss architecture was effective in addressing the problem of temporal biasing, leading to improved performance compared to previous studies.

Weber et al. (2019) conducted a study on AML in Bitcoin. The study aimed to address the challenge of accurately identifying criminal activity in large, growing datasets of Bitcoin transactions while minimizing the FPR. The researchers used traditional ML models (RF, LR, and MLP) and compared their performance to that of GCN. The results showed that RF outperformed LR, MLP, and GCN, making it a more effective method for classifying Bitcoin transactions as illicit. However, the study highlights the potential of graph-based approaches, as they may offer valuable insights into this field. Alarab et al. (2020a) applied ensemble learning, a combination of various supervised learning techniques, to the Elliptic Dataset and demonstrated that it outperforms other methods, including the one proposed by Weber et al. (2019).

Geng et al. (2022) introduced a GAT, an improved version of the GCN, to enhance the traceback accuracy for identifying illegal behavior. The results of the study showed the potential of the GAT as an effective approach for detecting illegal activities in blockchain networks. The findings of this study contribute to the growing body of literature on utilizing graph learning techniques for addressing security issues in blockchain systems.

Pocher et al. (2022) conducted a study comparing the performance of the GCN and Graph Attention Network (GAT) with that of the baseline approaches, such as RF proposed by Weber et al. (2019). Their results showed that GCN outperforms the baseline approaches, suggesting the potential value of GCN for this type of research.

Lin et al. (2020) developed a graph embedding method that integrates temporal and weighted information from financial transaction networks into node embeddings and applied it to an Ethereum network for phishing and non-phishing node classification. The study highlights the importance of considering temporal and weighted information in financial transaction networks for FD.

Caglayan & Bahtiyar (2022) proposed an approach to detect money laundering in financial transactions using Node2Vec, a graph embedding algorithm. The results were evaluated by calculating True Positives (TP), FP, True Negatives (TN), and FN on a banking transactions dataset containing money laundering transactions. The study showed that Node2Vec provided better results in detecting money laundering compared to other methods. Similarly, in the study by Lopes et al. (2022), Node2Vec was used to obtain vector representations of nodes and edges of criminal networks. The combined use of Node2Vec and predictive ML methods were found to be effective in distinguishing between criminal, non-criminal, and

mixed relationships in criminal networks. These studies highlight the potential of Node2Vec as a valuable tool in the analysis of financial and criminal transactions.

Further, Zhou et al. (2021) presented a distributed big data approach for detecting Internet financial fraud. The approach utilized Node2Vec to represent the topological features of the financial network graph as low-dimensional dense vectors. The resulting representation was then used to classify and predict financial fraud with the help of a deep neural network. The study showed that the proposed approach improved the efficiency of financial FD with better Precision, Recall and F1-Score. These findings suggest the potential of utilizing graph embedding and deep learning for effective and efficient Internet financial FD.

Li et al. (2022) proposed a novel method called TA-Struc2Vec for detecting fraudulent users in financial transaction networks. The method considers both structural and transaction amount homogeneity in the network and employs ML classification algorithms to classify users. The proposed approach, TA-Struc2Vec, was shown to learn stronger structural homogeneity and further transaction amount homogeneity compared to previous methods. The results of the study indicate the potential of considering both structural and transaction amount information in detecting fraudulent users in financial transaction networks.

Yu et al. (2022) found that phishing nodes in Ethereum networks can be identified by their rapid transfer of funds obtained from multiple inputs. To detect such fraud, the authors proposed a GCN approach and evaluated its performance using transaction records to construct a labeled graph for classification. These findings suggest that graph-based methods have the potential for detecting phishing fraud in cryptocurrency networks and may also be applicable to other types of fraud.

The reviewed literature underscores the importance of graph representation techniques in detecting fraudulent transactions. The use of GCN, DeepWalk, Node2Vec and their variants (e.g., GNN-based adversarial loss architecture, GAT) has demonstrated potential in improving FD detection rates. Integrating temporal and weighted information and combining graph representation with deep learning enhances the performance of FD models. These findings provide a basis for future research and showcase the potential of graph representation in addressing security concerns in financial and criminal transactions.

4. MATERIALS

This chapter delves into the fundamental materials utilized in the dissertation research, providing a comprehensive overview of the tools, Python libraries, and datasets employed throughout the study.

Emphasizing the collaborative nature of the project, the dataset selection, pre-processing, and exploratory data analysis were accomplished as a joint effort within the Uphold team. In addition, a fellow team member was involved in another fraud-related use-case in the context of a dissertation. With a common set of supervisors, it was inevitable that there would be some convergence in the materials employed and the exploration of those resources.

4.1 Development Tools

The programming language to be utilized for this dissertation is Python. Python is a popular choice for ML due to its simplicity, readability, and large community of developers. It has a wide range of libraries and frameworks that make it easy to implement various ML algorithms and techniques. These libraries provide pre-built functions and tools for tasks such as data processing, model training, and evaluation, which makes it easy for developers to quickly prototype and test their models. Overall, Python's ease of use, community support, and rich ecosystem of ML libraries make it an ideal choice for developing ML projects.

The programming environment utilized was Jupyter Notebook. Jupyter Notebook is an interactive computing platform that allows for the creation, execution, and sharing of code, along with the incorporation of explanatory text, visualizations, and multimedia content. Its web-based interface and support for multiple programming languages, including Python, make it a popular choice among researchers and data scientists. With its interactive and collaborative features, Jupyter Notebook promotes an iterative and exploratory approach to analysis, enhances reproducibility, and facilitates efficient communication of findings.

The main libraries used for this work are described next.

1. **Scikit-learn:** Scikit-learn (sklearn) is a widely used Python library for ML and data analysis. It offers a comprehensive suite of tools and algorithms that facilitate the entire ML workflow, from data preprocessing to model training and evaluation. In this

dissertation, sklearn was used to effectively address the challenges of predictive modeling and classification tasks. By leveraging its extensive collection of ML algorithms, such as RF classifier and LR classifier, different modeling approaches and techniques were explored. Sklearn's evaluation metrics, including Precision, Recall, and F1-Score, allowed to assess the performance of the models.

2. **Torch:** Torch is a widely used open-source ML framework that provides comprehensive tools and functionalities for building and training neural networks. Developed primarily by Facebook's AI Research lab, Torch offers a rich ecosystem for deep learning research and applications. In this work, the torch library was utilized to define and train neural network models, including the implementation of activation functions (e.g., Rectified Linear Unit (ReLU)) and the Adam optimizer for optimizing model parameters.
3. **Torch Geometric:** Torch Geometric is an extension library specifically designed for handling and analyzing graph-structured data within the PyTorch ecosystem. It offers efficient data structures and functions tailored to the unique characteristics of graph data. In the context of this research, the torch geometric library played a crucial role in processing graph-structured data and implementing GCNs. The library's Data class facilitated the storage and manipulation of input graph information, such as node features, edge indices, and labels. Moreover, the GCNConv module enabled the definition of GCNs, a fundamental component for NRL employed in the GCN models.
4. **Karate Club:** Karate Club is a Python library that offers graph embedding algorithms for unsupervised ML tasks on large-scale graphs. Developed to facilitate research in the field of graph analysis, Karate Club provides state-of-the-art techniques for learning low-dimensional representations of nodes in a graph. It encompasses algorithms such as DeepWalk and Node2Vec, which leverage random walks on graphs to capture the structural properties and relationships between nodes. By generating node embeddings, Karate Club enables the exploration and analysis of complex graph data, allowing for various downstream applications. In this dissertation, the Karate Club library was employed to extract informative node embeddings from graph data.

The supporting libraries used follow.

1. **Pandas:** Pandas is a versatile Python library utilized for efficient data manipulation and analysis. It offers powerful data structures, such as DataFrame and Series, which enable seamless handling of structured data. In the dissertation, pandas played a significant role in managing and preprocessing the datasets. By utilizing pandas' functions and methods, the data could be cleaned, transformed, and aggregated effectively. Additionally, pandas provided convenient tools for filtering, slicing, and merging datasets, facilitating the extraction of meaningful insights and preparation of the data for further analysis.
2. **Numpy:** NumPy is a fundamental library in Python used for numerical computations. It provides efficient data structures, including multi-dimensional arrays, along with a comprehensive set of mathematical functions and operations. In the dissertation, NumPy was employed for various purposes. Firstly, it served as a robust framework for storing and manipulating numerical data efficiently. Secondly, the library's mathematical functions were utilized for performing computations on the data, such as calculating descriptive statistics, applying mathematical transformations, and generating random numbers for simulations. NumPy's powerful array operations, such as element-wise operations and broadcasting, facilitated the manipulation and analysis of numerical data.
3. **Matplotlib:** Matplotlib is a versatile Python library widely used for creating visualizations and plots. It offers a broad range of plotting functions and customizable options, allowing for the creation of high-quality visual representations of data. In the dissertation, Matplotlib was instrumental in visualizing various aspects of the analysis. Line plots, scatter plots, bar charts, and histograms were generated to visualize patterns, relationships, and distributions in the data. Matplotlib's ability to customize plot aesthetics, including titles, labels, colors, and styles, aided in effectively presenting the results in a visually appealing and informative manner. Incorporating Matplotlib in the dissertation facilitated the communication of complex findings and supported the interpretation of the analysis results.
4. **NetworkX:** NetworkX is a widely used Python library designed to create, manipulate, and analyze complex networks and graphs. It provides a comprehensive set of tools

and algorithms for studying the structure, dynamics, and functions of networks. NetworkX offers a flexible and intuitive framework for representing and working with graph data, making it a valuable resource for network analysis in various domains. It provided essential functionalities for constructing, manipulating, and analyzing graphs in this dissertation. It facilitated operations such as adding edges, setting node attributes, and applying graph algorithms, enabling comprehensive network analysis and visualization.

5. **Seaborn:** Seaborn is a Python data visualization library that builds on Matplotlib to provide a simplified and high-level interface for creating visually appealing and informative statistical graphics. It offers a wide range of predefined plot types and styles, as well as integrated statistical algorithms, making it easy to generate complex plots and visualize relationships between variables. In the dissertation, Seaborn played a vital role in improving the analysis by producing visually compelling plots and enhancing the interpretation and communication of research findings.

4.2 Dataset

In the initial phase, real Uphold data was not available to conduct the research. Then, one of the main points to develop the methodology was to find an online available dataset, that was viable to conduct the project.

The chosen dataset was the Elliptic Dataset. This dataset is the world's largest labeled transaction dataset publicly available in any cryptocurrency and was created to motivate and enable the development of new techniques for detecting illicit cryptocurrency transactions.

The dataset primarily focuses on Bitcoin transactions, which is the most well-known and secure cryptocurrency. Moreover, after an intensive search, this is the most complete dataset that allows for achieving the goals of this dissertation since it is graph-based, which makes it adequate for its needs. Additionally, Bitcoin is the cryptocurrency where most studies focus their attention, it’s the most widely known cryptocurrency, and it’s the most secure one.

When someone wants to send Bitcoin to another person, they refer to the previous transactions where they received the Bitcoin as inputs. These inputs represent the funds they possess. By using cryptographic techniques, the sender verifies their ownership of these inputs and creates new outputs, assigning specific amounts of Bitcoin to the intended

recipients. These outputs serve as references for future transactions, allowing the recipients to spend the Bitcoin they receive. This system, known as the UTXO protocol, ensures the integrity and transparency of transactions within the Bitcoin network. Figure 2 illustrates the behavior of the UTXO protocol.

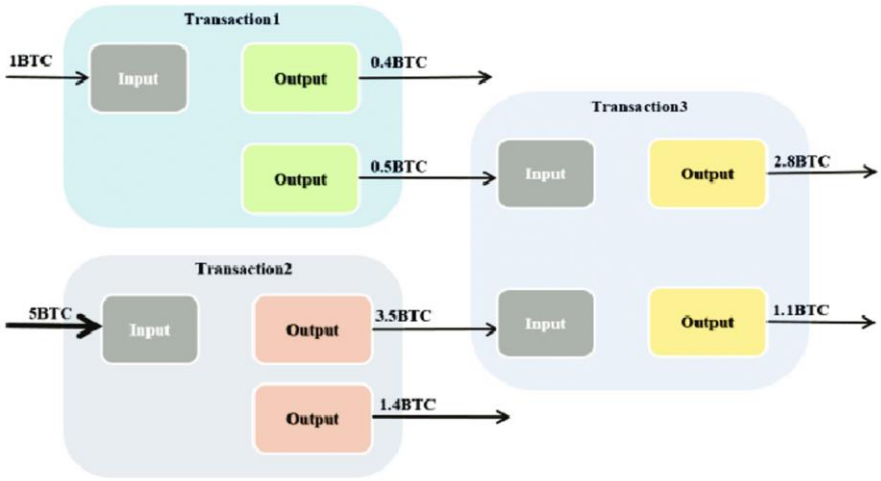

Figure 2: Bitcoin UTXO Protocol (Xue et al., 2021)

The Elliptic Dataset represents these transactions as graphs, where each node represents a transaction, and the edges represent the flow of transactions between them. Each transaction in the dataset is labeled as "licit," "illicit," or "unknown".

Each node is associated with a time step, indicating when a transaction was broadcasted in the Bitcoin network. The time steps range from 1 to 49 and are evenly spaced, with approximately two weeks between each step. Each time step represents a graph instance consisting of a connected component of transactions that occurred within a three-hour timeframe. It's important to note that there are no edges connecting the different time steps, meaning that the graphs within each time step are not interconnected.

The complete dataset is made of 203,769 nodes and 234,355 edges. 2% (4,545) of the nodes are labeled class 1 (illicit). 21% (42,019) are labeled class 2 (licit). The remaining transactions are labeled as unknown, which corresponds to 77%.

Therefore, the dimensions of the graph can be described as:

- V = 203,769 transactions (nodes)
- E = 234,355 edges

Each node has 166 features, with the first 94 features representing local information about the transaction – including the time step, number of inputs/outputs, transaction fee, output volume, and aggregated figures such as average BTC received (spent) by the inputs/outputs and average number of incoming (outgoing) transactions associated with the inputs/outputs. The remaining 72 features are aggregated features, obtained using transaction information one-hop backward/forward from the center node (giving the maximum, minimum, standard deviation, and correlation coefficients of the neighbor transactions for the same information data (number of inputs/outputs, transaction fee, etc.)).

Due to intellectual property issues, there is no exact description of all the features in the dataset, nor it does have the columns' names. This creates a constraint in the analysis of the dataset.

Figure 3 demonstrates the behavior of the graphs by each timestep, colored by the respective label.

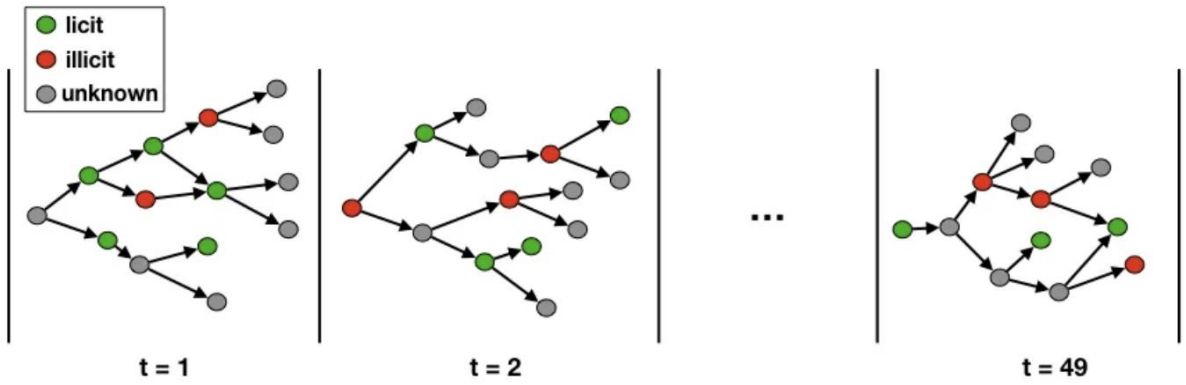

Figure 3: Elliptic Dataset Structure (Lorenz et al., 2020)

4.2.1 Data Description

The main dataset consists of three different datasets, each containing its own set of data, namely features, edges, and classes, as described in Table 1.

Table 1: Elliptic Dataset Description

Dataset	Description	Columns	Type
Classes	The Classes dataset includes the transaction ID and corresponding labels for each transaction. It helps identify and classify transactions based on their specific characteristics or attributes.	Transaction ID	categorical
		Class	categorical
Features	The Features dataset contains the transaction ID along with all the other relevant features. It encompasses a wide range of transaction-related information, allowing for a comprehensive analysis and understanding of each transaction's attributes and properties.	Transaction ID	categorical
		Time Step	integer
		Local Features (1-94)	integer
		Aggregated Features (1-72)	integer
Edges	The Edges dataset captures the connections or links between transactions. It consists of the source and target transactions, providing information about the relationships or associations between different transactions.	Source	categorical
		Target	categorical

4.2.2 Exploratory Data Analysis

Prior to applying ML algorithms, it is essential to undertake a preliminary step aimed at gaining a deeper comprehension of the label targets and the underlying dataset. This crucial process involves thoroughly analyzing the data to uncover meaningful insights and patterns. By undertaking this step, a better understanding of the dataset's characteristics and the specific labels being targeted is achieved. These insights serve as a foundation for subsequent steps, enabling to make informed decisions when applying ML algorithms. In the following chapter, the outcomes of this analysis are presented, focusing on the exploration of label targets.
Table 2 displays the distribution of transactions within the dataset, categorized into different classes.

Table 2: Elliptic Dataset Number of Transactions of Each Class

Labels	Number of Transactions
Licit	42019
Illicit	4545
Unknown	157205

The "Licit" class comprises a total of 42,019 transactions, representing legitimate and non-fraudulent activities. On the other hand, the "Illicit" class encompasses 4,545 transactions, indicating instances of fraudulent or illicit behavior associated. It is worth noting that a significant portion of the dataset falls under the "Unknown" class, with a total of 157,205 transactions. These transactions have not been definitively classified as either licit or illicit. These findings demonstrate that the dataset exhibits a high level of class imbalance.

Figure 4 illustrates the distribution of transactions across different time steps, showcasing the variability in their occurrence. The visualization portrays the fluctuations in the number of transactions over time.

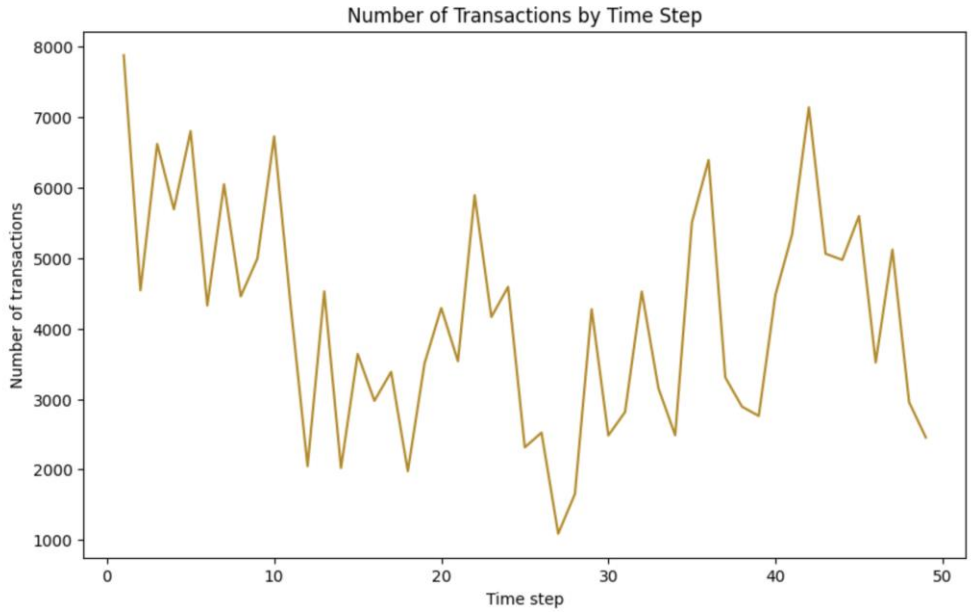

Figure 4: Number of Transactions by Time Step

The significant fluctuations depicted in Figure 4 emphasize the dynamic nature of transaction activity throughout the recorded time period. These variations highlight the ever-changing patterns and trends in transaction occurrences over time.

Figure 5 presents a comprehensive view by combining the information from Table 2 and Figure 4. It visually depicts the distribution of transactions across different classes over time.

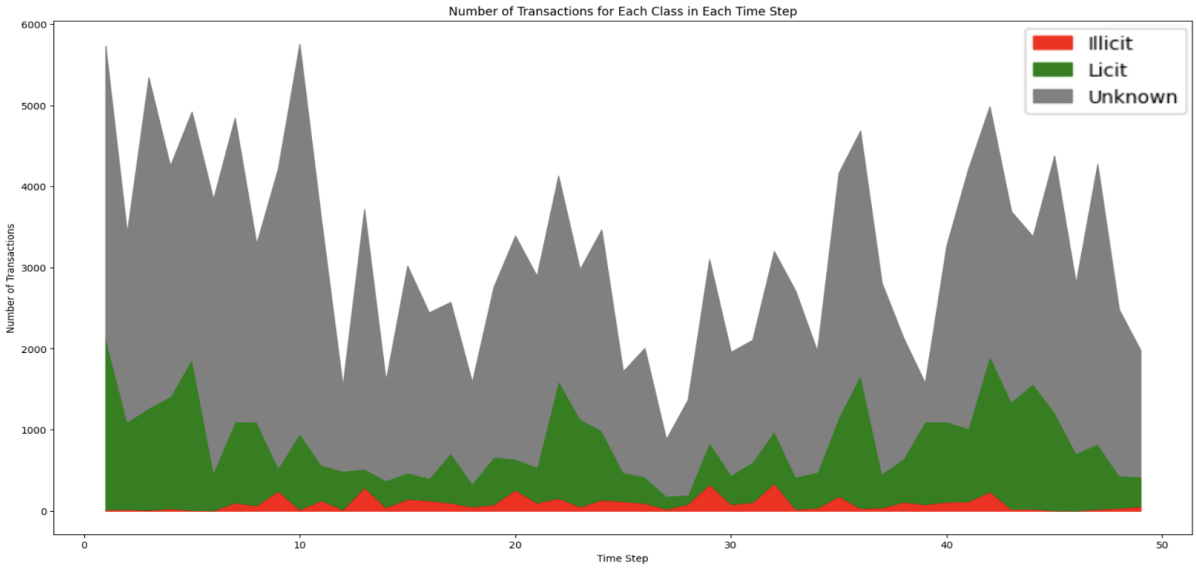

Figure 5: Number of Transactions Categorized by Class and Time Step

The visualization reveals that the overall pattern observed in the dataset is consistent across each time step. Notably, the prevalence of unknown transactions surpasses the number of labeled transactions, while illicit transactions remain a minority within the dataset.

Figure 6, a subset of Figure 5, specifically highlights the comparison between the number of illicit transactions and licit transactions.

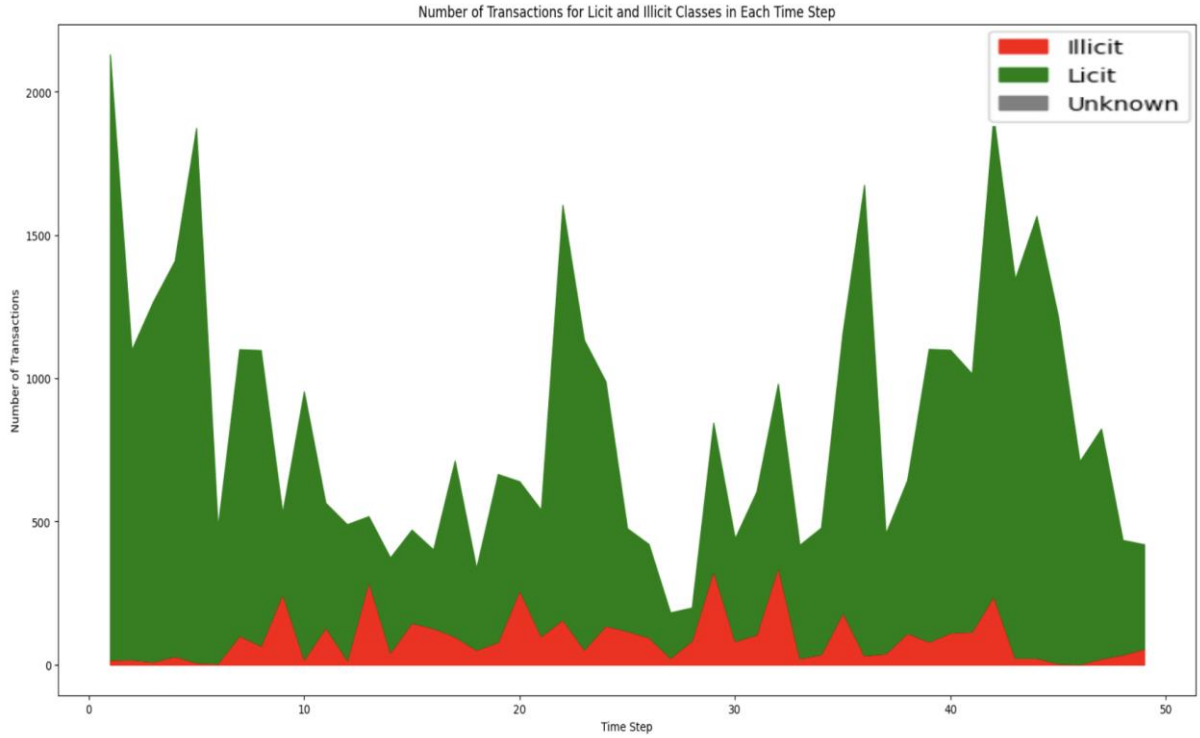

Figure 6: Number of Transactions Categorized by Licit and Illicit Class and Time Step

A notable observation is a significant decrease in the number of illicit transactions starting from time step 42. This decline can be attributed to the closure of a dark market, as mentioned by Weber et al. (2019). The closure of this illicit market had a profound impact on the number of fraudulent transactions, leading to a noticeable reduction in their occurrence.

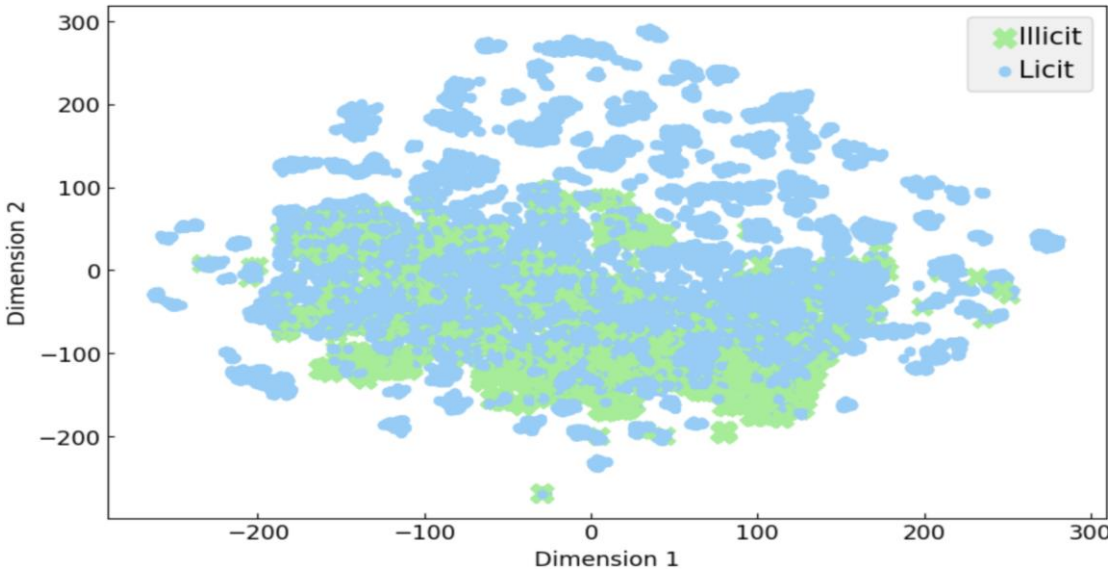

Figure 7: t-SNE Projection of Licit and Illicit Transactions

t-Distributed Stochastic Neighbor Embedding (t-SNE) captures the relationships and similarities between data points in the high-dimensional space and represents them in a lower-dimensional space while preserving these relationships as much as possible. Figure 7 presents a t-SNE visualization that provides insight into the distribution of illicit and licit transactions in a two-dimensional space.

The graph in Figure 7 reveals that the illicit transactions are scattered throughout the plot without a discernible pattern, overlapping with the distribution of licit transactions. This observation reinforces the notion that fraudsters attempt to mimic normal behavior, as their transactions blend in with the overall distribution of legitimate transactions.

5. MACHINE LEARNING ALGORITHMS

To comprehensively understand the algorithms employed in the experiments, a detailed explanation of various approaches is provided in this chapter. Firstly, it is discussed supervised techniques utilized to evaluate the performance of the supervised baseline. These techniques serve as a benchmark against which the efficacy of other methods is compared. Next, the realm of deep learning, specifically focusing on GCNs, is delved into. Furthermore, the utilization of node embeddings and their integration with supervised methods is explored.
Through the exploration of supervised learning algorithms, graph deep learning techniques such as GCNs and Skip-Graph Convolutional Networks (Skip-GCNs), and the incorporation of node embeddings, this chapter aims to provide a comprehensive understanding of the diverse range of algorithms employed in this research. These algorithms form the bedrock of the experimental setup, facilitating a robust evaluation of FD methodologies.

5.1 Algorithms Selection

The primary focus of this work is to explore and compare the performance of traditional ML methods with graph-oriented techniques. Assessing the feasibility of graph-oriented methods is aimed to shed light on their effectiveness in FD.
To accomplish the objective of this dissertation, the research builds upon the previous work conducted by Weber et al. (2019) and Pocher et al. (2022) on the same dataset. These studies have already compared the performance of three supervised algorithms, namely RF, LG, and MLP, with a GCN and Skip-GCN.
It is important to note that RF, LR, and MLP are traditional supervised learning algorithms that are commonly used for handling non-graph data. These algorithms have shown promising results in various ML applications, and they serve as important benchmarks in this dissertation to compare their performance with the specialized graph-based models (GCN and Skip-GCN). Weber et al. (2019) have previously explored the utilization of node embeddings generated by the GCN, demonstrating improved performance when incorporating these embeddings into supervised models. This finding prompts a crucial question: Is the GCN alone the optimal solution, or do the embeddings it produces play a significant role in achieving superior performance?

It is worth noting that in the literature review, it was demonstrated that NRL, specifically node embeddings, has been used in the context of FD and AML.
Hence, this dissertation aims to investigate this question by testing alternative methods of calculating node embeddings. Specifically, the research will compare the performance of GCN embeddings with embeddings generated through other methods, namely Node2Vec and DeepWalk. By evaluating these diverse embeddings, it is sought to gain a comprehensive understanding of the potential of GCN and node embeddings in achieving superior performance in FD.

5.2 Supervised Learning

Supervised learning is a powerful approach where algorithms learn from labeled training data to make predictions or decisions about unseen data. By leveraging existing knowledge and patterns within the data, these algorithms can generalize and infer meaningful insights.
In the context of this dissertation, supervised learning algorithms will be employed, namely RF, LR, and MLP. These algorithms have proven to be effective in various ML tasks.
It is important to note that while these supervised learning algorithms are widely used in the context of AML and have proven to be effective, they do not inherently incorporate graph information. Graph data presents unique challenges due to its relational nature. Therefore, specialized graph-based methods are often required to handle such data effectively. To address this limitation and explore the potential of deep learning in the context of graph data, this dissertation also incorporates GCNs and Skip-GCN. These models are also used as supervised learning approaches.

5.3 Supervised Baseline

In this chapter, the three supervised learning methods utilized as the baseline models in this dissertation will be described.

5.3.1 Random Forest

RF is an ensemble learning method that combines multiple Decision Trees (DT) to perform classification tasks. It has gained popularity due to its robustness, scalability, and effectiveness in handling high-dimensional datasets.

The RF algorithm operates on the principle of ensemble learning, where multiple DTs are combined to make accurate predictions. The DTs are constructed by recursively splitting the dataset based on features, aiming to create nodes that separate instances of different classes. Various splitting criteria, such as Gini impurity or information gain, are utilized to determine the best feature for splitting.

A RF introduces randomness in two aspects. Firstly, it employs bootstrap aggregating (bagging), where each DT is trained on a random subset of the training data, sampled with replacement. This technique enhances diversity among the trees and helps reduce overfitting. Secondly, at each split in a DT, only a random subset of features is considered. This random feature selection further prevents overfitting and promotes the generalization capability of the RF.

During the prediction process, each DT independently makes predictions based on the input features. For classification tasks, the RF combines the predictions of all the trees using majority voting or average.

RF offers several advantages over individual DTs. They provide improved accuracy due to the ensemble of trees, where the collective decisions compensate for individual errors. RF is robust against overfitting and resistant to outliers, resulting in more reliable predictions.

Figure 8 visually depicts the operational workflow of RF, providing a visual representation that reinforces the concepts described earlier.

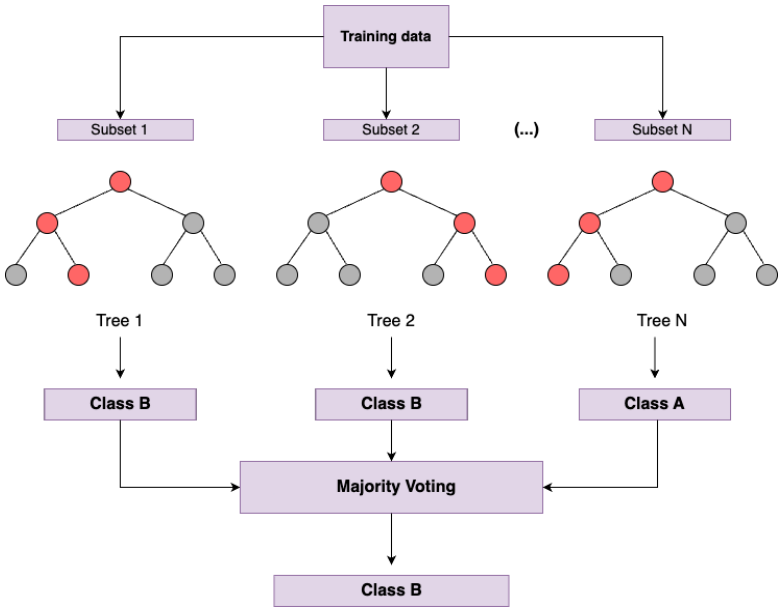

Figure 8: Simplified Random Forest Model

5.3.2 Logistic Regression

LR is a popular ML algorithm widely used for binary classification tasks. LR models the relationship between a set of input features and the probability of an instance belonging to a particular class.

At its core, LR assumes that the relationship between the features and the class probabilities can be approximated using a logistic function. This function combines a linear model with a non-linear sigmoid function. The linear model computes a weighted sum of the input features, and the sigmoid function maps the output to a probability between 0 and 1. The sigmoid function, also known as the logistic function, is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$, where z represents the weighted sum of the input features. Figure 9 illustrates the Sigmoid function.

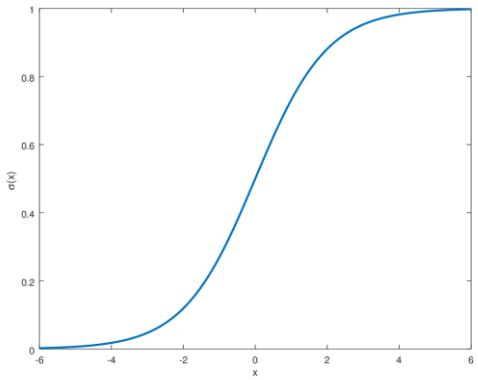

Figure 9: Sigmoid Function

To train a LR model, the parameters of the logistic function are estimated using techniques such as maximum likelihood estimation or optimization algorithms like gradient descent. The training process iteratively adjusts the parameters to maximize the likelihood of the observed data. Once the model is trained, it can make predictions by computing the probability of an instance belonging to a class based on the learned parameters and assigning the instance to the class with the highest probability.

One of the advantages of LR is its interpretability. The model's parameters provide insights into the impact of individual features on the predicted probabilities. By examining the signs and magnitudes of the coefficients, one can determine whether a feature positively or negatively affects the likelihood of a certain class.

5.3.3 Multilayer Perceptron

MLP is a widely used ANN for binary classification tasks. It consists of an input layer, one or more hidden layers, and an output layer. Each layer contains interconnected artificial neurons that process information.

Neurons in an MLP perform a weighted sum of their inputs and apply an activation function to produce an output. Common activation functions include the sigmoid (logistic) function (like the LR) and the ReLU function defined as $\mathrm{ReLU}(x) = \max(0, x)$. The ReLU function is illustrated in Figure 10. The choice of activation function depends on the problem and desired properties of the network.

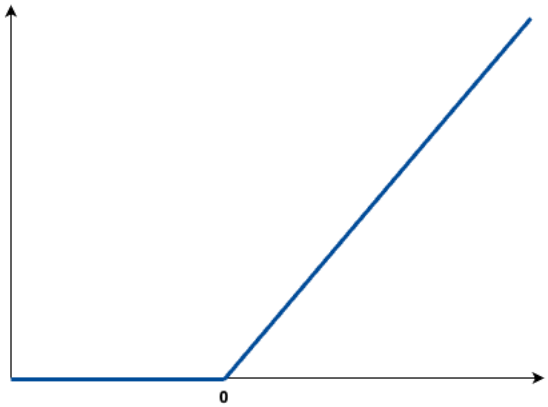

Figure 10: ReLU Function

The network follows a feedforward approach, where information flows from the input layer through the hidden layers to the output layer. This process is known as forward propagation.

Each layer's output serves as input for the subsequent layer, ultimately leading to the output layer's classification prediction.

MLPs are trained using the backpropagation algorithm, which adjusts the weights in the network to minimize prediction errors. The algorithm calculates the error between the predicted outputs and the expected outputs. It then propagates the error backwards through the network, updating the weights using gradient descent. This iterative process fine-tunes the network to improve its classification performance.

Hyperparameters, such as the number of hidden layers, the number of neurons per layer, learning rate, and batch size, influence an MLP's performance.

Its advantages include the ability to capture complex patterns, flexibility in network architecture, strong generalization capabilities, and effective handling of high-dimensional data.

Figure 11 visually depicts the operational workflow of MLP, providing a visual representation that reinforces the concepts described earlier.

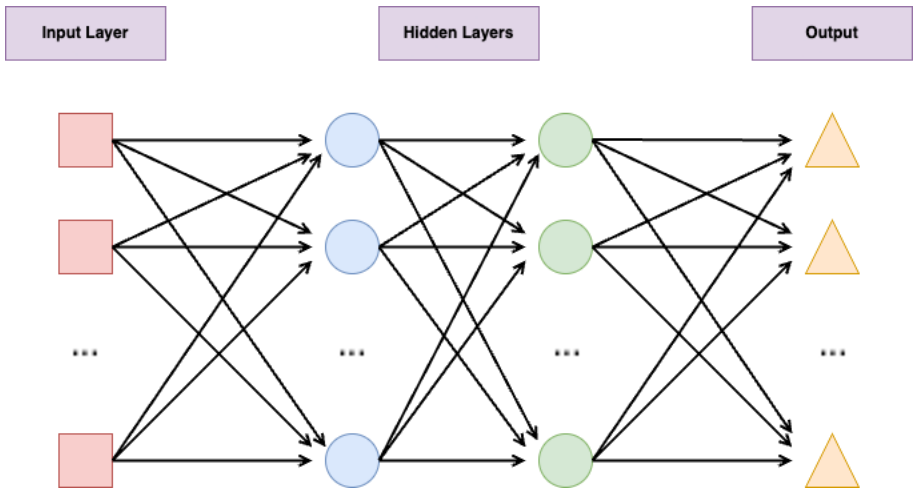

Figure 11: Simplified Multilayer Perceptron Model

5.4 Deep Learning

Deep learning, a subfield of ML, has emerged as a powerful approach to solving complex tasks. It offers significant advantages by automatically learning hierarchical representations from raw data. Unlike traditional methods that rely on manual feature engineering, deep learning

models extract increasingly complex features through multiple layers of interconnected neurons, allowing them to capture high-level representations and abstract concepts.

Deep learning models are built upon neural network architectures, consisting of input, hidden, and output layers. These layers are composed of interconnected neurons that process input data through weighted computations and nonlinear activation functions. The depth of these networks enables them to learn intricate patterns and relationships in the data. Training deep neural networks involves backpropagation, which utilizes optimization algorithms to iteratively adjust the model's weights and biases based on computed error or loss. This process allows the model to learn optimal parameters by propagating error backwards through the network, minimizing the discrepancy between predicted and ground truth values.

The equation to propagate the feature representations from one layer to the next layer in a neural network during the forward pass can be expressed as follows:

$$\mathrm{H}^{i+1} = \sigma(\mathrm{W}^{i} * \mathrm{H}^{i} + \mathrm{b}^{i})$$

This equation represents the feed-forward layer and can be explained as:

- H^{i+1} is the feature representation at layer i+1.
- σ is the activation function.
- W^{i} is the weights at layer i.
- H^{i} is the feature representation at layer i.
- b^{i} is the bias at layer i.

One deep learning model utilized in this dissertation is the MLP described in the Supervised Baseline section.

Despite the efficacy of traditional ML techniques within the field of AML, the complexity of graph data has presented several challenges for these algorithms. To address this, GNNs have been developed as a specialized class of deep learning methods, specifically designed for the inference of graph-structured data. GNNs offer a direct application to graph data and provide a convenient means for performing node-level, edge-level, and graph-level predictions.

In addition to GNNs, GCNs have emerged as a popular and effective class of deep learning models for graph-structured data, which will be explained and detail in the following section.

5.5 Graph Convolutional Networks

Traditional supervised models, like RF, LR, and MLP, face challenges when applied to graph data due to the unique characteristics of graphs. Graphs are complex structures with interconnected nodes and edges, where the relationships between nodes play a crucial role. Traditional models often struggle to capture these relationships and exploit the graph structure effectively. They are designed for tabular or sequential data and do not inherently incorporate the topology of graphs. Therefore, there is a need for specialized models that can handle graph-specific patterns and dependencies.

GCNs are a specialized type of deep learning model specifically designed for graph-structured data. GCNs directly operate on the graph structure, leveraging the connections and features of nodes to process the data effectively. By incorporating information from neighboring nodes and individual node features, GCNs can capture complex relationships and learn meaningful node representations.

The fundamental inputs for GCNs are the graph structure and node features. The graph structure, typically represented as an adjacency matrix or an edge list, captures the connections between nodes, while the node features represent the characteristics associated with each node. GCNs consist of multiple layers, each with a unique function, enabling hierarchical processing of the graph data.

The input layer serves as the starting point for GCNs, taking the graph structure and node features as input and defining the initial node representations. The subsequent GCN layers, also known as message-passing layers, play a pivotal role in GCNs. They propagate information between nodes by aggregating features from neighboring nodes and updating node representations accordingly. These layers effectively capture both local and global dependencies within the graph.

In each graph convolutional layer, the node embeddings from the previous layer are combined with the information from their neighboring nodes, typically through a weighted sum or concatenation operation. This aggregation process allows each node to incorporate information from its immediate neighbors, capturing local patterns.

Additionally, by stacking multiple graph convolutional layers, information can propagate across the graph, capturing global dependencies and higher-order relationships. As the node embeddings pass through these layers, they are gradually refined, capturing increasingly complex patterns and relationships within the graph data.

The number of convolutional layers in a GCN is a crucial parameter that influences the model's capacity to capture intricate relationships within graph data. Deeper architectures with more layers can potentially capture more complex patterns and dependencies, allowing for a more detailed representation of the graph. However, increasing the number of layers also introduces the risk of overfitting, particularly when the training data is limited. On the other hand, shallower architectures may have a limited ability to capture fine-grained relationships but can be computationally more efficient. Hence, the choice of the number of layers in a GCN should strike a balance between model complexity and generalization capability.

To introduce non-linearities and enhance the model's capacity to capture complex patterns and relationships, a non-linearity layer is commonly included in GCNs. This layer incorporates non-linear activation functions, such as ReLU (illustrated in Figure 10) into the model.

Finally, the output layer transforms the refined node representations into the desired output format for the specific task, such as node classification or link prediction. In the case of classification tasks, a softmax activation function is often applied to the output layer. The softmax function converts the raw scores or logits from the previous layer into probabilities, representing the likelihood of each class. These probabilities sum up to 1, and each value indicates the probability of the corresponding class.

Figure 12 demonstrates the flow of information in a GCN, starting from the input graph and passing through the hidden layers and activation functions. The output layer produces the logits, which are then transformed into predicted class probabilities using the softmax function.

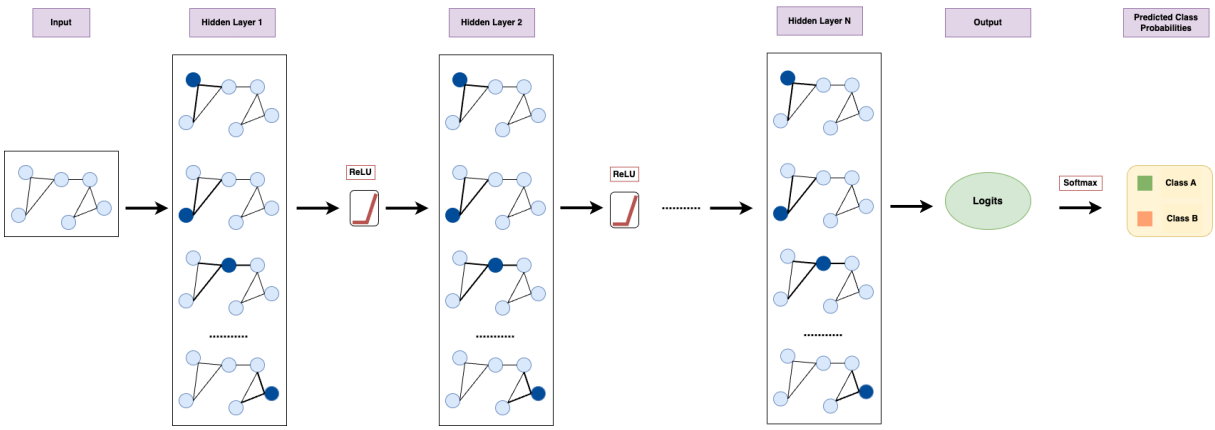

Figure 12: Simplified Graph Convolutional Network Model

By connecting the input layer, graph convolutional layers (hidden layers), output layer, and non-linearity layer (activation function), GCNs gradually refine the node representations,

leveraging the graph structure and node features to capture intricate relationships and patterns within the graph data.

The convolutional layer is similar to the feed-forward layer and is defined as:

$$\mathrm{H}^{i+1} = \sigma(W^{i} * \mathrm{H}^{i} * \mathrm{A})$$

The equation to the feed-forward layer can be explained as:

- H^{i+1} is the feature representation at layer i+1.
- σ is the activation function.
- W^{i} is the weights at layer i.
- H^{i} is the feature representation at layer i.
- A is a matrix that represents the edges or connections between the nodes.

The main distinction between the feed-forward layer of a neural network and the convolutional layer of the GCN, as captured by their respective equations, lies in the incorporation of the graph structure. While a traditional neural network uses a feed-forward layer with weights and biases, a GCN adds a graph adjacency matrix (A) to its convolutional layer, allowing it to leverage the relationships between nodes in a graph for more effective information propagation and feature extraction.

The equation to represent the complete process of the GCN with softmax is as follows:

$$Z = softmax(A * \sigma(\ldots(\mathrm{A} * \sigma(\mathrm{A} * \mathrm{X} * \mathrm{W}^{0}) * W^{1}) * \mathrm{W}^{2}) * \ldots * \mathrm{W}^{L})$$

In this equation:

- X represents the input feature matrix.
- W^{0}, W^{1}, W^{2}, …, W^{L} are the weight matrices at each layer.
- A* X * W^{0} represents the propagation of the input features through the first layer.
- The output of each layer is passed through the activation function σ.
- The result of one layer, $\sigma(\mathrm{A} * \mathrm{X} * \mathrm{W}^{0})$, is then multiplied by A to incorporate the graph structure information.
- This process is repeated for each layer, with the output of the previous layer being used as the input to the next layer.
- Finally, the resulting feature representation from the last layer is passed through the softmax function to obtain the output probabilities for each node in the graph.

Training a GCN is essential to learn meaningful representations of the graph data. During the training process, the GCN adjusts its parameters to minimize the discrepancy between the predicted outputs and the ground truth labels. This optimization is achieved by iteratively updating the model's weights based on the difference between the predicted and actual outputs. To facilitate this optimization, an optimizer, and a loss function are employed.

In the context of this dissertation, the Adam optimizer was utilized. Adam is a popular optimization algorithm that combines the benefits of both adaptive learning rate methods and momentum-based methods. It dynamically adjusts the learning rate for each parameter, allowing for efficient convergence during training. The Adam optimizer calculates adaptive learning rates for each parameter by considering the first and second moments of the gradients.

Furthermore, the cross-entropy loss function was employed to train the GCN for FD. Cross-entropy loss is commonly used in classification tasks and measures the dissimilarity between the predicted probability distribution and the true distribution of the target labels. By minimizing the cross-entropy loss, the GCN learns to make accurate predictions and effectively classify nodes as either fraud or non-fraud. This loss can be given by the following equation:

$$LCE = -\sum_{i=1}^{n} t_i log(\mathrm{p}_i) \text{ , for n classes,}$$

Where t_i is the true label and p_i is the Softmax probability for ^{ith} class.

5.6 Skip-Graph Convolutional Networks

Skip-GCNs are a variant of the traditional GCNs that incorporate skip connections, inspired by skip connections commonly used in deep neural networks. The purpose of skip connections in Skip-GCNs is to facilitate the flow of information from earlier layers to later layers, enabling the model to access both local and global information simultaneously.

By incorporating skip connections, Skip-GCNs aim to address some limitations of traditional GCNs and enhance their performance in capturing complex patterns and dependencies within graph-structured data. While Normal GCNs rely solely on the propagation of information through the graph convolutional layers, Skip-GCNs introduce additional connections that directly connect earlier layers to later layers. These skip connections provide shortcuts for information flow and help alleviate the vanishing gradient problem, which can hinder the learning process in deep neural networks.

In a Skip-GCN, the skip connections are typically implemented by concatenating or adding the node features from earlier layers to the outputs of later layers. This combination of features from different layers allows the model to utilize both local and global information effectively. By incorporating information from multiple layers, Skip-GCNs can capture fine-grained details and long-range dependencies in the graph data. This is particularly advantageous when dealing with large-scale graphs or graphs with complex connectivity patterns.

By enabling the direct flow of information, Skip-GCNs can capture more complex relationships and patterns within the graph data. Additionally, the skip connections enhance the model's ability to model long-range dependencies and leverage both local and global information effectively.

Overall, Skip-GCNs offer an extension to the traditional GCN architecture, providing a more powerful and expressive framework for graph-based learning tasks.

Figure 13 demonstrates the application of skip connections between layers and the concatenation of the features to connect the output of one layer to the input of subsequent layers.

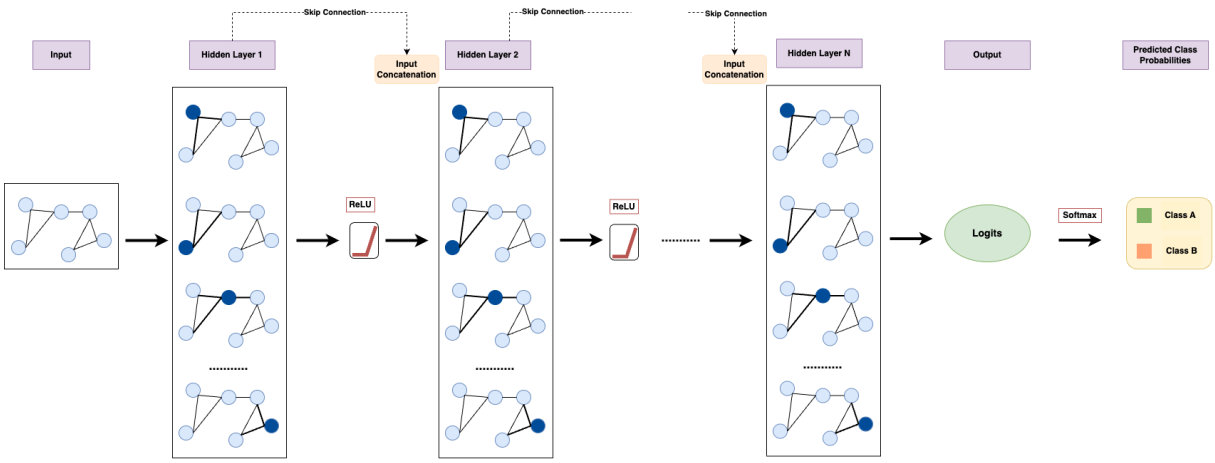

Figure 13: Simplified Skip-Graph Convolutional Network Model

The equation to represent the complete process of the Skip-GCN with softmax is as follows:

$$Z = softmax(A * \sigma(\dots \sigma(\mathrm{A}\,\sigma(\mathrm{A} * \mathrm{X} * \mathrm{W}^0) * \mathrm{W}^1 + X * \mathrm{W}^s) * W^2 + X * W^s) * W^3 + \cdots + X * W^s)$$

In this equation:

- X represents the input feature matrix
- $\mathrm{W}^0$and W^1 are the weight matrices at the two layers of the Skip-GCN.
- W^s is the weight matrix used for the skip connection.

- A * X * W^0 represents the propagation of the input features through the first layer.
- The output of each layer is passed through the activation function σ.
- The output of the first layer, $\sigma(A * X * W^0)$, is then multiplied by A to incorporate the graph structure information.
- Additionally, the features from the input layer, X, are multiplied by the weight matrix W^s and added to the output of the later layer ($\sigma(A * X * W^0 * W^1) * W^1$) using skip connections.
- This combination of skip connections allows the model to access both local and global information simultaneously.
- Finally, the resulting feature representation from the last layer is passed through the softmax function to obtain the output probabilities for each node in the graph.

5.7 Representation Learning

5.7.1 Node Embeddings

Node embeddings play a vital role in representing and understanding graph-structured data. The raw representation of nodes in a graph may lack meaningful patterns or structural information, making it challenging to extract valuable insights from the data.

Node embeddings address this challenge by transforming the nodes into dense, low-dimensional vectors that capture the inherent relationships and structural properties of the graph. These embeddings encode the essential characteristics of the nodes, enabling downstream ML models to operate efficiently on graph data.

Node embeddings aim to capture semantic similarities and contextual relationships between nodes. By mapping nodes to a continuous vector space, node embeddings enable the application of powerful mathematical operations and similarity measures, facilitating various downstream tasks such as node classification, link prediction, and graph clustering.

It is worth noting that node embeddings should be used in conjunction with the input node features rather than as standalone representations. While the embeddings capture structural information and underlying relationships within the graph, the input node features provide attribute-level details about the nodes, such as textual descriptions or numerical attributes associated with each node. By combining both embeddings and features, a more

comprehensive representation of the graph data can be achieved, leveraging the strengths of each modality.

The integration of node embeddings and input node features allows for a more holistic understanding of the graph dataset. The embeddings capture the global structure and topology of the graph, while the features offer fine-grained attribute-level information. This combination provides complementary insights that enhance the overall performance and interpretability of graph-based ML models.

Figure 14 illustrates the representation of node embeddings in a two-dimensional space.

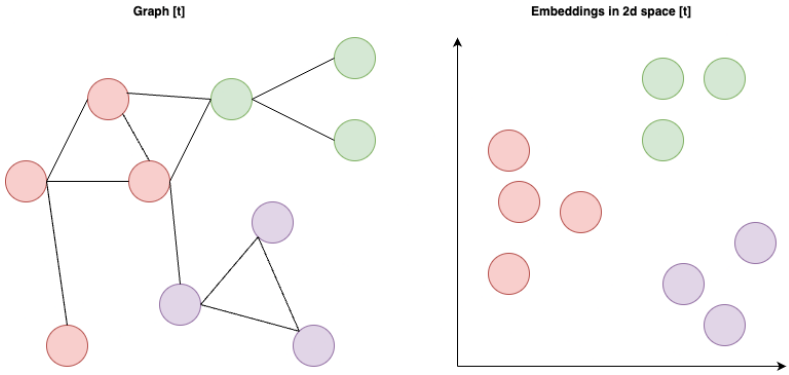

Figure 14: Simplified Node Embeddings Representation

5.7.2 DeepWalk

DeepWalk is a popular NRL algorithm specifically designed for graphs. It is inspired by the concept of word embeddings and employs a similar approach to learning node embeddings. DeepWalk uses random walks on the graph to generate sequences of nodes, treating them as sentences. It then applies techniques like Skip-gram to learn embeddings for the nodes.

Random walk is a process where a walker moves from one node to another by following randomly chosen edges. Starting from a given node, the walker traverses the graph by randomly selecting an edge connected to the current node and moving to the adjacent node. This process is repeated iteratively, with each step being independent of the previous choices. The random walk continues until a predetermined number of steps, or a specific stopping condition is met.

By utilizing random walks, DeepWalk captures the local neighborhood information around each node and learns distributed representations that encode the structural properties of the graph. These representations can capture similarities between nodes based on their network proximity.

5.7.3 Node2vec

Node2Vec is another popular NRL algorithm that extends upon DeepWalk. It addresses the limitation of DeepWalk, where the random walks only consider the breadth-first or depth-first sampling strategy. Node2Vec introduces a flexible biased random walk strategy that enables the exploration of different types of neighborhoods within the graph.

The flexibility of Node2Vec lies in its ability to balance between breadth-first sampling (capturing the global graph structure) and depth-first sampling (capturing local neighborhood information). By defining two parameters, the return, and in-out parameters, Node2Vec can adjust the exploration strategy during random walks, allowing the algorithm to capture a wider range of structural properties within the graph.

The Node2Vec algorithm starts by generating random walks on the graph, similar to DeepWalk. However, it introduces two parameters: p and q. The parameter p controls the likelihood of revisiting nodes within the same neighborhood, allowing the algorithm to capture more local information. On the other hand, the parameter q controls the likelihood of exploring nodes further away, enabling the algorithm to capture more global information.

By adjusting the values of p and q, Node2Vec can balance between exploring diverse neighborhoods and focusing on specific areas of the graph. This flexibility allows for a more nuanced representation of the graph's structural characteristics.

5.7.4 Graph Convolutional Network Embeddings

In addition to their ability to capture the structural information of graph data, GCNs also generate node embeddings that are highly suitable for supervised learning tasks. By leveraging convolutional operations, GCNs propagate information from neighboring nodes and incorporate it into the representation of each node. This process enables the GCN to learn expressive and informative embeddings that encapsulate both the local and global characteristics of the graph. These node embeddings can then be utilized as input features for downstream supervised models, such as classification or regression algorithms. By leveraging the learned embeddings, supervised models can effectively leverage the rich structural information of the graph and make accurate predictions or classifications based on the labeled data. The integration of GCNs and supervised models allows for the seamless combination of

NRL and traditional ML techniques, leading to improved performance in various graph analysis tasks.

5.8 Evaluation Metrics

Evaluation metrics serve as a critical component to improve comprehension and streamline the assessment of prediction quality among the diverse ML methods. This chapter serves as a comprehensive introduction to the metrics chosen for this purpose, aiming to establish a standardized framework for evaluating and quantifying the efficacy of the predictions generated.

5.8.1 Precision

Precision is a fundamental evaluation metric used to measure the accuracy and reliability of a classification model. It quantifies the proportion of correctly predicted positive instances out of all instances predicted as positive. In other words, Precision focuses on the correctness of positive predictions made by the model. It can be calculated using the following formula:

$$Precision = \frac{TP}{TP + FP}$$

Where:

- TP represents the number of correctly predicted positive instances.
- FP represents the number of instances incorrectly predicted as positive.

By examining Precision, insights can be gained into the model's ability to minimize FP and accurately identify positive instances. A higher Precision score indicates a lower rate of FP, reflecting a higher level of Precision in the model's predictions.

5.8.2 Recall

Recall, also known as sensitivity or True Positive Rate (TPR), is an essential evaluation metric in classification tasks. It measures the ability of a model to correctly identify all positive instances out of the total actual positive instances. Recall focuses on capturing the completeness of positive predictions made by the model. Recall can be calculated using the following formula:

$$Recall = \frac{TP}{TP + FN}$$

Where FN represents the number of instances incorrectly predicted as negative.

By examining Recall, insights can be gained into the model's ability to minimize FN and accurately capture positive instances. A higher Recall indicates that the model is proficient at identifying positive instances, ensuring fewer instances are missed or falsely classified as negative.

5.8.3 F1-Score

The F1-Score is a widely used evaluation metric that combines Precision and Recall into a single measure, providing a balanced assessment of a model's performance. It takes into account both the ability to minimize FP (Precision) and the ability to minimize FN (Recall). The F1-Score can be calculated using the following formula:

$$F1\_Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1-Score considers both Precision and Recall equally, giving equal weight to both measures. It is particularly useful in situations where it is needed to strike a balance between Precision and Recall. A higher F1-Score indicates a better trade-off between Precision and Recall, signifying a model's overall effectiveness in making accurate positive predictions while minimizing FP and FN.

The F1-Score is a valuable evaluation metric for assessing the overall performance of classification models and is often used as a benchmark when comparing different algorithms.

6. EXPERIMENTS

This section aims to offer a thorough understanding of the proposed solutions and the experimental setup. It provides a comprehensive overview of the experiments conducted. The following is a detailed description of the experiments, presenting the methodologies employed, thereby ensuring clarity in understanding.

6.1 Data Preparation

In this chapter, delving into the vital step of data preparation, the focus is on discussing the important modifications made to ensure the dataset's appropriateness for the problem at hand and its compatibility with the chosen ML models.

6.1.1 Data Preprocessing

This subchapter focuses on the initial steps taken to transform and filter the dataset for compatibility with ML models.
It included the following steps:

1. **Converting categorical labels to integers:** To facilitate the analysis, it was performed a conversion of the categorical labels into integers. Initially, the labels for the transactions were represented as "1", "2", and "unknown". To ensure compatibility with the ML models, those were mapped to the following integers: "1" was assigned the value 1, "2" was assigned the value 0, and "unknown" was assigned the value -1. This transformation allows for easier numerical computations and enables seamless integration with the ML algorithms and models employed during the analysis process.
2. **Filtering the dataset:** To ensure the analysis aligns with the supervised learning approach, the dataset was filtered to include only labeled transactions. This step was crucial as labeled transactions provide the necessary ground truth information for training and evaluating the supervised learning models.
3. **Renaming feature columns:** To enhance clarity and improve understanding, the columns in the features dataset were assigned representative names. This step was undertaken as the original columns lacked identifying names. Representative names

such as *"Local_Feature_1"* and *"Aggregated_Feature_1"* were chosen to provide meaningful descriptions of the features. The purpose of this modification is to facilitate easier interpretation of the dataset and promote a more intuitive understanding of the underlying information they represent. By employing descriptive names for the features, the dataset becomes more comprehensible, and the significance of each feature is effectively conveyed, contributing to a more interpretable data analysis process.

4. **Merging label and feature datasets:** In order to consolidate the information for each transaction, the classes and features datasets were merged. Initially, the label and feature information were stored in separate datasets, which presented a challenge for analysis. By merging these datasets, the relevant label and feature information for each transaction were combined. This integration ensures that each transaction is associated with its corresponding label and feature data, creating a unified dataset that can be effectively utilized for the subsequent ML models.
5. **Normalize Data:** In order to ensure compatibility with the MLP model, the data underwent a normalization process using the Min-Max scaling technique. Normalization is necessary for neural networks because it helps to standardize the input data and bring all features onto a similar scale. By applying Min-Max scaling, the features were rescaled to a specific range, typically between 0 and 1, preserving the relative relationships between the data points. This normalization step facilitates faster and more stable convergence during training, improves the model's generalization capability, and enhances its ability to capture meaningful relationships and patterns within the data.
6. **Correct indexes:** To construct the graphs, it is essential to ensure that the indexes are correctly assigned, as the edges and nodes rely on accurate index values to establish precise connections. To maintain the integrity of the graph representation, the indexes for both nodes and edges were assigned and validated carefully. This involved thorough verification to guarantee that each node possessed a unique and consistent index value, and that the edges between nodes properly referenced the correct indexes. By adhering to this procedure, the graph can accurately capture and represent the relationships between nodes, enabling effective analysis and interpretation of the data.

After the preprocessing, the cleaned dataset encompassed 46,564 transactions and 36,624 edges, so the dimensions of the graph can be described as follows.

- V = 46,564 transactions (nodes)
- E = 36,624 edges

6.1.2 Preparing for Node2Vec and DeepWalk

This subchapter focuses on preparing the inputs for Node2Vec and DeepWalk algorithms, which generate node embeddings. These algorithms require a graph representation of the dataset, which is constructed using the NetworkX library.
The steps involved in this process are as follows:

1. **Constructing graphs for each time step:** To capture the temporal aspect of the dataset, graphs were constructed for each time step. This step allows for the incorporation of temporal information into the graph representation.
2. **Representing transactions as nodes and establishing edges between nodes:** In the graph, each transaction serves as a node, preserving its individual attributes and properties. Edges were then created between nodes to reflect the interactions or associations between the corresponding transactions. These edges captured the relationships and dependencies within the dataset, enabling Node2Vec and DeepWalk algorithms to learn and generate node embeddings based on these connections.

By constructing graphs for each time step and representing transactions as nodes with interconnected edges, the resulting graph data structure provides a suitable input for Node2Vec and DeepWalk algorithms.

6.1.3 Preparing for Graph Convolutional Networks

The preparation for GCNs is covered in this subchapter, focusing on the steps required to format the input data for compatibility with GCN models.
It involved:

1. **Node Features:** The node features, representing the attributes or properties associated with each node, were organized. These features provide essential

information for the GCN to learn and make predictions based on the characteristics of each transaction.

2. **Adjacency Matrix (edge_index):** The adjacency matrix, also referred to as *edge_index* in this context, was constructed. This matrix captures the connectivity between nodes in the graph, indicating which nodes are connected to each other through edges. It serves as a crucial component for understanding the relationships and dependencies within the graph.
3. **Graph Structure:** The graph structure was defined by establishing the connections between transactions. Edges were created between nodes to represent the interactions or associations between the corresponding transactions. This step formed the foundation for the GCN to propagate information and learn from the interconnected nodes.
4. **Input Data Formatting:** The input data was formatted to ensure compatibility with the GCN model. This involved preparing the node features and adjacency matrix in a suitable format that could be effectively fed into the GCN architecture. Proper formatting of the input data enables the GCN to process and analyze the information to make predictions or perform other tasks.
5. **Normalize Data:** In order to ensure compatibility with the GCN model, the data underwent a normalization process using the Standard Scaler technique. Normalization is crucial for neural networks as it helps standardize the input data and bring all features onto a similar scale. By applying Standard Scaler, the features were rescaled to have a mean of 0 and a standard deviation of 1, thus transforming the data to a standard normal distribution. This normalization step facilitates faster and more stable convergence during training, improves the model's generalization capability, and enhances its ability to capture meaningful relationships and patterns within the data.

6.2 Experimental Setup

This chapter provides an overview of the experiments conducted in this dissertation, specifically highlighting the methods employed to establish the supervised baseline, as well as

the employment of GCN, Node2Vec, and DeepWalk techniques. The forthcoming sections will present a detailed account of the experimental setup, outlining each step taken in the process. To ensure consistency with the methodology used by Weber et al. (2019) on the identical dataset, a similar approach is adopted for partitioning the data into sequential train and test datasets in all the experiments. The training set consists of 29,894 labeled transactions, encompassing all samples up to the 34th time step. On the other hand, the test set comprises 16,670 labeled transactions, starting from the 35th time step and beyond.

To maintain reproducibility and consistency during the experiments, the code's execution began by setting a random seed value (e.g., 8347658). This seed value was used to initialize the random number generator, dictating the sequence of random operations performed throughout the experiments. This approach enabled the experiments to be replicated reliably, ensuring consistent evaluation and comparison of the models. To introduce further randomness and variability, a unique random state was generated for each one of the models. The random state was generated using a random integer function. This random state value was then passed as a parameter when initializing the models.

For the purpose of evaluating the performance of each model in this dissertation, the Precision, Recall, and F1-Score metrics were employed using the sklearn library. However, instead of relying on the default threshold defined by sklearn to extract these metrics, the threshold at which the F1-Score was maximum was determined. Precision and Recall were then extracted specifically for that threshold. This approach aimed to obtain the optimal balance between Precision and Recall for each model. By using this approach, a more comprehensive assessment of the models' performance was achieved, considering the specific threshold that maximized the F1-Score.

Employing this customized threshold selection strategy provided a more refined evaluation of the models' capabilities in identifying and classifying the relevant transactions. By optimizing the F1-Score, a higher level of accuracy and effectiveness in detecting fraudulent activities within the dataset was aimed.

It is important to note that Weber et al. (2019) did not provide a specific structure for either the preprocessing or the architecture. Therefore, this study had complete freedom to design the GCN and choose the appropriate preprocessing steps that were deemed suitable. Hence, the variations in my results compared to theirs can be attributed to these differences.

6.2.1 Supervised Baseline

In the supervised baseline experiments, two scenarios were created to evaluate the performance of the models. In the first scenario, the supervised models were trained using only the local features. Subsequently, in the second scenario, both the local and aggregated features (all features) were utilized. The models employed for these experiments were RF, LR, and MLP.

The purpose of conducting the supervised baseline experiments was to establish a benchmark against which the performance of graph-oriented models could be compared. By evaluating the models' performance using different sets of features, including both local and aggregated information, a comprehensive understanding of their capabilities could be obtained. This approach provides valuable insights into the potential improvements that graph-based techniques may offer over traditional supervised models.

To maintain consistency with the work of Weber et al. (2019), sklearn was utilized to implement the tree models in the supervised baseline experiments. The LR model was trained using the default parameters provided by sklearn. For RF, specific parameter settings were applied, including n_estimators=50 and max_features=50. In the case of MLP the following parameter values were used: hidden_layer_sizes=(50,), solver='adam', max_iter=500, and learning_rate_init=0.001.

A comprehensive summary of the parameters used for each model can be found in Table 11 from Appendix I.

6.2.2 Graph Convolutional Networks

In this dissertation, the utilization of GCNs for detecting patterns of fraudulent behavior within the cryptocurrency market is explored. GCNs offer a promising approach due to their ability to analyze and extract meaningful information from graph-structured data, thereby capturing the inherent relationships and dependencies present in transaction networks.

This section provides a comprehensive account of the experimental setup for the GCN models employed in this study. Two variants of GCNs were implemented: the Normal GCN and the Skip-GCN. The Normal GCN follows the traditional architecture of a GCN, allowing the model to learn and propagate information through the graph structure. On the other hand, the Skip-GCN introduces skip connections, which enhance the model's performance by enabling

information to flow through multiple layers of the network. By incorporating skip connections, the Skip-GCN can effectively capture both local and global patterns, thereby improving its ability to detect fraudulent behavior.

Additionally, the GCNs were explored through two implementations, following the scenarios implemented in the supervised baseline. Firstly, both the GCN and Skip-GCN models were trained solely using the Local Features as input. Secondly, both models were trained using All Features, encompassing a more comprehensive range of input data. These scenarios were considered to evaluate and compare the performance of the models under different feature combinations, enabling a deeper understanding of their detection capabilities.

To provide a detailed account of the experimental setup, this section includes information about the hyperparameters and model configuration. The hyperparameters used in the experiments are consistent with the work of Weber et al. (2019).

Table 12 from Appendix II presents the parameters used for both the GCN and Skip-GCN models.

Despite implementing the Normal GCN and Skip-GCN variants, the initial experimental results were unsatisfactory. Therefore, a second approach to the experiment was conducted, requiring a complete overhaul of the approach in terms of architecture and inputs. This decision was made to address the limitations and challenges encountered in the initial approach and to strive for improved results.

Consequently, this section also provides a detailed account of the experimental setup for the revised approach, which focuses on a new model architecture. The specific details and configuration of the new model architecture will be explained, highlighting the changes made to address the shortcomings of the initial approach.

Overall, the first approach implemented the GCN models using the PyTorch library, with custom classes defining the individual layers and the complete model architectures. These classes provided the necessary structure and functionality to train and propagate information through the GCN models. These classes leverage PyTorch's neural network module, nn.Module, to construct the GCN-based architecture. This approach code is illustrated in Figure 33 from Appendix IV.

The "GCN" class represents a graph convolutional layer. It initializes attributes such as input and output features, weight, and bias. The forward method performs graph convolution operations for matrix multiplications and returns the output.

The "NormalGCNModel" class, represents the GCN model. It consists of two "GCN" layers. The forward method applies ReLU activation after the first GCN layer and returns the final output. The "SkipGCNModel" class represents the Skip-GCN model. It adds skip connections by concatenating input features with the output of the first GCN layer. The concatenated features are passed to the second GCN layer.

In this first approach, the architecture of the GCN models relied on manually constructing the adjacency matrix. This matrix, with the shape of (N, N), captured the connections and relationships between transactions in the cryptocurrency network. The N represents the number of transactions or nodes in the network. The adjacency matrix was built by identifying the corresponding nodes and edges, mapping the transaction IDs to node indices, and incrementing the matrix entries to represent the links between transactions.

In addition to the adjacency matrix, the inputs for the GCN models included a feature matrix. The feature matrix, with a shape of (N, F), encapsulated the various attributes of the nodes. Here, N represents the number of transactions or nodes, and F represents the number of features associated with each transaction.

However, despite these efforts, the initial results obtained from the first approach did not meet the desired expectations. The model's performance in detecting fraudulent activities in the cryptocurrency market fell short of the desired F1-Score. As a result, it became evident that modifications were necessary to improve the model's effectiveness in capturing the underlying patterns of fraudulent behavior.

In response to the unsatisfactory results, a second approach was conducted aiming to address the limitations and strive for improved performance. This involved a complete overhaul of the architecture and inputs used in the initial approach. The objective was to adapt the model's architecture and inputs to better align with the expectations, with the anticipation that these changes would lead to enhanced performance and more accurate detection.

Following, it will be provided a detailed account of the changes made in the architecture and inputs.

In the second approach, which implementation is explicated in Figure 34 from Appendix V, the GCN models were implemented using the PyTorch Geometric library, with two model classes: "NormalGCN" and "SkipGCN."

The "NormalGCN" class takes the number of node features, hidden channels, number of classes, and dropout rate as inputs. It initializes two GCNConv layers, where the first layer

performs graph convolution on the input features, and the second layer maps the hidden channels to the number of classes. The forward method applies ReLU activation after the first GCNConv layer and returns the final output.

The "SkipGCN" class is similar to "NormalGCN" but additionally creates a weight matrix to capture skip connections. The forward method performs graph convolution using the first layer, applies ReLU activation, concatenates the input features with the output of the first layer, and passes it to the second layer. Skip connections are added by computing the element-wise sum of the output and the matrix multiplication of the input features with the weight matrix.

The inputs for the GCN models in the second approach consist of a feature matrix ("x") and an edge index matrix ("edge_index"). The "edge_index" matrix represents the connections between nodes in the graph. It has a shape of (2, E), where E represents the number of edges in the graph. Each column of the "edge_index" matrix corresponds to an edge, with the first row representing the source node and the second row representing the target node of the edge. On the other hand, the feature matrix, denoted as "x," represents the node features and has a shape of (N, F), where N represents the number of transactions or nodes in the graph, and F represents the number of features associated with each transaction. Each row of the feature matrix corresponds to the features of a specific node.

After passing the input features through the graph convolution layers and obtaining the final output, the softmax function was applied to the output. By applying softmax, the GCN models produce a probability distribution over the classes for a given input graph.

To effectively train and evaluate the GCN models, PyTorch Geometric Data objects were created. These Data objects contain the necessary inputs and labels required for training and testing the GCN models.

The training process utilized the Adam optimizer along with the cross-entropy loss function, explained in Graph Convolutional Networks. To address the class imbalance issue, more importance was assigned to the illicit class during training. This was achieved by adjusting the weights assigned to each class in the cross-entropy loss function. In this case, the weights were set to [0.3, 0.7], indicating that the illicit class was assigned a weight of 0.7 and the licit class was assigned a weight of 0.3.

It is important to note that the second approach yielded far superior results compared to the first one. However, even with these improvements, the achieved performance still did not

satisfy the objectives. Consequently, additional measures were explored to further enhance the model's effectiveness in capturing the underlying patterns of fraudulent behavior.
The efforts to improve the model included the following steps:

- Creating a batch by time step: An approach attempted was creating a batch for each time step, as each time step had its own graph. By treating each time step as a separate graph, the model may better capture temporal patterns and enhance its understanding of fraudulent behavior over time.
- Attempting to use mini batching: Mini batching involves training the model on smaller subsets or batches of the data instead of the entire dataset or graph at once. This can speed up training and improve efficiency. However, implementing mini batching in this study proved challenging because the features (x) and the edge indexes must correspond within the same batch. Issues arose with the indexes when creating the batches, preventing the correct implementation of mini batching.
- Train for more epochs: Increasing the number of training epochs allows the model to learn from the data for a longer period, potentially capturing more complex relationships and improving its performance. Additionally, increasing the number of training epochs did not yield the desired improvements, likely due to the problem of overfitting.

Since attempting to use mini-batching and train for more epochs did not produce significant results, these will not be presented in the Results and Discussion chapter. Still, it remains pertinent to document these endeavors as they were undertaken within the scope of this dissertation.

6.2.3 Node Embeddings

In order to gain insights into the role of node embeddings and their impact on performance, a comparative analysis was conducted between the supervised baseline and alternative methods utilizing node embeddings. The objective was to evaluate the effectiveness of incorporating these embeddings, obtained through different techniques, by concatenating them with the original features in the dataset.

To begin with, the embeddings produced by the GCN, described in Graph Convolutional Networks chapter, were extracted. These embeddings were then combined with the original features of the nodes in the dataset.

Additionally, DeepWalk and Node2Vec were implemented to generate node embeddings. These algorithms capture structural information by traversing the graph and encoding nodes as low-dimensional vectors. The resulting embeddings were also concatenated with the original features of the nodes.

By combining the original features with these diverse sets of embeddings, it was aimed to explore the potential benefits and limitations of each embedding technique. Table 13 from Appendix III describes the parameters used to run the Node2Vec and the DeepWalk.

Subsequently, all the node embeddings were tested with RF, LR, and MLP following the scenarios outlined in Supervised Baseline chapter. In the first scenario, each of the four obtained embeddings was combined solely with the local features of the nodes. In the second scenario, both the local and aggregated features were utilized in conjunction with the embeddings. This comprehensive analysis aimed to assess the effectiveness of each embedding technique by examining its performance within different classification models and feature combinations.

To enhance the quality of the embeddings obtained from DeepWalk and Node2Vec, extensive experimentation was conducted with numerous combinations of parameters. This was necessary as the initial embeddings did not yield satisfactory results when employed in the supervised baseline method. By systematically varying the parameters, an effort was made to obtain more representative and informative embeddings that could potentially improve the overall performance of the models. The parameter exploration process aimed to optimize the embedding generation process and ensure a more accurate representation of the underlying graph structure.

7. Results and Discussion

This section provides a detailed summary of the outcomes obtained during the experimentation of the methodologies explained earlier. Moreover, the main discoveries stemming from this study will be examined considering the results.

7.1 Supervised Baseline

The findings of this research indicate that the performance of the supervised baseline models surpassed the results achieved by Weber et al. (2019). This enhancement can be credited to the study's emphasis on identifying the best thresholds for maximizing the F1-Score, instead of relying on the default threshold provided by the sklearn library. By thoroughly investigating the thresholds that produced the highest F1-Scores, this approach significantly improved the model's performance within the scope of this particular study.

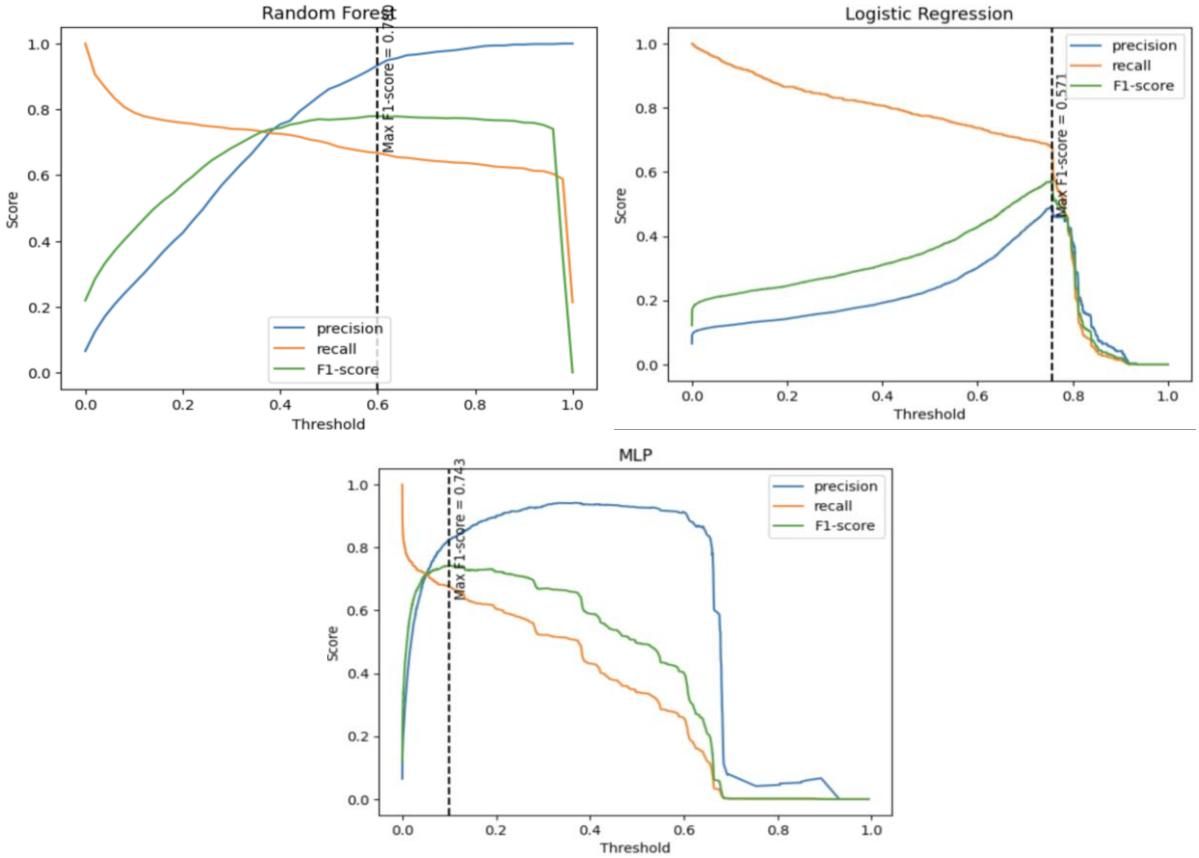

Figure 15: Metrics by Threshold (Local Features)

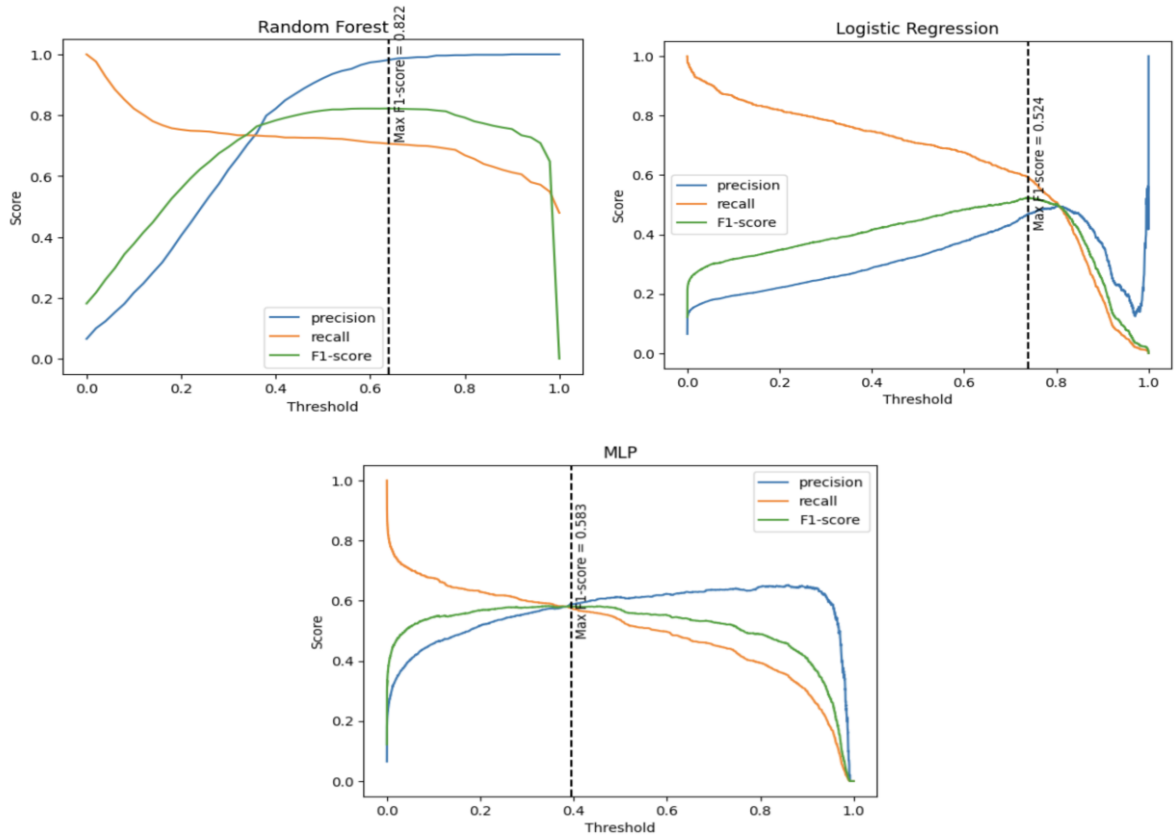

Figure 16: Metrics by Threshold (All Features)

Figure 15 and Figure 16 depict the progression of Precision, Recall, and F1-Score for various thresholds. The threshold that yields the highest F1-Score is determined as the optimal threshold. From there, the corresponding Precision and Recall values are extracted. The outcomes of each model for the supervised baseline are summarized in Table 3. It provides a comprehensive overview of the performance metrics, including Precision, Recall, and F1-Score, for each model under two different scenarios: All Features and Local Features.

Table 3: Results Supervised Baseline

Scenario	Model	Precision	Recall	F1-Score
Local Features	RF	0.950	0.661	0.780
	LR	0.493	0.678	0.571
	MLP	0.826	0.675	0.743
All Features	RF	0.987	0.705	0.822
	LR	0.469	0.593	0.524
	MLP	0.590	0.576	0.583

First and foremost, it is observed that the RF model consistently outperforms both LR and MLP models across both scenarios. This indicates that RF demonstrates superior predictive capability in this specific classification task.

When comparing the performance under different feature sets, it is found that RF exhibits improvement when utilizing All Features as opposed to Local Features. This is particularly evident in the increase in the F1-Score, suggesting that the incorporation of a broader range of features enhances the RF model's overall predictive performance.

On the other hand, the LR and MLP models experience a decline in performance when incorporating All Features. This is contrary to the results achieved by Weber et al. (2019), that found that the Aggregated Features related to the neighbors of each transaction (included in All Features) always improved the performance of the models.

This discrepancy could suggest that these models may be more sensitive to the inclusion of additional features, resulting in a deterioration of their predictive capabilities. However, it is crucial to acknowledge the limitations of this study. Further research should delve into investigating the reasons behind the decline in LR and MLP performance when incorporating All Features and explore potential strategies to mitigate this issue. One possible avenue for exploration is hyperparameter tuning, as the parameters used in this study were aligned with those used by Weber et al. (2019) to ensure a fair comparison. Nevertheless, in practical terms, the optimal parameters for the local features may not be the same when utilizing All Features due to the introduction of extra dimensionality.

Also, the replicability and applicability of the studies conducted by Pocher et al. (2022) and Weber et al. (2019) are questionable, as their reported results have not been achieved even when employing the same methodology. This raises doubts about the generalizability of their findings to other contexts. It is important to consider that the inability to replicate these studies' outcomes suggests potential limitations or unique factors specific to their experimental conditions, which may hinder their applicability beyond the original research settings.

Figure 17 depicts the evolution of the performance of each model by timestep for Local Features.

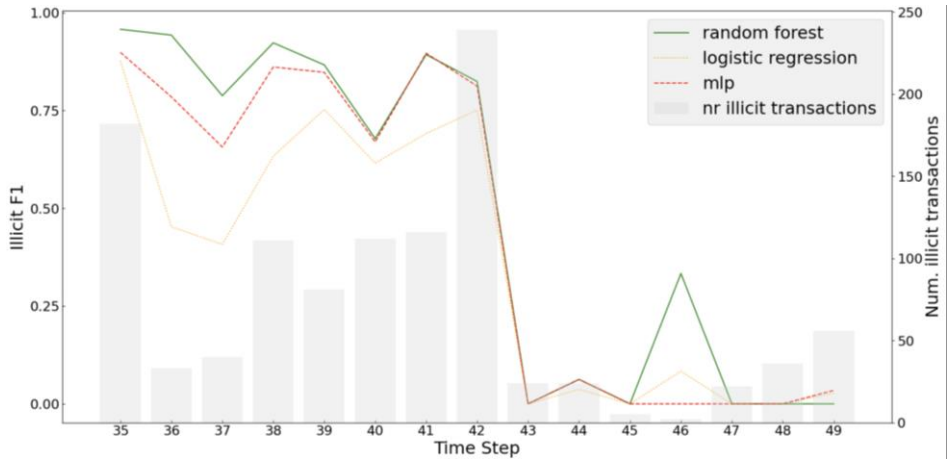

Figure 17: Evolution of F1-Score by Timestep (Local Features)

Figure 18 depicts the evolution of the performance of each model by timestep for All Features.

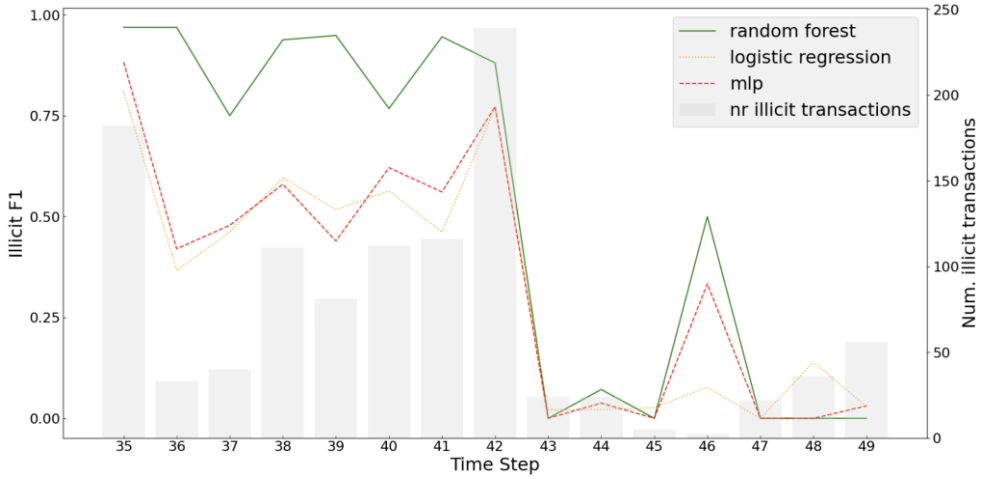

Figure 18: Evolution of F1-Score by Timestep (All Features)

After examining Figure 17 and Figure 18 it becomes clear that the abrupt termination of the dark market, which took place at time step 43, has a noteworthy impact on the performance of each supervised classifier. The decline, particularly regarding the number of illicit transactions, has already been acknowledged in 4.2.2.

In order to analyze the performance of the RF algorithm in predicting transaction labels, Figure 19 presents a visual comparison between the distribution of the actual labels and the labels predicted by RF on Local Features using t-SNE projection.

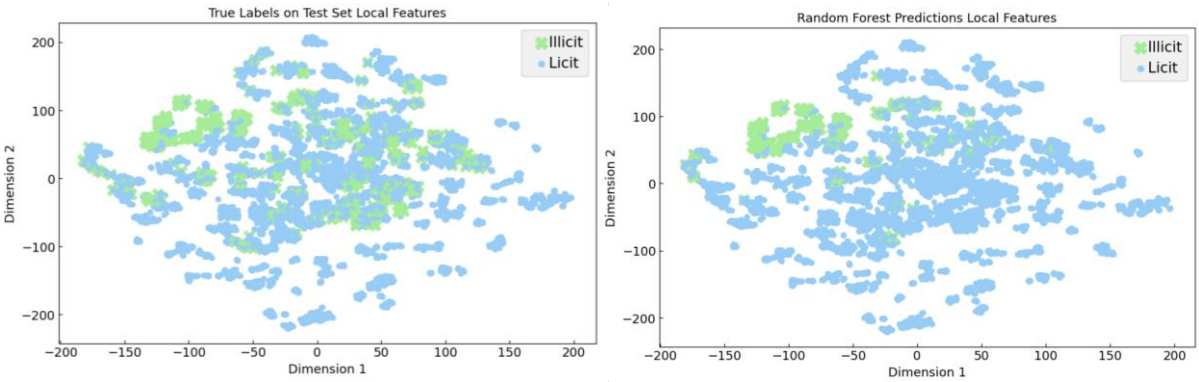

Figure 19: t-SNE Projection on Local Features for Random Forest

It is observed that there is a cluster where RF is very good at predicting illicit transactions and just mislabels a small amount as illicit, which justifies the high Precision of this model in this situation. However, it fails to capture a vast number of illicit transactions, therefore it has a smaller Recall.

Figure 20 presents a visual comparison between the distribution of the actual labels and the labels predicted by LR on Local Features using t-SNE projection.

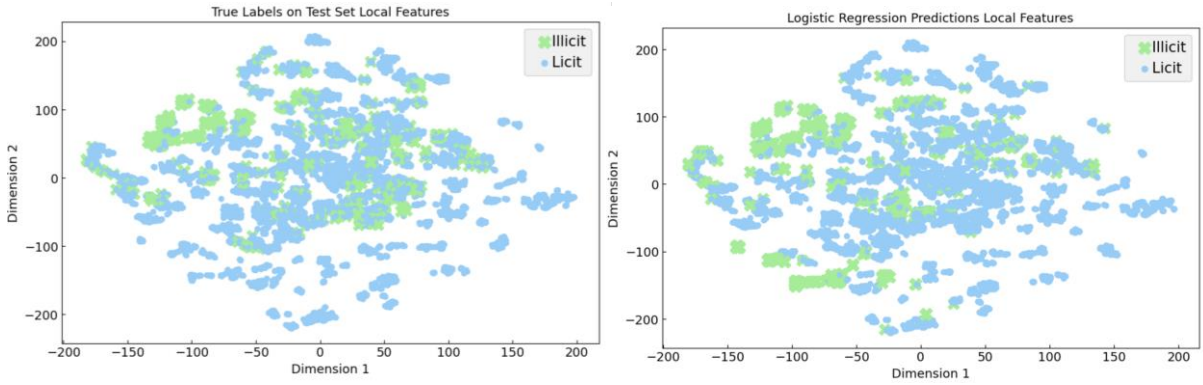

Figure 20: t-SNE Projection on Local Features for Logistic Regression

LR can detect a large number of illicit transactions. However, there is a large number of transactions it mislabels as so which explains its bad results in terms of Precision but a similar Recall to RF.

Figure 21 presents a visual comparison between the distribution of the actual labels and the labels predicted by MLP on Local Features using t-SNE projection.

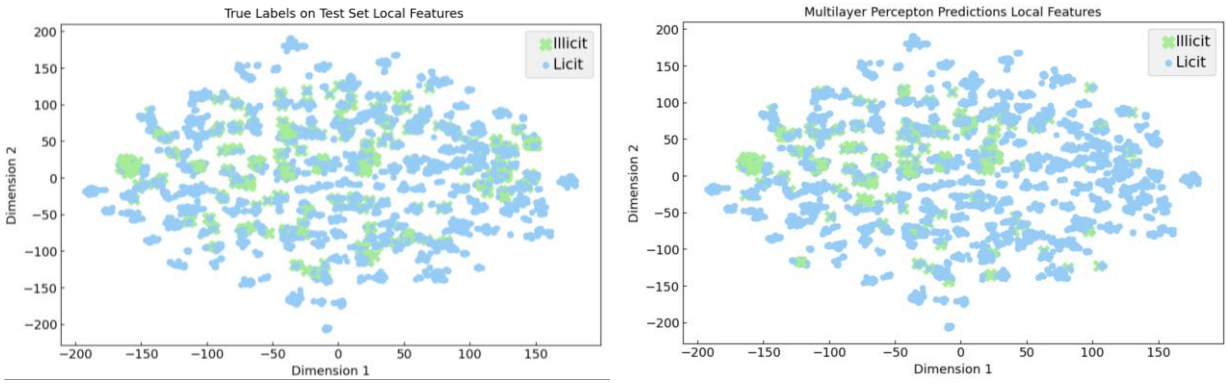

Figure 21: t-SNE Projection on Local Features for Multilayer Perceptron

MLP is able to detect a large number of illicit transactions, however, it mislabels a great amount too. Therefore, MLP does not have such great Precision as RF, but it even has a higher Recall.

Figure 22 presents a visual comparison between the distribution of the actual labels and the labels predicted by RF on All Features using t-SNE projection.

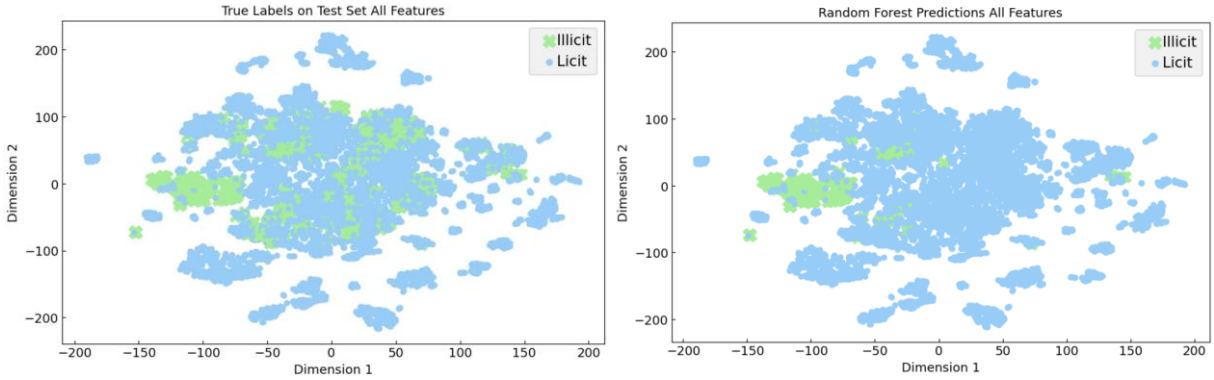

Figure 22: t-SNE Projection on All Features for Random Forest

As it happened with the Local Features, RF captured a cluster of illicit transactions but failed to detect most of the rest. It explains why it has such great Precision, but the Recall is lower.

Figure 23 presents a visual comparison between the distribution of the actual labels and the labels predicted by LR on All Features using t-SNE projection.

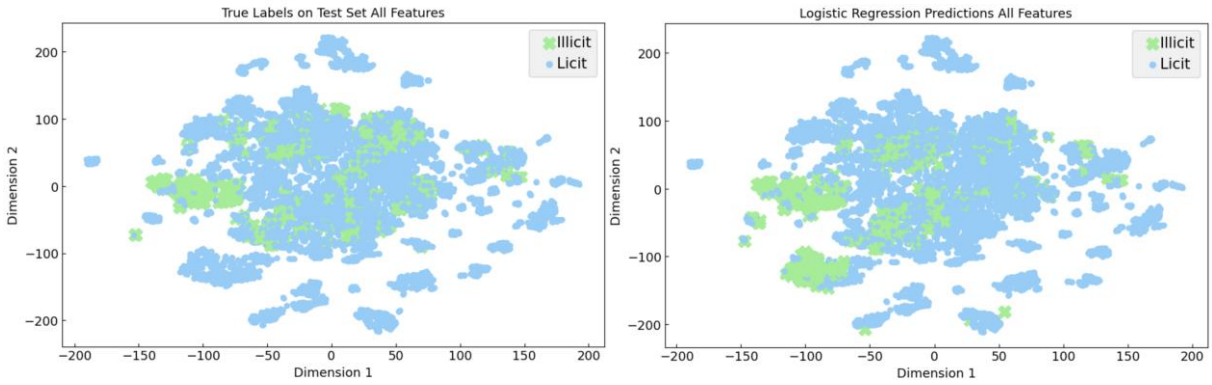

Figure 23: t-SNE Projection on All Features for Logistic Regression

LR model has a high Recall but low Precision. It successfully captures the cluster of illicit transactions identified by the RF model, but it also labels a significant number of legitimate transactions as illicit.

Figure 24 presents a visual comparison between the distribution of the actual labels and the labels predicted by MLP on All Features using t-SNE projection.

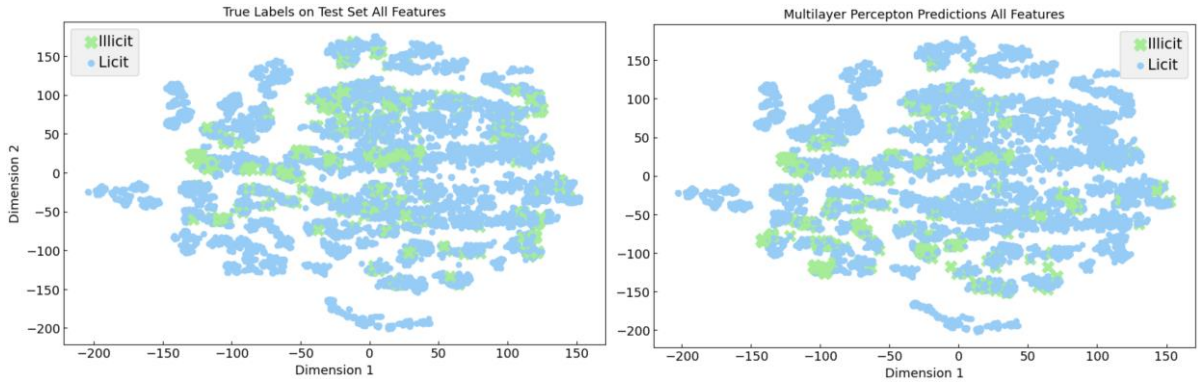

Figure 24: t-SNE Projection on All Features for Multilayer Perceptron

In the case of the Local Features, MLP had a similar performance to the RF. However, in the case of All Features, it mislabels a high number of licit transitions as illicit and fails to detect a vast number of illicit ones, demonstrating poor performance in terms of Recall and Precision.

7.2 Graph Convolutional Networks

Table 4 outlines the results achieved for the GCN and the Skip-GCN.

Table 4: Results Graph Convolutional Networks

Scenario	Model	Precision	Recall	F1-Score
Local Features	GCN	0.767	0.488	0.597
	Skip-GCN	0.770	0.599	0.599
All Features	GCN	0.819	0.527	0.642
	Skip-GCN	0.757	0.577	0.675

Primarily, the models were trained solely using the Local Features as input. Secondly, both models were trained using All Features, encompassing a more comprehensive range of input data. These scenarios were considered to evaluate and compare the performance of the models under different feature combinations, enabling a deeper understanding of their detection capabilities.

The hyperparameters used in the experiments are consistent with the work of Weber et al. (2019), surpassing their reported results for the GCN. However, the results fell short to those obtained by Pocher et al. (2022).

The results indicate that the Skip-GCN model, although not fully achieving the author's results, outperforms the GCN model when All Features are utilized. Nonetheless, the differences observed in both cases are not statistically significant, thereby corroborating the findings of the original authors considering the scenario of All Features.

By incorporating both the input features and the embeddings generated by the convolutional layers, Skip-GCN benefits from the complementary information captured by these components. This suggests that the merging of input features and convolutional embeddings enhances the model's capability to capture and utilize significant information, resulting in improved overall performance as reflected by the higher F1-Score. However, although Skip-GCN exhibits a higher F1-Score compared to GCN, it is worth noting that the difference between the two models' F1-Scores is not substantial. The improvement in F1-Score obtained by Skip-GCN may be considered modest in this context.

When considering only the Local Features, it is observed that both GCN and Skip-GCN demonstrate nearly identical F1-Scores. This implies that in this specific scenario, their performance in terms of F1-Score is comparable. When considering only the Local Features, it is observed that both GCN and Skip-GCN exhibit a lower overall performance compared to when All Features are utilized. This indicates that the exclusion of the aggregated features,

which incorporate graph-related information such as information about the neighbors of each transaction, negatively impacts the models' ability to capture relevant patterns and detect fraudulent behavior.

The results suggest that while GCN is designed to extract meaningful information from graphs, its performance is notably improved when the features specifically related to the underlying graph structure (aggregated features) are already included in the input features. This implies that incorporating the aggregated features provides crucial contextual information to GCN, enhancing its capability to identify and utilize important graph-related patterns. Therefore, it can be inferred that the inclusion of graph-related features contributes significantly to the overall performance of GCN in detecting fraudulent behavior within the cryptocurrency market. However, it is important to note that the input features that consider neighbors' information are distinct from those extracted by GCN, making them complementary rather than exclusive.

To further strengthen this conclusion, it is crucial to address potential concerns of overfitting during the model training process and conduct comprehensive hyperparameter tuning for the GCN model. This step is particularly important considering that Local Features and All Features have different dimensionality, meaning that the optimal parameters for one may not necessarily be suitable for the other. By conducting rigorous hyperparameter tuning, it would be possible to gain valuable insights into optimizing the performance of GCN in detecting fraudulent behavior within the cryptocurrency market using the incorporated graph-related features. By addressing these considerations, the study can provide a more comprehensive and robust validation of the contribution of graph-related features to the overall performance of GCN.

It is worth noting that despite the superior performance of Skip-GCN over GCN, the RF model, a traditional ML method, consistently achieves the highest F1-Score. The literature, as described in the Literature Review chapter, has shown significant interest in emerging technologies such as graph-oriented deep learning methods, with high expectations for their superior performance compared to traditional methods. Deep learning, with its ability to learn hierarchical representations automatically, and graph-oriented methods, which leverage the inherent graph structure to capture relational information, have been considered as promising approaches.

However, it is intriguing to observe that, contrary to these expectations, the RF model consistently outperforms the graph-oriented deep learning methods in the experiments conducted in this research. This finding challenges the notion that graph-oriented deep learning methods are always superior. It emphasizes the importance of considering various factors, including dataset characteristics, the complexity of relationships within the data, and interpretability requirements when selecting an appropriate method.

While graph-oriented methods have demonstrated great potential in specific domains, the performance superiority of RF in this study suggests that the effectiveness of traditional methods should not be overlooked. This highlights the need for careful evaluation and comparison of different techniques, considering the specific context and requirements of the task at hand. It is essential to recognize that no single method can be universally superior, and the selection of an appropriate method should be based on a thorough understanding of the problem and the available data.

Figure 25 presents a visual comparison between the distribution of the actual labels and the labels predicted by GCN on Local Features using t-SNE projection.

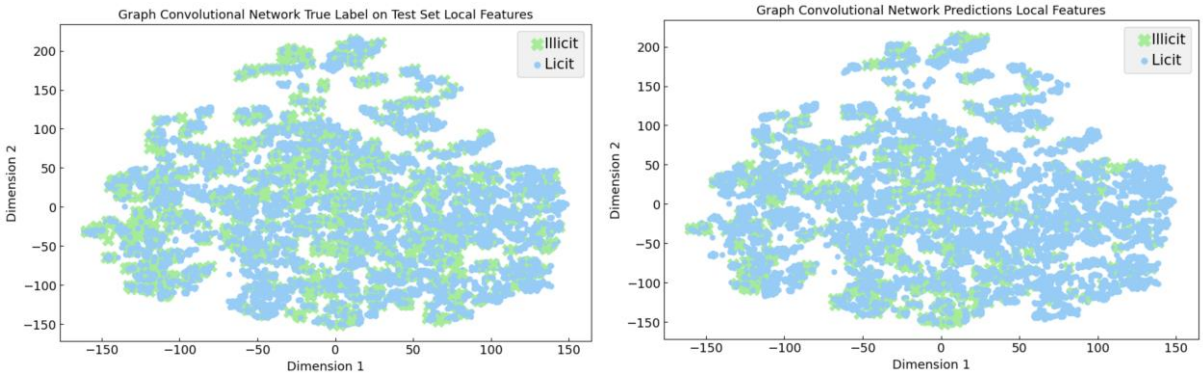

Figure 25: t-SNE Projection on Local Features for Graph Convolutional Network

By Figure 25, it's not visually perceptible what happens in terms of Precision. However, in terms of Recall, it is possible to understand that GCN fails to identify a large number of transactions, which explains the bad performance in this aspect.

Figure 26 presents a visual comparison between the distribution of the actual labels and the labels predicted by Skip-GCN on Local Features using t-SNE projection.

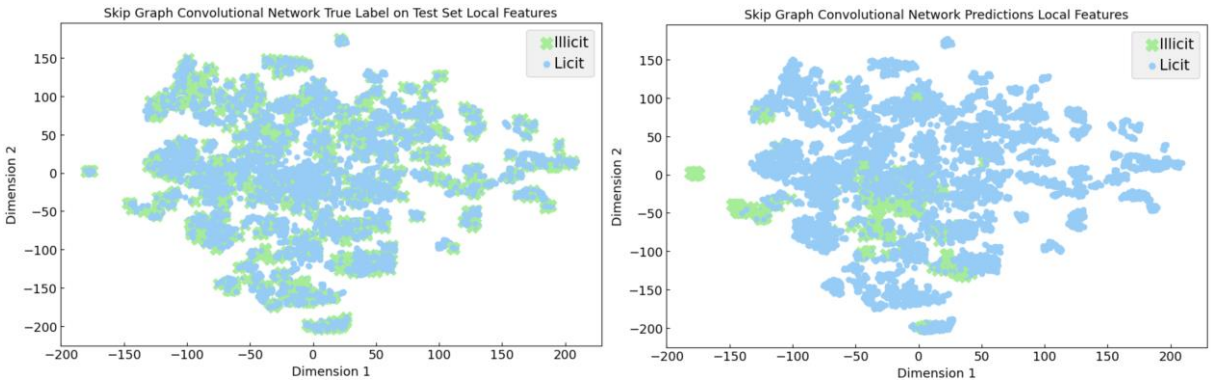

Figure 26: t-SNE Projection on Local Features for Skip-Graph Convolutional Network

It's perceptible by the images that Skip-GCN is missing out on correctly detecting a significant portion of the illicit transactions present in the dataset, which ends up in a low Recall. Also, the model tends to identify certain transaction clusters as illicit, even though not all of them are truly illicit, which does not result in a very high Precision.

Figure 27 presents a visual comparison between the distribution of the actual labels and the labels predicted by GCN on All Features using t-SNE projection.

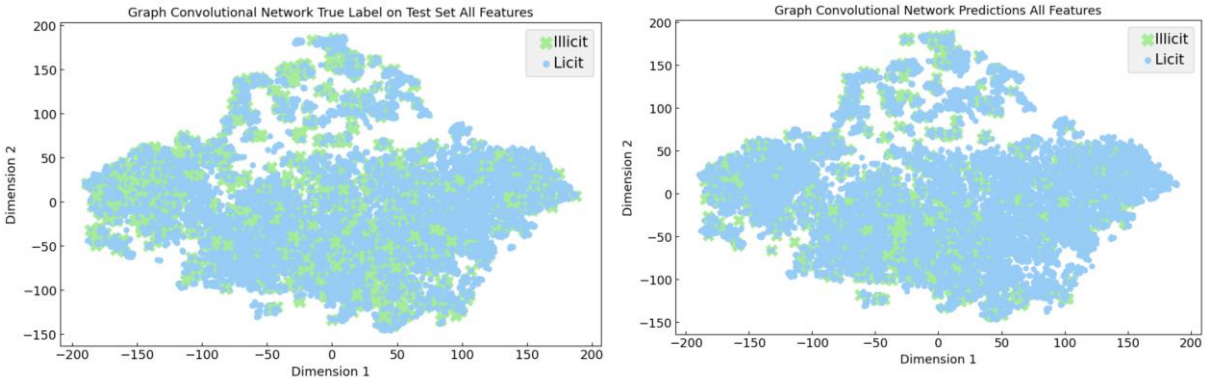

Figure 27: t-SNE Projection on All Features for Graph Convolutional Network

In this scenario, GCN identifies a large number of illicit transactions correctly, having a good result in terms of Precision but it mislabels many transactions as licit, which explains the bad results in terms of Recall.

Figure 28 presents a visual comparison between the distribution of the actual labels and the labels predicted by Skip-GCN on All Features using t-SNE projection.

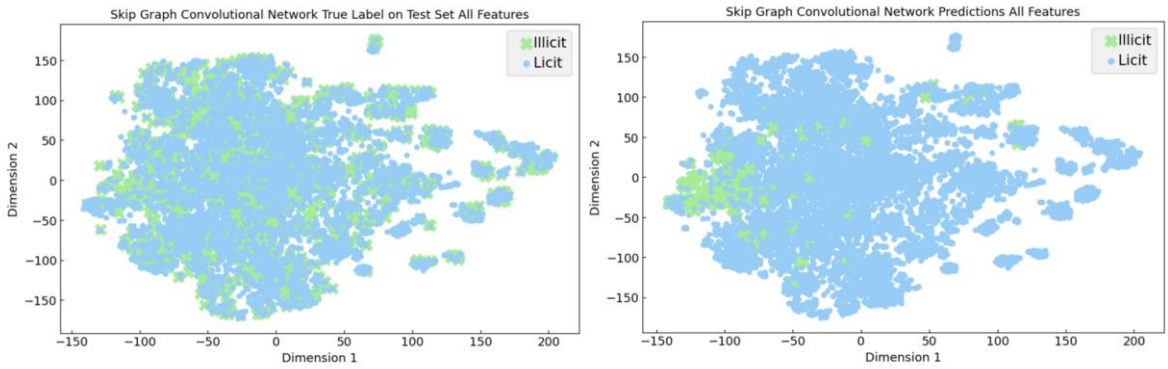

Figure 28: t-SNE Projection on All Features for Skip-Graph Convolutional Network

Skip-GCN (in the scenario it considers All features) presents relatively good results in terms of Precision since most of the transactions it labels as illicit are so. However, there's a large number it fails to label as illicit, mislabeling them as licit which explains why it results in a lower Recall than when it considers only the Local Features

7.3 Node Embeddings

Table 5 presents the outcomes achieved through the combination of embeddings acquired from three techniques: DeepWalk, Node2Vec, and the GCN, along with the original features. The results obtained using the embeddings can be compared with the results obtained in the baseline for each model.

Table 5: Results Node Embeddings

Scenario	Model	Embeddings	Precision	Recall	F1-Score	Baseline F1-Score
Local Features	RF	DeepWalk	0.930	0.698	0.798	0.780
		Node2Vec	0.924	0.698	0.795	
		GCN	0.944	0.688	0.796	
	LR	DeepWalk	0.324	0.584	0.417	0.571
		Node2Vec	0.343	0.591	0.434	
		GCN	0.192	0.681	0.299	
	MLP	DeepWalk	0.341	0.546	0.420	0.743
		Node2Vec	0.650	0.508	0.567	
		GCN	0.786	0.620	0.693	
All Features	RF	DeepWalk	0.981	0.708	0.822	0.822
		Node2Vec	0.979	0.710	0.823	
		GCN	0.983	0.710	0.825	
	LR	DeepWalk	0.411	0.569	0.477	0.524
		Node2Vec	0.462	0.566	0.509	
		GCN	0.427	0.643	0.513	
	MLP	DeepWalk	0.414	0.569	0.477	0.583
		Node2Vec	0.707	0.570	0.631	
		GCN	0.680	0.546	0.598	

The experimental results indicate that the inclusion of node embeddings in conjunction with the original features did not yield a significant improvement in the overall performance of the models when compared to the Baseline F1-Score.

It is worth mentioning that the supervised baseline already achieved very good results, as indicated by the maximum F1-Score threshold. Consequently, the addition of embeddings, which increased the dimensionality of the data, did not yield substantial improvements across the models, especially in the case of LR.

However, it is important to note that these conclusions may not be meaningful due to the additional dimensionality introduced by the node embeddings. As discussed in the Supervised Baseline section, conducting hyperparameter tuning in future research would be advisable to ensure that the best parameters are applied to each set of features, considering the

incorporation of node embeddings. In this study, the same parameters used in the supervised baseline were applied. Therefore, further investigation through hyperparameter tuning is warranted to fully explore the potential benefits of incorporating node embeddings and to optimize their impact on the model performance.

Interestingly, the RF model showed improved performance when utilizing the embeddings. Similarly, the MLP model demonstrated a slight enhancement when incorporating All Features and embeddings. However, it is important to note that these improvements were relatively minimal in both cases.

On the other hand, the LR model consistently exhibited worsened performance when utilizing the embeddings.

In summary, the findings indicate that the inclusion of embeddings in conjunction with the original features did not significantly enhance the performance of the models compared to the Baseline F1-Score. The limited improvements observed in the RF and MLP models, coupled with the consistent performance deterioration in LR, suggest that the impact of embeddings on the overall performance of the models is modest, and their addition may not always be beneficial, particularly when considering dimensionality effects.

The comparison of performance among the three types of embeddings, DeepWalk, Node2Vec, and GCN, in terms of F1-Score reveals interesting insights.

In some scenarios, DeepWalk embeddings demonstrate competitive performance, as seen in the RF model where it achieves an F1-Score of 0.822. This suggests that DeepWalk captures meaningful structural information in the graph, allowing the RF model to make accurate predictions.

Similarly, the GCN embeddings do not consistently exhibit superior performance compared to DeepWalk or Node2Vec embeddings. While the GCN embeddings show promising results in certain scenarios, such as achieving an F1-Score of 0.825 in the RF model under the "All Features" scenario, they do not consistently outshine the other embedding methods.

These findings indicate that the choice of embedding method does not guarantee consistent superiority across all scenarios and models. The performance of each embedding technique can be influenced by various factors, including the characteristics of the dataset, the complexity of the underlying graph structure, and the specific modeling algorithm employed. Therefore, it is crucial to carefully evaluate and select the most appropriate embedding

method based on the specific task and dataset at hand, rather than relying solely on general assumptions.

The observed variability in the performance of different embedding methods highlights the need for a thorough analysis and comparison when incorporating embeddings into ML models. Researchers and practitioners should consider not only the expected performance but also the context-specific characteristics and requirements of the task at hand. Moreover, it is essential to assess the impact of embeddings on different models and scenarios to understand their potential benefits and limitations accurately.

In conclusion, the comparison of DeepWalk, Node2Vec, and GCN embeddings based on their F1-Score performance reveals that Node2Vec does not consistently outperform DeepWalk, and GCN embeddings are not always superior to the other methods. The results emphasize the importance of carefully evaluating and selecting the most appropriate embedding technique based on the specific task, dataset, and modeling algorithm, rather than relying solely on general expectations.

7.4 Results Overview

In this concluding section of the chapter, a comprehensive overview of the results is presented to capture the meaningful conclusions for the objectives of this dissertation, which aimed to assess the feasibility of using graph-oriented deep learning methods, specifically GCN, for detecting fraud and AML in the context of cryptocurrencies.

Figure 29 depicts a comparison of the performance of the Supervised Baseline with the GCN and Skip-GCN.

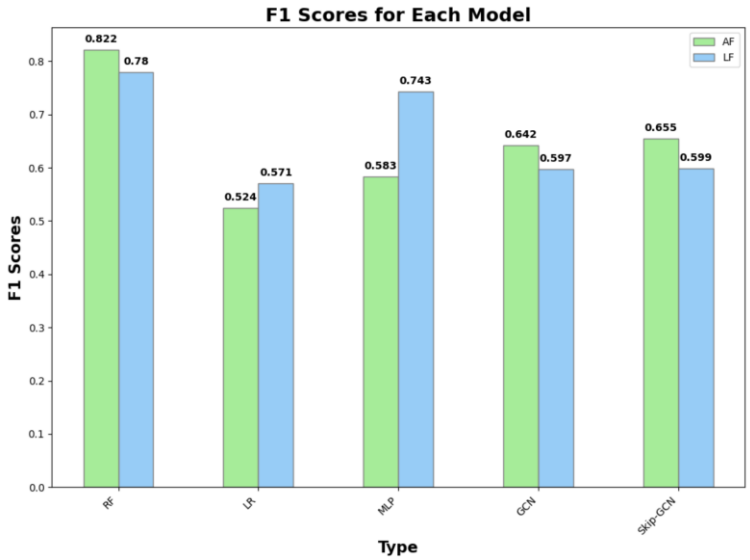

Figure 29: Comparison between Supervised Baseline and Graph Convolution Networks

It is evident from Figure 29 that the RF method outperforms other methods in terms of overall performance. This observation suggests that while graph-oriented methods, GCN and Skip-GCN, demonstrate promising results, the RF approach remains the most effective in this particular scenario. These findings emphasize the importance of considering traditional methods alongside the hype surrounding deep learning and the potential of graph-oriented techniques, as they continue to offer strong performance in the context of fraud and AML detection in cryptocurrencies.

Furthermore, Figure 29 reveals that the performance of the RF model improves when utilizing All Features compared to using only Local Features. This finding indicates that although the RF model cannot inherently capture the graph structure, the process of feature engineering to incorporate graph-related features can significantly enhance its performance. This underscores the importance of considering the network of transactions in the context of detecting fraud and AML in cryptocurrencies, as it provides valuable insights for improving the effectiveness of traditional models like RF.

Figure 30 presents the comparison between the performance of the traditional ML methods in the supervised baseline and their performance when incorporating DeepWalk embeddings.

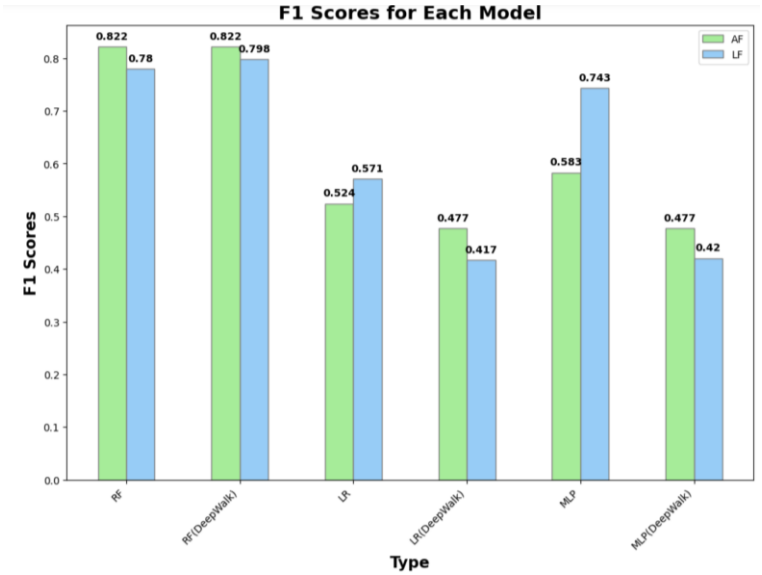

Figure 30: Comparison between Supervised Baseline and the DeepWalk Embeddings

Figure 31 presents the comparison between the performance of the traditional ML methods in the supervised baseline and their performance when incorporating Node2Vec embeddings.

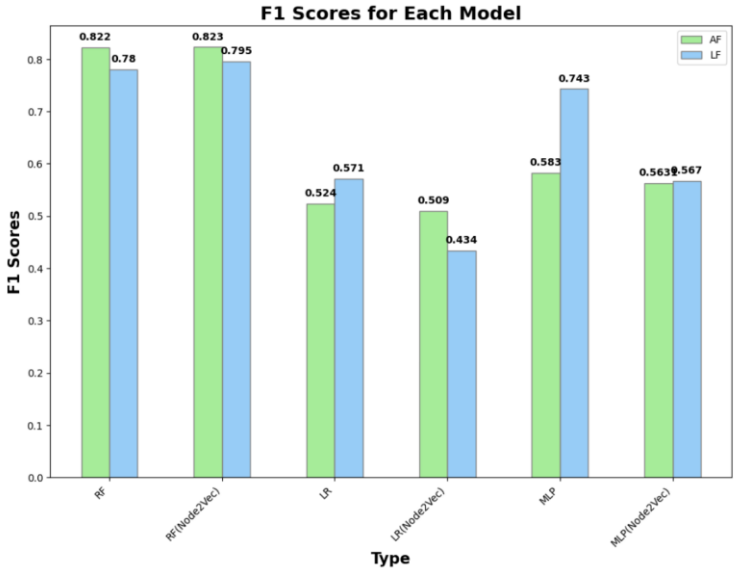

Figure 31: Comparison between Supervised Baseline and the Node2Vec Embeddings

Figure 32 presents the comparison between the performance of the traditional ML methods in the supervised baseline and their performance when incorporating GCN embeddings.

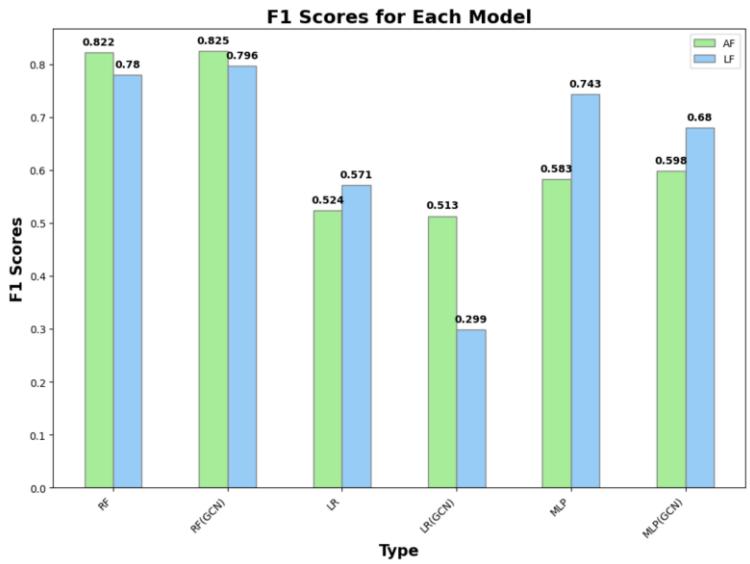

Figure 32: Comparison between Supervised Baseline and the Graph Convolutional Network Embeddings

The results from the analysis indicate that incorporating embeddings, regardless of the method used, generally led to a slight improvement in the performance of the RF algorithm. This suggests that node embeddings have the potential to enhance the detection capabilities of RF when applied to fraud and AML detection in cryptocurrency transactional graphs.

However, it is worth noting that the performance of the LR algorithm consistently worsened when embeddings were incorporated. This could indicate that the specific nature of the embeddings used in this study did not complement the inherent characteristics of LR for this particular task.

It is important to emphasize that these results should not be considered definitive, as hyperparameter tuning was not extensively performed in this research. Therefore, future studies should consider comprehensive hyperparameter tuning to further optimize the performance of the ML methods with node embeddings.

Considering the findings from the literature review and the results obtained in this study, the utilization of node embeddings shows promise for detecting fraud and AML activities in cryptocurrency transactional graphs. However, the impact of node embeddings on the overall detection performance needs to be further examined and fine-tuned to achieve optimal results.

8. APPLICATION OF THE METHOD TO UPHOLD PROPRIETARY DATASET

While the study focused on investigating AML activities, it is important to acknowledge that the Elliptic Dataset was not specifically tailored for AML detection. The dataset's labels encompassed various types of fraudulent transactions, potentially limiting its ability to accurately represent the specific characteristics and patterns associated with AML.

Another notable limitation of the analysis was the exclusion of unlabeled transactions, which constituted the majority of the dataset. Consequently, a significant amount of information in terms of nodes and edges in the graph was disregarded. This exclusion of crucial data could have hindered the performance of GCNs in capturing meaningful node embeddings and detecting intricate relationships since these unknown nodes could have contained useful features or connections to other nodes in the dataset, and their removal might lead to an incomplete representation of the underlying graph structure.

Furthermore, the removal of a substantial number of transactions resulted in a smaller dataset size. This reduction in data volume may have had adversely impacted the model's capacity to extract valuable insights and achieve optimal performance levels.

Therefore, it is imperative to consider exploring this methodology using Uphold's proprietary dataset to address these limitations and continue the study of whether graph-based methods perform better that the traditional approaches. This exploration can assess whether applying these techniques would enhance Uphold's ability to detect suspicious activities within its internal transactional data.

This chapter aims to investigate the Uphold Dataset and assess its potential for applying the same methodology employed in the analysis of the Elliptic Dataset.

It is important to emphasize again that the dataset selection and pre-processing were accomplished as a joint effort within the Uphold team. In addition, a fellow team member was involved in another fraud-related use-case in the context of a dissertation. With a common set of supervisors, it was inevitable that there would be some convergence in the materials employed and the exploration of those resources.

8.1 Uphold’s Dataset

Uphold is a leading global digital money platform that serves individuals in more than 184 countries. It offers a wide range of financial services and supports over 200 currencies, including both traditional and digital options, as well as commodities. Uphold is renowned for its seamless foreign exchange and cross-border remittance solutions, enabling members worldwide to easily manage their finances. One of the standout features of Uphold is its innovative 'Anything-to-Anything' trading experience, allowing customers to trade directly between different asset classes. Uphold's platform is built on proprietary technologies and e-money apps, ensuring robustness, security, and efficiency in every transaction. By utilizing these cutting-edge tools, Uphold aims to redefine individuals' financial asset management in today's rapidly evolving digital landscape.

Uphold dataset encompasses 21,656,252 transactions within the Uphold ecosystem. This dataset covers the period from January 1, 2021, to January 1, 2023, illustrating the transfer of funds among various users. The transactions involve a diverse range of currencies, including conventional ones like the Euro and Dollar, as well as cryptocurrencies such as Bitcoin, Ethereum, and others. Each transaction has a corresponding label indicating if the transaction is licit or fraudulent (money laundering).

The dataset provided represents a collection of transactions between Uphold users, where each line corresponds to a single transaction. Consequently, the dataset establishes edges between the two user nodes involved in each transaction.

It is important to highlight that this dataset differs from the Elliptic Dataset in a significant way. Unlike the Elliptic Dataset, which focuses exclusively on Bitcoin transactions, this dataset encompasses a wider range of currencies. Moreover, while the Elliptic Dataset treats transactions as nodes, in this case, transactions are represented as edges connecting the user nodes.

8.1.1 Dataset Description

Table 6 describes the structure of the dataset.

Table 6: Uphold Dataset Description

Attribute	Description	Type
Origin User	User who sent the funds	categorical
Destination User	User who received the funds	categorical
Created At	Timestamp of the transaction	categorical
Origin Currency	Currency of the funds in the origin	categorical
Destination Currency	Currency of the funds at the destination	categorical
Amount	The amount transacted in United Sates Dollar	float
Application ID	ID of the application that triggered the transaction	categorical
Resulted in SAR	Boolean indicating if the transaction resulted in a Suspicious Activity Report (SAR)	Boolean

The column Resulted in SAR represents the class label and corresponds to an AML flag. This means that in the case of this dataset, the labels are specific for AML, contrary to the Elliptic Dataset.

Table 7 displays the distribution of transactions within the dataset, categorized into different classes.

Table 7: Uphold Dataset Number of Transactions of Each Class

Labels	Number of Transactions
Licit	21477866
Illicit	178385
Unknown	0

The dataset provided by Uphold shows a noticeable imbalance between the number of Licit and Illicit transactions. Interestingly, the dataset has an illicit rate of 0.82%, which differs significantly from the Elliptic Dataset. However, it's important to consider that this Uphold dataset provides a more realistic depiction of the actual prevalence of fraud. However, contrary to the Elliptic Dataset, there are no unlabeled transactions.

It's crucial to recognize the limitations associated with identifying fraudulent behaviors using AML labels. These labels are bound by predefined rules, which can limit the effectiveness of accurately detecting all potentially fraudulent activities.

8.1.2 Preprocessing

To proceed with the subsequent steps effectively, certain modifications were performed on the dataset to ensure its suitability for the given task. These preprocessing steps were implemented to enhance the clarity and quality of the data.

The following steps were undertaken to further refine the dataset for the task at hand:

- **Conversion of Labels:** The boolean labels (f / t) were converted to integers (1 / 0) to facilitate analysis and modeling.
- **Conversion of 'created_at' Column:** The 'created_at' column was transformed into a DateTime format to enable chronological analysis and time-based operations. After the process of Feature Engineering, this column was eliminated since the DateTime format is not appropriate for the ML models.
- **Creation of Transaction ID:** A unique transaction ID was generated for each transaction in order to facilitate identification and tracking throughout the dataset.
- **Conversion of IDs:** The user IDs, currencies, and application IDs, initially represented as strings, were converted to integers. Additionally, a mapping indexing system was created to ensure that the user IDs correspond accurately between the origin and destination.
- **Building a Graph:** In addition to the previous preprocessing steps, a graph structure was constructed based on the dataset. This involved leveraging the origin and destination information from the transactions to create a graph representation. Each user ID was treated as a node in the graph, and the transactions between users were represented as edges connecting the respective nodes.

By performing these preprocessing steps, the dataset was optimized for subsequent tasks, providing clearer and more structured data for analysis and modeling purposes. By incorporating the graph structure into the preprocessing phase, the dataset became enriched with a network representation that could be leveraged for graph-based analyses and algorithms.

8.1.3 Feature Engineering

Feature engineering emerges as a potent method that unlocks valuable information from raw data, generating new variables that can enhance the predictive abilities of FD models. In this chapter, the process of feature engineering applied to the Uphold Dataset will be thoroughly explored, specifically focusing on the creation of numerous novel columns derived from expert knowledge in the field. These fresh features have been carefully designed to potentially improve the detection of fraudulent transactions.

Subsequent sections will delve into the specifics of these engineered features, revealing their distinctive attributes. By delving into these intricacies, the goal is to deepen the understanding of the underlying patterns and insights that can be leveraged to enhance the FD capabilities of the developed system.

Importantly, it should be noted that the label was associated with the transaction, however, as will be further discussed in 8.2, the label was shifted to a user level.

Given the unique characteristics of the dataset, the newly crafted features can be classified into three primary categories: transactional attributes, network characteristics, and user behavior features. This categorization enables a better comprehension of the different types of features involved.

Table 8 depicts the transactional attributes. These attributes provide information about each transaction recorded in the dataset.

Table 8: Uphold Dataset Transactional Attributes Description

Category	Attribute	Description	Type
Transactional Attributes	Transaction ID	ID of each transaction	integer
	Origin User	User who sent the funds	integer
	Destination User	User who received the funds	integer
	Origin Currency	Currency of the funds in the origin	integer
	Destination Currency	Currency of the funds at the destination	integer
	Amount	Amount transacted in USD	float
	Application ID	ID of the application that triggered the transaction	integer
	Year	Year in what the transaction occurred	integer
	Month	Month in what the transaction occurred	integer
	Day	Day in what the transaction occurred	integer

It is important to note that since the graph incorporates transactions, the attributes listed in Table 8, are considered as edge features, providing detailed information about each transaction recorded in the dataset and contributing to the analysis of transactional relationships within the graph.

Table 9 contains the network characteristics which provide insights into the transactional interactions between users in the network.

Table 9: Uphold Dataset Network Characteristics Description

Category	Attribute	Description	Type
Network characteristics	Common Neighbors	Number of common neighbors shared between the origin user and the destination user in the network/graph	integer
	Jaccard Coefficient	Similarity between the sets of neighbors of the origin user and the destination user	float
	Preferential Attachment	Represents the product of the degrees of the origin user and the destination user	integer
	Betweenness Centrality	Importance of the user in the network based on the number of shortest paths that pass through it	float
	Clustering Coefficient	Tendency of the user's neighbors to be connected to each other	float

In the analyzed network, where the nodes represent users and the edges represent transactions between those users, the network characteristics can be attributed accordingly, with the Betweenness Centrality and Clustering Coefficient serving as node features, reflecting the importance of a user within the transactional interactions network and local clustering tendencies, while the Common Neighbors, Jaccard Coefficient, and Preferential Attachment, act as edge features, providing insights into the transactional relationships, the similarity of interactions, and transactional preferences, respectively.

Table 10 describes user behavior features. It includes various attributes related to user transactions and their behavior patterns.

Table 10: Uphold Dataset User Behavior Description

Category	Attribute	Description	Type
User Behaviour	In-degree	Number of Transactions the user received	integer
	Out-degree	Number of Transactions the user sent	integer
	Amount Sent	Amount sent by the user	float
	Amount Received	Amount received by the user	float
	Average Amount Sent	The average amount that the user sent by transaction	float
	Average Amount Received	The average amount that the user received by the transaction	float
	Ratio of Sent/Received Amount	Ration of the amount sent and received by the user	float
	Ratio of Sent/Received Amount	Ration of the amount sent by received by the user	float
	User Frequency	Frequency in which the user transacts	float
	Active days	Number of days the user made transactions	integer
	Total Transactions	Total number of transactions the user was involved	integer
	Days Since Last Transaction	Number of days elapsed since the user made his last transaction	integer
	Days Since First Transaction	Number of days elapsed since the user made his first transaction	integer
	Amount Difference from Last Transaction	Difference of the amount transacted from the last transaction made by the user	float
	Range of Amount Sent	Difference of the maximum and minimum amount sent by the user	float
	Range of Amount Received	Difference of the maximum and minimum amount received by the user	float
	Resulted in SAR	Boolean indicating if the user was involved in a Suspicious Activity Report (SAR)	Boolean

Since all the features listed in Table 10 pertain to user-related attributes, they can be classified as node features, providing valuable insights and characterizing the properties and characteristics of individual nodes within the graph.

8.2 Dataset Applicability Discussion

By applying the same method to the Uphold Dataset, the objective is to determine whether graph-based methods outperform traditional ML approaches in the context of Uphold transactions.

It has already been discussed that on the Elliptic Dataset, RF (a traditional approach outperformed other methods), including the GCN (a graph-based approach). It was also observed that the node embeddings produced by GCN, Node2Vec, and DeepWalk did not play a significant role in improving the performance of the traditional ML methods.

Hence, the aim is to investigate whether the Uphold Dataset possesses unique characteristics that can potentially yield significant results and provide valuable insights, possibly differing from the findings of the Elliptic Dataset.

However, the application of the same method used on the Elliptic Dataset to the Uphold Dataset poses several questions and limitations that require careful consideration and discussion.

The Uphold Dataset, being specific to AML, provides a more comprehensive perspective on the application of ML compared to the Elliptic Dataset, as it specifically focuses on AML tasks. In contrast, the Elliptic Dataset addressed fraud in general, encompassing a broader range of fraudulent activities.

One advantage of the Uphold Dataset is that it does not contain any nodes with unknown labels, which contributes to a more complete dataset. Consequently, there is no need to remove any nodes during preprocessing, ensuring that all available information is utilized in the subsequent analysis. By leveraging a complete dataset, the ML algorithms applied to the dataset can potentially capture a richer representation of the underlying graph structure, especially the GCN and the node embedding methods.

Another noteworthy advantage of the Uphold Dataset is its significantly larger scale compared to the Elliptic Dataset. The increased size of the Uphold Dataset offers a more extensive and diverse collection of AML-related data points, enabling a more comprehensive analysis and

potentially yielding more robust and accurate ML models. The larger dataset allows for a more thorough exploration of the underlying patterns and relationships, enhancing the potential effectiveness of the applied ML algorithms in detecting money laundering activities.

Furthermore, it is important to acknowledge that the increased size and higher connectivity of the Uphold Dataset introduce additional challenges in terms of computational requirements. The larger dataset, coupled with the absence of node removal during preprocessing, leads to higher dimensionality and potentially more interconnected nodes in the graph. As a result, implementing ML methods on this dataset necessitates a significantly higher processing computational capacity. The increased computational demands pose a substantial limitation in effectively applying ML techniques to the Uphold Dataset. The processing power required to train and optimize models on such a large and complex dataset can be extensive.

Alternatively, applying undersampling to address the issue of dataset size and imbalance in the Uphold Dataset would require an extensive level of undersampling, given its exceptionally large size and a high degree of imbalance, which may not be the optimal approach.

The Uphold Dataset and the Elliptic Dataset differ in terms of the roles assigned to nodes and edges. In the Uphold Dataset, nodes represent individual users, while edges represent transactions conducted by these users. The labels are associated with the transactions, making it challenging to perform node classification directly on this dataset.

However, a possible approach to enable node classification in the Uphold Dataset is by converting the label to a user's perspective. Since money laundering transactions involve both users, it can be assumed that both users involved in such transactions are fraudulent. This stems from the inherent nature of money laundering, where both parties involved in a fraudulent transaction are often aware of the illegality of their activities and can be categorized as fraudsters. This perspective shift allows the label to be relative to the nodes rather than the edges, making node classification feasible in this case.

Nevertheless, it is important to note that the use case and problem addressed with the Uphold Dataset will still differ from the Elliptic Dataset. With the Uphold Dataset, the objective would be to predict and classify users, whereas the Elliptic Dataset focuses on predicting and classifying individual transactions.

The usage of predefined rules to label activities in the Uphold Dataset introduces certain limitations that should be considered. These limitations stem from the classification of

transactions as fraudulent (money laundering) based solely on predetermined criteria, without considering the broader context. As a result, there is a possibility that some flagged activities may be deemed potentially fraudulent, but upon further examination or judgment in a tribunal, they might be found to be non-fraudulent. The outcome of these judgments is not known in advance, which adds an element of uncertainty to the labeling process. Therefore, it is crucial to acknowledge that the limitations associated with the predefined rules, particularly FP for labeling, have the potential to undermine the reliability of the ML models applied to the Uphold Dataset.

It is crucial to fully comprehend the potential and limitations of applying the method to the Uphold Dataset. Although this dataset is more specific to the problem at hand and encompasses a comprehensive network with unknown label nodes, it poses challenges due to its larger size and imbalanced nature. Additionally, it is important to note that the constructed graph is based on users rather than transactions, and the labels are derived from predefined system rules.

Exploring the application of this method on the Uphold Dataset could provide valuable insights into the performance of these methods when dealing with a significantly lower fraud rate (0.82% in Uphold's dataset compared to 10% in the Elliptic Dataset). Furthermore, it would be interesting to investigate whether the transition from predicting on transactions to predicting on users yields different outcomes. Since no nodes were eliminated in this case, it becomes intriguing to understand if the GCN and node embeddings can effectively identify more critical information and yield better results.

Additionally, it is worth noting that the features in the Uphold Dataset, as far as available information suggests, differ from those in the Elliptic Dataset. Therefore, it would be valuable to investigate whether there exists a specific set of features that performs better in this particular context. Alternatively, exploring whether the features derived from the feature engineering process can enhance the model's performance would also be of interest.

However, a notable challenge arises when attempting to split the data between training and testing sets, as all nodes in the network are interconnected. It becomes crucial to ensure that the nodes present in the training set do not overlap with the nodes in the testing set, and vice versa. This challenge stems from the difficulty of finding two separate graphs (one for training and the other for testing) that have no interconnections. This interconnectivity poses a dilemma since splitting the data in a traditional sense may require removing or isolating

certain nodes from the graph, potentially resulting in a loss of valuable information. Further, this also prevents data leakage and ensures that the model is tested on unseen user interactions.

8.3 Implementation Constraints

During the internship at Uphold, the dissertation project faced significant time constraints due to the limited internship duration and an impending end date. The goal of exploring the Uphold dataset required a substantial amount of time due to its extensive dimensionality.

Unfortunately, the dataset was provided by the company only towards the end of the internship, leaving very little time for in-depth analysis and subsequent implementation of the method. Despite efforts to understand the dataset's intricacies and perform feature engineering, the compressed timeline posed a major challenge.

Moreover, the dataset's size and increased complexity, in comparison to the Elliptic Dataset, demanded a considerable amount of time for processing and experimentation. For instance, Logistic Regression, one of the explored methods, required an excessive two-day runtime during the experimental phase due to the dataset's magnitude.

Regrettably, as the internship concluded, the implementation process could not be continued due to privacy concerns and the data security protocols in place at Uphold. Consequently, no conclusive results or findings could be derived from this phase of the project.

9. CONCLUSIONS AND FUTURE WORK

The rise of cryptocurrencies has brought about a fundamental shift in the way financial transactions are conducted, offering new avenues for innovation and financial inclusion. However, this digital revolution has also presented unprecedented challenges, particularly concerning the detection and prevention of illicit activities such as money laundering. As cryptocurrencies gain traction, organizations must equip themselves with robust measures and strategies to combat these illicit activities effectively. The implementation of AML controls becomes indispensable in safeguarding the integrity of the cryptocurrency market. Therefore, it is important to research this topic, as it is essential to understand how illicit transactions can be detected in a real cryptocurrency ecosystem.

As the transactions of cryptocurrencies between users can be represented as a network, this research aimed to investigate whether graph-oriented methods offer superior advantages over traditional ML methods.

Primarily, extensive research has been conducted to thoroughly examine the fundamental concepts crucial for this dissertation. Specifically, a comprehensive understanding has been established regarding the mechanics of cryptocurrencies, the detection and prevention of fraud and money laundering, the principles, and practices of AML, the application of ML techniques in AML, and the potential impact of graphs, graph analysis, and graph-oriented methods within this domain. Following, a thorough literature review was conducted that encompassed recent advancements in the domain of using ML for AML and FD, along with the study of graph analysis and graph learning techniques. This review aimed to identify the most effective methods and those that have been extensively studied. This research involved an extensive analysis of 54 scientific articles.

After completing the initial study and research phase, the dissertation moved into a more practical phase. As a result of the unavailability of a proprietary dataset comprising genuine transactional data at the beginning, an alternative widely utilized dataset in the field, known as the Elliptic Dataset, was selected for research purposes. The Elliptic Dataset provides a representative sample of cryptocurrency transactions and has been widely used in previous research on FD and AML. To begin the practical phase, exploratory data analysis was conducted on the Elliptic Dataset. This analysis aimed to understand the dataset's features

and distribution of transactions. It also allowed for a better understanding of the dataset's suitability for the research objectives.

In parallel with the data analysis, a study conducted by Weber et al. (2019) was selected since it provides a suitable framework to evaluate the feasibility of using ML and graph analysis in the context of AML. By selecting this study, it is aimed to draw valuable insights that can be applied to Uphold's platform, which serves as one of the focuses of the research. Weber's study is particularly relevant for comparison and potential adaptation to the Uphold context due to its focus on a similar problem.

Building on Weber's study, a range of algorithms were selected and described for implementation in this research. The initial phase involved the implementation of traditional ML methods, namely RF, LR, and MLP, which served as the baseline models for performance comparison against the graph-oriented methods. These traditional methods have been widely used in AML and FD research, providing a solid foundation for evaluation. Subsequently, more sophisticated techniques were employed, including GCN and Skip-GCN. Furthermore, strategies that incorporate node embeddings were applied, specifically GCN embeddings, Node2Vec, and DeepWalk.

Additionally, to assess the impact of feature engineering on the construction of graph-related features, the investigation was divided into two scenarios. The first scenario focused solely on Local Features, which are transaction-related features inherent to the dataset. In the second scenario, these Local Features were concatenated with aggregated features derived from the graph structure, which corresponds to the All Features scenario.

Regarding the supervised baseline, the RF method demonstrated the best performance in terms of F1-Score for both feature sets, namely All Features and Local Features, achieving scores of 0.78 and 0.822, respectively. It is worth noting that incorporating graph-related features improved the performance of RF, while it adversely affected the performance of LR and MLP models. Intuitively this would suggest that incorporating graph-related features worsens the model's performance, this might not be the case, because hyperparameter tuning would be necessary. RF exhibited exceptional Precision, meaning it made highly accurate positive predictions.

In the analysis of the GCN and Skip-GCN models, it was found that Skip-GCN demonstrated superior performance compared to GCN. Interestingly, during the experimentation phase, it was observed that both GCN and Skip-GCN achieved better results when utilizing the All

Features scenario, which includes both local transaction-related features and graph-derived aggregate features. It is crucial to conduct further research to ensure that overfitting is not occurring and to validate these findings. Nevertheless, these initial results suggest that while the GCN is designed to leverage the graph structure, its performance is enhanced when incorporating graph-related features as input alongside the Local Features.

However, it is important to note that despite the promising performance of the graph-based models, the traditional ML method, RF, still outperformed both the GCN and Skip-GCN models. These results challenge the common belief that graph-oriented deep learning methods are always superior in graph-related tasks. It highlights the significance of considering multiple factors, such as dataset characteristics, the complexity of relationships within the data, and interpretability requirements when choosing an appropriate modeling approach.

Furthermore, the analysis of embeddings did not yield a definitive conclusion. While the performance of the supervised baseline methods was not significantly improved, further research on hyperparameter tuning is necessary to address the challenge of increased dimensionality. It is important to explore different embedding techniques and fine-tune their parameters to fully leverage the potential of embeddings in enhancing the performance of graph-based models.

It is important to note that all the implemented code is available on GitHub and can be accessed at https://github.com/AnaPaulaMartins886/Fraud-Detection-and-AML-with-ML-and-GCN.

In the final stage of the research, an opportunity arose to work with real data provided by Uphold, the sponsor of the dissertation. This allowed for an assessment of the applicability of the methodology on their proprietary dataset. The discussion encompassed various aspects, including an examination of the dataset itself, the process of feature engineering, and a comparison of the dataset's advantages over the Elliptic dataset. Additionally, the limitations encountered prior to the implementation of the methodology were discussed, while also exploring potential approaches for its execution. However, due to time constraints and the end of the internship, the implementation process could not continue.

The main challenge encountered in this dissertation was the significant computational time required to train the models, particularly the more complex ones. These advanced models demand extensive computational resources and can be time-consuming to train effectively.

To address this issue, Uphold provided a higher-performance personal computer with enhanced computational resources, enabling more efficient and effective training of the complex models. Even so, it still took considerable time to train the models. The extended training time poses a challenge in terms of efficiency and scalability when applying these models to large datasets or in real-time applications. Another major difficulty faced during the research was the understanding and preprocessing of inputs for the GCN and Skip-GCN models. These graph-based models operate on graph structures, requiring a deep understanding of graph theory and the specific implementation details. Additionally, there is a lack of standardized architectures and readily available libraries for their implementation. It is necessary to construct the GCN model by instantiating the appropriate layers and then train and evaluate the model using the provided APIs. This necessitated significant time and effort in investigating and designing appropriate architectures for the specific research objectives.

This research significantly contributes to the expanding knowledge base in the field of cryptocurrency FD and offers valuable insights for future studies seeking to enhance the efficacy of fraud and AML detection in this domain. By conducting a thorough analysis and evaluation, it was gained a deeper understanding of the underlying mechanisms and factors that influence the effectiveness of these detection methods.

In the pursuit of future work, the following areas can be further enhanced and investigated:

- **Hyperparameter Tuning:** It is important to delve into hyperparameter tuning to gain a comprehensive understanding of the impact caused by the increase in dimensionality when incorporating additional features.
- **Semi-supervised Learning:** Since the majority of transactions in the dataset were unlabeled, incorporating semi-supervised learning techniques can be a valuable avenue for future research. By leveraging both labeled and unlabeled data, semi-supervised learning methods can provide a more comprehensive representation of the transactional graph and enhance fraud and AML detection capabilities.
- **Evaluation of Graph-Oriented Methods:** Expanding the scope of research involves testing various graph embedding techniques and exploring alternative graph neural networks. By doing so, we can gain insights into their efficacy and compare their performance against the existing methods.

- **Diversifying Real-World Datasets:** To ensure the validity and generalizability of the findings, it is advisable to extend the exploration of these approaches to other real-world datasets. This will help determine whether the observed results hold in different contexts.
- **Innovative Feature Engineering Techniques:** Exploring novel feature engineering techniques is crucial to capture the intricate relationships within the cryptocurrency transaction network. Consider incorporating additional graph-related features or deriving new features from the dataset, thereby uncovering hidden patterns, and improving the overall performance of the models.

By addressing these areas, one can advance our understanding and make more strides in the field of cryptocurrency analysis and graph-based modeling.

REFERENCES

Alarab, I., & Prakoonwit, S. (2022). Graph-Based LSTM for Anti-money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data. *Neural Processing Letters*. https://doi.org/10.1007/s11063-022-10904-8

Alarab, I., Prakoonwit, S., & Nacer, M. I. (2020a). *Comparative Analysis Using Supervised Learning Methods for Anti-Money Laundering in Bitcoin*. https://doi.org/10.1145/3409073.3409078

Alarab, I., Prakoonwit, S., & Nacer, M. I. (2020b). *Competence of Graph Convolutional Networks for Anti-Money Laundering in Bitcoin Blockchain*. https://doi.org/10.1145/3409073.3409080

Albrecht, C., Duffin, K. M. K., Hawkins, S., & Morales Rocha, V. M. (2019). The use of cryptocurrencies in the money laundering process. *Journal of Money Laundering Control*, *22*(2), 210–216. https://doi.org/10.1108/JMLC-12-2017-0074/FULL/PDF

Astrakhantseva, I., Astrakhantsev, R., & Los, A. (2021). Cryptocurrency fraud schemes analysis. *SHS Web of Conferences*, *106*, 02001. https://doi.org/10.1051/shsconf/202110602001

Bank, E. C. (2015). Virtual currency schemes – a further analysis. In *European Central Bank* (Issue February).

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bhowmik, M., Sai Siri Chandana, T., & Rudra, B. (2021). Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain. *Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021*, 539–541. https://doi.org/10.1109/ICCMC51019.2021.9418470

Brenig, C., Accorsi, R., & Müller, G. (2015). Economic Analysis of Cryptocurrency Backed Money Laundering. *ECIS 2015 Completed Research Papers*. https://doi.org/10.18151/7217279

Caglayan, M., & Bahtiyar, S. (2022). Money Laundering Detection with Node2Vec. *Gazi University Journal of Science*, *35*(3), 854–873. https://doi.org/10.35378/GUJS.854725

Chen, Z., le Dinh, ·, Khoa, V., Ee, ·, Teoh, N., Nazir, · Amril, Ettikan, ·, Karuppiah, K., Kim, ·, & Lam, S. (2018). Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Knowl Inf Syst*, *57*, 245–285. https://doi.org/10.1007/s10115-017-1144-z

Choithani, T., Chowdhury, A., Patel, S., Patel, P., Patel, D., & Shah, M. (2022). A Comprehensive Study of Artificial Intelligence and Cybersecurity on Bitcoin, Crypto Currency and Banking System. *Annals of Data Science*, 1–33. https://doi.org/10.1007/S40745-022-00433-5/TABLES/2

Daigavane, A., Ravindran, B., & Aggarwal, G. (2021). Understanding Convolutions on Graphs. *Distill*, *6*(9), e32. https://doi.org/10.23915/DISTILL.00032

Dyntu, V., & Dykyi, O. (2018). CRYPTOCURRENCY IN THE SYSTEM OF MONEY LAUNDERING. *Baltic Journal of Economic Studies*, *4*(5), 75–81. https://doi.org/10.30525/2256-0742/2018-4-5-75-81

Eigelshoven, F., Parry, D., & Eigelshoven, F. (2021). *AIS Electronic Library (AISeL) Cryptocurrency Market Manipulation – A Systematic Literature Review Cryptocurrency Market Manipulation : A Systematic Literature Review*. 0–17.

Fawaz Ibrahim, R., Mohammad Elian, A., & Ababneh, M. (2021). *Illicit Account Detection in the Ethereum Blockchain Using Machine Learning*. https://doi.org/10.1109/ICIT52682.2021.9491653

Gandal, N., Hamrick, J. T., Moore, T., & Oberman, T. (2018). Price manipulation in the Bitcoin ecosystem. *Journal of Monetary Economics*, *95*, 86–96. https://doi.org/10.1016/j.jmoneco.2017.12.004

Geng, Z., Cao, Y., Li, J., & Han, Y. (2022). Novel blockchain transaction provenance model with graph attention mechanism. *Expert Systems with Applications*, *209*. https://doi.org/10.1016/J.ESWA.2022.118411

Han, J., Huang, Y., Liu, S., & Towey, K. (2020). Artificial intelligence for anti-money laundering: a review and extension. *Digital Finance*, *2*(3–4), 211–239. https://doi.org/10.1007/s42521-020-00023-1

Kamišalić, A., Kramberger, R., & Fister, I. (2021). Synergy of Blockchain Technology and Data Mining Techniques for Anomaly Detection. *Applied Sciences 2021, Vol. 11, Page 7987*, *11*(17), 7987. https://doi.org/10.3390/APP11177987

Khazane, A., Rider, J., Serpe, M., Gogoglou, A., Hines, K., Bruss, C. B., & Serpe, R. (2019). DeepTrax: Embedding Graphs of Financial Transactions. *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, 126–133. https://doi.org/10.48550/arxiv.1907.07225

Khrestina, M. P., Dorofeev, D. I., Kachurina, P. A., Usubaliev, T. R., Dobrotvorskiy, A. S., Khrestina, M. P., Dorofeev, D. I., Kachurina, P. A., Usubaliev, T. R., & Dobrotvorskiy, A. S. (2017). Development of Algorithms for Searching, Analyzing and Detecting Fraudulent Activities in the Financial Sphere. In *European Research Studies Journal: Vol. XX*.

Kim, J., Lee, S., Kim, Y., Ahn, S., & Cho, S. (2023). Graph Learning-Based Blockchain Phishing Account Detection with a Heterogeneous Transaction Graph. *Sensors 2023, Vol. 23, Page 463*, *23*(1), 463. https://doi.org/10.3390/S23010463

Korejo, M. S., Rajamanickam, R., & Muhamad, M. H. (2021). The concept of money laundering: a quest for legal definition. *Journal of Money Laundering Control*, *24*(4), 725–736. https://doi.org/10.1108/JMLC-05-2020-0045/FULL/PDF

Labanca, D., Primerano, L., Markland-Montgomery, M., Polino, M., Carminati, M., & Zanero, S. (2022). Amaretto: An Active Learning Framework for Money Laundering Detection. *IEEE Access*, *10*, 41720–41739. https://doi.org/10.1109/ACCESS.2022.3167699

Li, R., Liu, Z., Ma, Y., Yang, D., & Sun, S. (2022). Internet Financial Fraud Detection Based on Graph Learning; Internet Financial Fraud Detection Based on Graph Learning. *IEEE Transactions on Computational Social Systems*, *PP*. https://doi.org/10.1109/TCSS.2022.3189368

Lin, D., Wu, J., Yuan, Q., & Zheng, Z. (2020). T-EDGE: Temporal WEighted MultiDiGraph Embedding for Ethereum Transaction Network Analysis. *Frontiers in Physics*, *8*. https://doi.org/10.3389/FPHY.2020.00204

Lopes, D. D., Cunha, B. R. da, Martins, A. F., Gonçalves, S., Lenzi, E. K., Hanley, Q. S., Perc, M., & Ribeiro, H. v. (2022). Machine learning partners in criminal networks. *Scientific Reports 2022 12:1*, *12*(1), 1–9. https://doi.org/10.1038/s41598-022-20025-w

Lorenz, J., Inês Silva, M., Aparício, D., João Tiago Ascensão, F., Pedro Bizarro, F., & Tiago Ascensão, J. (2020). *Machine learning methods to detect money laundering in the Bitcoin blockchain in the presence of label scarcity*. https://doi.org/10.1145/3383455

Ma, X., Wu, J., Member, S., Xue, S., Yang, J., Zhou Quan Sheng, C. Z., Xiong, H., & Akoglu, L. (2021). *A Comprehensive Survey on Graph Anomaly Detection with Deep Learning*. 1. https://doi.org/10.1109/TKDE.2021.3118815

Mabunda, S. (2018). Cryptocurrency: The New Face of Cyber Money Laundering. *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems, IcABCD 2018*. https://doi.org/10.1109/ICABCD.2018.8465467

Monamo, P. M., Marivate, V., & Twala, B. (2017). *A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers*. 188–194. https://doi.org/10.1109/ICMLA.2016.0039

Motamed, A. P., & Bahrak, B. (2019). Quantitative analysis of cryptocurrencies transaction graph. *Applied Network Science*, *4*(1). https://doi.org/10.1007/S41109-019-0249-6

Ostapowicz, M., & Żbikowski, K. (2019). Detecting Fraudulent Accounts on Blockchain: A Supervised Approach. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11881 LNCS*, 18–31. https://doi.org/10.48550/arxiv.1908.07886

Pocher, N., Zichichi, M., Merizzi, F., Shafiq, M. Z., & Ferretti, S. (2022). *Detecting Anomalous Cryptocurrency Transactions: an AML/CFT Application of Machine Learning-based Forensics*.

Rejeb, A., Rejeb, K., & Keogh, J. G. (2021). Cryptocurrencies in Modern Finance: A Literature Review. *ETIKONOMI*, *20*(1), 93–118. https://doi.org/10.15408/ETK.V20I1.16911

Sabry, F., Labda, W., Erbad, A., & Malluhi, Q. (2020). Cryptocurrencies and artificial intelligence: Challenges and opportunities. *IEEE Access*, *8*, 175840–175858. https://doi.org/10.1109/ACCESS.2020.3025211

Sait Canbaz, Y., Doğrusöz, U., Çeliksoy, M., Güngör Finance Turkcell Ödeme Hizmetleri AŞ, F., & Kurban, K. (2020). *Hydra: detecting fraud in financial transactions via graph based representation and visual analysis; Hydra: detecting fraud in financial transactions via graph based representation and visual analysis*. https://doi.org/10.1109/ISMSIT50672.2020.9255191

Singh, A., Gupta, A., Wadhwa, H., Asthana, S., & Arora, A. (2021). Temporal Debiasing using Adversarial Loss based GNN architecture for Crypto Fraud Detection. *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*, 391–396. https://doi.org/10.1109/ICMLA52953.2021.00067

Song, W., Zhang, W., Wang, J., Zhai, L., Jiang, P., Huang, S., & Li, B. (2023). *Blockchain Data Analysis from the Perspective of Complex Networks: Overview* (Vol. 28, Issue 1). https://www.webofscience.com

Sureshbhai, P. N., Bhattacharya, P., & Tanwar, S. (2020). KaRuNa: A blockchain-based sentiment analysis framework for fraud cryptocurrency schemes. *2020 IEEE International Conference on Communications Workshops, ICC Workshops 2020 - Proceedings*. https://doi.org/10.1109/ICCWORKSHOPS49005.2020.9145151

Tang, J., Lu, X., Xiang, Y., Shi, C., & Gu, J. (2023). Blockchain search engine: Its current research status and future prospect in Internet of Things network. *Future Generation Computer Systems*, *138*, 120–141. https://doi.org/10.1016/J.FUTURE.2022.08.008

Tharani, J. S., Charles, E. Y. A., Hou, Z., Palaniswami, M., & Muthukkumarasamy, V. (2021). Graph based visualisation techniques for analysis of blockchain transactions. *Proceedings - Conference on Local Computer Networks, LCN*, *2021-October*, 427–430. https://doi.org/10.1109/LCN52139.2021.9524878

Trautman, L. (2014). *Virtual Currencies Bitcoin & What Now After Liberty Reserve, Silk Road, and Mt. Gox?* (Vol. 20). http://scholarship.richmond.edu/jolt/vol20/iss4/3

Trozze, A., Kamps, J., Akartuna, E. A., Hetzel, F. J., Kleinberg, B., Davies, T., & Johnson, S. D. (2022). Cryptocurrencies and future financial crime. *Crime Science*, *11*(1), 1–35. https://doi.org/10.1186/S40163-021-00163-8/TABLES/8

Vassallo, D., Vella, V., & Ellul, J. (2021). Application of Gradient Boosting Algorithms for Anti-money Laundering in Cryptocurrencies. *SN Computer Science*, *2*, 143. https://doi.org/10.1007/s42979-021-00558-z

Weber, M., Domeniconi, G., Chen, J., Karl Weidele, D. I., Bellei Elliptic, C., Robinson Elliptic, T., Leiserson, C. E., Bellei, C., & Robinson, T. (2019). *Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics*.

Xia, F., Sun, K., Yu, S., Aziz, A., Wan, L., Pan, S., & Liu, H. (2021). Graph Learning: A Survey. *IEEE Transactions on Artificial Intelligence*, *2*(02), 109–127. https://doi.org/10.1109/TAI.2021.3076021

Yang, L., Dong, X., Xing, S., Zheng, J., Gu, X., & Song, X. (2019). An abnormal transaction detection mechanim on bitcoin. *Proceedings - 2019 International Conference on Networking and Network Applications, NaNA 2019*, 452–457. https://doi.org/10.1109/NANA.2019.00083

Yu, T., Chen, X., Xu, Z., & Xu, J. (2022). *MP-GCN: A Phishing Nodes Detection Approach via Graph Convolution Network for Ethereum*. https://doi.org/10.3390/app12147294

Yuan, Y., Member, S., & Wang, F.-Y. (2018). Blockchain and Cryptocurrencies: Model, Techniques, and Applications; Blockchain and Cryptocurrencies: Model, Techniques, and Applications. *SYSTEMS*, *48*(9). https://doi.org/10.1109/TSMC.2018.2854904

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2018). Network Representation Learning: A Survey. *IEEE Transactions on Big Data*, *6*(1), 3–28. https://doi.org/10.1109/TBDATA.2018.2850013

Zhang, Y., & Trubey, P. (2019). Machine Learning and Sampling Scheme: An Empirical Study of Money Laundering Detection. *Computational Economics*, *54*(3), 1043–1063. https://doi.org/10.1007/S10614-018-9864-Z

Zhou, H., Sun, G., Fu, S., Wang, L., Hu, J., & Gao, Y. (2021). Internet Financial Fraud Detection Based on a Distributed Big Data Approach with Node2vec. *IEEE Access*, *9*, 43378–43386. https://doi.org/10.1109/ACCESS.2021.3062467

APPENDIX I – SUPERVISED BASELINE MODELS' PARAMETERS

Table 11: Supervised Models' Parameters

Model	Parameters
RF	n_estimators=50, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_f, action_leaf=0.0, max_features=50, max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None*[1], verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
LR	penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None*[1], solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None
MLP	hidden_layer_sizes=(50,), activation='relu' , solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=500, shuffle=True, random_state=None*[1], tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000

[1] It is important to note that the default value for the random_state parameter was set as None* in all models. The asterisk (*) indicates that this is the default value, which implies that the random state was not explicitly set in the table. However, in the actual code implementation, a specific random state was generated.

APPENDIX II – GRAPH CONVOLUTIONAL NETWORK'S PARAMETERS

Table 12: Graph Convolutional Networks' Parameters

Model	Number of Input Features	Number of Output Features	Number of Hidden Layers	Learning Rate	Number of epochs	Number of Hidden Channels
GCN	166	2	2	0.001	1000	100
Skip-GCN	166	2	2	0.001	1000	100

APPENDIX III – NODE2VEC AND DEEPWALK PARAMETERS

Table 13: Node Embeddings' Parameters

Model	WalK Number	Walk Lenght	p	q	Dimensions	Number of epochs	Learning Rate
DeepWalk	10	80	-	-	128	1	0.05
Node2Vec	10	80	1.0	1.0	128	1	0.05

APPENDIX IV – GCN ARCHITECTURE FIRST APPROACH CODE

```python
class GCN(nn.Module):
    def __init__(self, in_features, out_features, bias=True):
        super(GCN, self).__init__()
        self.in_features = in_features
        self.out_features = out_features
        self.weight = nn.Parameter(torch.FloatTensor(in_features, out_features))
        if bias:
            self.bias = nn.Parameter(torch.FloatTensor(out_features))
        else:
            self.register_parameter('bias', None)
        self.reset_parameters()

    def reset_parameters(self):
        stdv = 1. / math.sqrt(self.weight.size(1))
        self.weight.data.uniform_(-stdv, stdv)
        if self.bias is not None:
            self.bias.data.uniform_(-stdv, stdv)

    def forward(self, input, adj):
        support = torch.mm(input, self.weight)
        output = torch.spmm(adj, support)
        if self.bias is not None:
            return output + self.bias
        else:
            return output
```

```python
class NormalGCNModel(nn.Module):
    def __init__(self, in_features, hidden_features, out_features, Ab):
        super(NormalGCNModel, self).__init__()
        self.gcn1 = GCN(in_features, hidden_features)
        self.gcn2 = GCN(hidden_features, out_features)

    def forward(self, x, adj):
        x = F.relu(self.gcn1(x, adj))
        x = self.gcn2(x, adj)
        return x
```

```python
class SkipGCNModel(nn.Module):
    def __init__(self, in_features, hidden_features, out_features, dropout=None):
        super(SkipGCNModel, self).__init__()
        self.layer_gcn = GCN(in_features, hidden_features)
        self.dropout = nn.Dropout(dropout) if dropout is not None else None
        self.layer_gcn_skip = GCN(hidden_features + in_features, out_features)

    def forward(self, x, adj, training=False):
        x1 = F.relu(self.layer_gcn(x, adj))
        if self.dropout is not None:
            x1 = self.dropout(x1) if training else x1
        x2 = torch.cat([x, x1], dim=-1)
        x2 = self.layer_gcn_skip(x2, adj)
        return x2
```

Figure 33: Graph Convolutional Network Implementation (1st approach)

APPENDIX V – GCN ARCHITECTURE REVISED APPROACH CODE

```python
class NormalGCN(torch.nn.Module):
    def __init__(self, num_node_features, hidden_channels, num_classes, dropout):
        super(NormalGCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, num_classes)

    def forward(self, data, training=False, return_embeddings=False):
        x = self.conv1(data.x, data.edge_index)
        if return_embeddings:
            return x
        x = x.relu()
        x = self.dropout(x) if training else x
        x = self.conv2(x, data.edge_index)
        return x

```

```python
class SkipGCN(torch.nn.Module):
    def __init__(self, num_node_features, hidden_channels, num_classes, dropout):
        super(SkipGCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, hidden_channels)
        self.conv2 = GCNConv(hidden_channels + num_node_features, num_classes)
        self.weight = nn.init.xavier_normal_(Parameter(torch.Tensor(num_node_features, num_classes)))

    def forward(self, data, training=False, return_embeddings=False):
        x = self.conv1(data.x, data.edge_index)
        if return_embeddings:
            return x
        x1 = x.relu()
        x1 = self.dropout(x1) if training else x1
        x2 = torch.cat([data.x, x1], dim=-1)
        x2 = self.conv2(x2, data.edge_index)
        x2 = x2 + torch.matmul(data.x, self.weight)
        return x2
```

Figure 34: Graph Convolutional Network Implementation (Revised Approach)