Fraud Detection in Cryptocurrency Transaction Networks



Adriano Romário Correia Chessa
Master in Data Science
Department of Computer Science
Faculty of Sciences of University of Porto
2026

Fraud Detection in Cryptocurrency Transaction Networks

Adriano Romário Correia Chessa

Dissertation carried out as part of the Master in Data Science

Department of Computer Science
2026

Supervisor
Rita P. Ribeiro, Assistant Professor, Faculty of Sciences,
University of Porto

Co-supervisor
Pedro Ribeiro, Associate Professor, Faculty of Sciences,
University of Porto

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Rita P. Ribeiro and Dr. Pedro Ribeiro, for their continued guidance, support, and patience throughout the development of this work. I am also very grateful to my family for their encouragement and unwavering support throughout this journey, and especially to Rita Biltes for her continuous support. I would like to thank my friends for their encouragement and support throughout this endeavor, especially Pedro Pedrosa and finally, I would like to express my sincere appreciation to Arie Dia Fauzan and Krzysztof Sygut, for their understanding and for granting me the time and flexibility needed to complete this dissertation.

Abstract

Fraud detection in decentralized finance (DeFi) remains a critical challenge due to the highly imbalanced, heterogeneous, and relational nature of Ethereum transaction data. This work proposes a graph-based fraud detection framework that models Ethereum transactions as directed bipartite heterogeneous graphs, where wallet and transaction nodes are connected by directed edges representing the flow of funds. To preserve temporal integrity, the BCCC-DeFiFraudTrans-2025[3] dataset is partitioned chronologically, with transactions spanning 2017-2020 used for model development, and the period 2021-2022 held exclusively for final evaluation. Four models are evaluated and compared, XGBoost was used as a baseline and compared to XGBoost with SMOTE and two graph-based approaches, graphSAGE, and Embed-SMOTE (a GraphSMOTE variant in which synthetic nodes are generated directly in the embedding space to address class imbalance). Model selection follows an expanding-window cross-validation design. Validation AUC-PR is used as the primary selection and early-stopping metric, with classification thresholds tuned post-training by maximising F1-score over the precision-recall curve on the validation set. Experimental results demonstrate that the graph-based approaches capture relational patterns in transaction behaviour that are not accessible to tabular baselines, offering improved detection of fraudulent activity. However, Embed-SMOTE did not outperform the standard GraphSAGE model, suggesting that synthetic node generation in the embedding space may not provide sufficient additional discriminative signal for this task.

Keywords: Ethereum, class imbalance, graph neural networks, bipartite graphs, anomaly detection

Resumo

A deteção de fraude em finanças descentralizadas (DeFi) continua a ser um desafio devido à natureza desequilibrada, heterogénea e relacional dos dados de transações Ethereum. Este trabalho propõe uma framework de deteção de fraude baseada em grafos que modela as transações Ethereum como grafos heterogéneos bipartidos dirigidos, onde nós de carteiras e de transações estão ligados por arestas dirigidas que representam o fluxo de fundos. Para preservar a integridade temporal, o conjunto de dados BCCC-DeFiFraudTrans-2025[3] é dividido cronologicamente, com as transações do período 2017-2020 utilizadas para o desenvolvimento dos modelos e o período 2021-2022 reservado exclusivamente para a avaliação final. Quatro modelos são avaliados e comparados, XGBoost como modelo de referência tabular, XGBoost com sobreamostragem SMOTE e dois modelos baseados em grafos, o GraphSAGE, e Embed-SMOTE (uma variante do GraphSMOTE em que nós sintéticos são gerados diretamente no espaço de embeddings para lidar com o desequilíbrio entre classes). A AUC-PR da validação é utilizada como métrica primária de seleção, com os limiares de classificação ajustados após o treino através da maximização do F1 sobre a curva de precision-recall no conjunto de validação. Os resultados experimentais demonstram que as abordagens baseadas em grafos capturam padrões relacionais no comportamento das transações que não são acessíveis a modelos tabulares, possibilitando uma deteção melhorada de atividade fraudulenta. No entanto, o Embed-SMOTE não superou o modelo GraphSAGE, sugerindo que a geração de nós sintéticos no espaço de embeddings poderá não fornecer sinal discriminativo adicional suficiente para esta tarefa.

Palavras-chave: Ethereum, desequilíbrio de classes, redes neuronais para grafos, grafos bipartidos, deteção de anomalias

Contents

Acknowledgements i

Abstract iii

Resumo v

Contents ix

List of Tables xi

List of Figures xiii

Acronyms xv

1 Introduction 1

1.1 Motivation and Context 1

1.2 Problem Definition and Research Goal 2

1.3 Dissertation Outline 3

2 Background and Literature Review 5

2.1 The Scale and Growth of Cryptocurrency Crime 5

2.2 Blockchain and Ethereum Background 7

2.3 Classical Fraud Detection 7

2.4 Graph-Based Fraud Detection 9

2.5 Graph Neural Networks 9

2.6 Class Imbalance in Graph Learning . . . 11
2.7 Evaluation in Imbalanced Settings . . . 12
2.8 Limitations and Gaps . . . 13

3 Case Study on Ethereum transactions **15**
3.1 Dataset Description . . . 15
3.2 Data Quality and Preprocessing . . . 17
3.3 Exploratory Data Analysis (EDA) . . . 19
3.3.1 Class-conditional distributions . . . 22
3.3.2 Correlation matrix . . . 26

4 Methodology **29**
4.1 Pipeline Overview . . . 29
4.1.1 Adaptations from Literature Review . . . 30
4.1.2 Cross-validation fold definition . . . 30
4.2 Bipartite Graph Construction . . . 31
4.2.1 Wallet Node Features . . . 33
4.2.2 Transaction (Tx) Node Features . . . 33
4.3 Baseline Model: eXtreme Gradient Boosting (XGBoost) . . . 34
4.4 Graph SAmple and aggreGatE (GraphSAGE) for Heterogeneous Graphs . . . 35
4.5 Addressing Class Imbalance: Embed-SMOTE . . . 36
4.6 Random Seed Strategy and Reproducibility . . . 38
4.7 Computational Resources and Software Environment . . . 38

5 Results **39**
5.1 Statistical test selection . . . 39
5.2 Models results . . . 40
5.3 Comparison with prior work . . . 42
5.4 Discussion . . . 42

6 Conclusions **45**

6.1 Main findings 45

6.2 Contributions 46

6.3 Limitations and future work 46

Bibliography **49**

A Appendix **55**

List of Tables

4.1 Fold-level graph statistics 34

5.1 Cross-validation performance for all models 41

5.2 Final test performance for all models 42

A.1 Dataset transaction features 55

A.2 Dataset Wallet feature 56

List of Figures

2.1 Crypto fraud losses and complaints (2017-2025) 6

2.2 GraphSAGE neighbourhood sampling 10

2.3 Graph classification before and after synthetic nodes generation 12

3.1 Ether transactions sample window 16

3.2 Wallet tag distribution 17

3.3 Flag distribution by year 20

3.4 Transaction frequency heatmaps 21

3.5 Violin plots of transaction features related to gas costs 23

3.6 Violin plots of transaction features related to errors 25

3.7 Violin plots of transaction features related to gas usage 26

3.8 Pearson correlation matrix 27

3.9 Correlation network of features 28

4.1 End-to-end experimental pipeline 30

4.2 Expanding-Window Cross-Validation protocol 31

4.3 Bipartite graph schema 32

5.1 Ranking metrics across validation runs 40

5.2 Classification metrics across validation runs 41

Acronyms

AUC-PR Area Under the Precision-Recall Curve

AUC-ROC Area Under the Receiver Operating Characteristic Curve

API Application Programming Interface

DL Deep Learning

DeFi Decentralised Finance

DAOs Decentralised Autonomous Organisations

EDA Exploratory Data Analysis

ETH Ether

EVM Ethereum Virtual Machine

EW-CV Expanding-Window Cross-Validation

Embed-SMOTE Embedding-Space SMOTE

GraphSAGE Graph SAmple and aggreGatE

GNN Graph Neural Network

IQR Interquartile Range

IC3 Internet Crime Complaint Center

ICO Initial Coin Offering

LR Logistic Regression

LSTM Long Short-Term Memory

ML Machine Learning

MLP Multi-Layer Perceptron

PoW Proof of Work

PoS Proof of Stake

ReLU Rectified Linear Unit

RF Random Forest

SMOTE Synthetic Minority Over-sampling Technique

SVM Support Vector machines

Tx Transaction

XGBoost eXtreme Gradient Boosting

Chapter 1

Introduction

This chapter introduces the dissertation by presenting the motivation and context that led to the study, with particular attention to the growing relevance of cryptocurrency fraud, its economic impact and the need for more effective analytical approaches. This chapter also defines the research problem addressed in this work and concludes by outlining the structure of the dissertation to serve as a guide for the subsequent chapters.

1.1 Motivation and Context

The emergence of decentralized digital currencies such as Bitcoin and Ether (ETH) has changed the architecture of the traditional financial system, which historically relied on trusted third-party intermediaries such as central banks and payment processors to validate and settle Transaction (Tx)s [1]. By replacing these intermediaries with cryptographic protocols and distributed consensus mechanisms, blockchain technology introduced an alternative model for traditional payments, one in which value can be transferred across borders in a short period of time and without requiring the involvement of a central authority [2].

Global cryptocurrency adoption continued to expand in 2025, reaching more than 740 million individuals in 151 countries, with the United States of America and Europe dominating the on-chain Tx volumes and receiving respectively $2.2 trillion and $2.6 trillion over the last year [3, 4]. Beyond simple currency transfer, blockchain networks, such as Ethereum, have enabled a broader ecosystem of decentralized applications including Decentralised Finance (DeFi), gaming, and digital infrastructure. These use cases position cryptocurrency not only as an alternative payment method or store of value, but as a foundation for new classes of programmable financial instruments [5].

Unfortunately, the same properties that make blockchain technology an attractive alternative also create conditions that are advantageous to malicious users. As adoption has scaled, so has the sophistication and volume of fraud, raising urgent questions about how detection and prevention can keep pace. Addressing these questions is not only technically relevant but has

also direct social importance, both to protect the assets of retail users and to strengthen trust in this payment alternative [6].

1.2 Problem Definition and Research Goal

Detecting fraudulent activity on the Ethereum blockchain remains a challenging problem. The pseudonymity of participant identities, the diversity of fraud typologies, and the severe class imbalance inherent to fraud datasets complicate the development of effective and generalizable fraud detection systems. Existing approaches have demonstrated promising results in controlled settings, but leave room for improvement in terms of performance with imbalanced data, the impact of graph representation choices, and comparative evaluation of oversampling strategies in graph-based learning [7, 8].

This project aims to address these limitations by modelling Ethereum Tx data as a graph and applying methods based on Graph Neural Network (GNN) to identify fraudulent Txs within an imbalanced dataset. Specifically, the work investigates the Graph SAmple and aggreGatE (GraphSAGE) architecture [8] and an extension incorporating Embedding-Space SMOTE (Embed-SMOTE) [9, 10] to mitigate class imbalance, benchmarking both against a tabular baseline to quantify the structural advantage granted by graph-based modelling. The project further examines the impact of different modeling choices and evaluates strategies to handle class imbalance at both the data and the architecture level. The findings are intended to contribute to the growing body of literature on GNN-based fraud detection in cryptocurrency networks and to provide empirical evidence for the practical applicability of graph-based approaches to real-world blockchain security challenges.

To guide this work and achieve the proposed objectives, one main research question and two more specific sub-questions were formulated. These questions are investigated using the Ethereum BCCC-DeFiFraudTrans-2025[3] dataset as a case study, and therefore the scope of the study is limited to this dataset rather than to Ethereum fraud classification in general.

RQ 1: Does incorporating the relational context of Txs through GNNs improve fraud detection compared with classical tabular Machine Learning (ML) models?

RQ 1.1: To what extent do GNNs outperform tabular baselines in detecting fraud under severe class imbalance?

RQ 1.2: Which imbalance-handling strategy return the best performance for detecting fraud in these Txs?

1.3 Dissertation Outline

The remainder of this document is organized as follows:

Chapter 2 Reviews the literature on blockchain fraud, classical and graph-based fraud detection methods, class imbalance and identifies the main research gaps in the current state-of-the-art.

Chapter 3 Describes the BCCC-DeFiFraudTrans-2025[3] dataset, the preprocessing pipeline and the exploratory data analysis.

Chapter 4 Details the experimental methodology, including the temporal cross-validation design, the construction of the bipartite Ethereum Tx graphs, the eXtreme Gradient Boosting (XGBoost) baseline, the GraphSAGE model, the Embed-SMOTE extension, and the evaluation setup.

Chapter 5 Presentation and analysis of the experimental results, comparing model performance across tabular and graph-based approaches, as well as across different imbalance-handling strategies.

Chapter 6 Summarises the principal findings, answers the research questions, discusses the limitations of the work, and identifies directions for future research.

Appendix A Provides a complete description of the features of the dataset used in this study.

Chapter 2

Background and Literature Review

This chapter outlines the strategy used to review the existing literature on the subject and identifies the main limitations and challenges in the field. It begins by introducing the context of cryptocurrency crime and the fundamentals of blockchain technology, before reviewing classical and graph-based fraud detection methods, Graph Neural Network (GNN)s, class imbalance, and the main gaps that motivate this dissertation. The literature review was conducted as a structured, systematic review using Google Scholar, Connected Papers[1], and Litmaps[2] to identify and expand relevant publications. These tools were used to identify foundational studies, explore citation networks, and identify related work in blockchain fraud detection, graph-based learning, and class imbalance handling. The search and screening were guided by relevance to the research questions, methodological similarity, and proximity to Ethereum and blockchain fraud detection, with priority given to peer-reviewed sources and widely cited contributions.

2.1 The Scale and Growth of Cryptocurrency Crime

The decentralized and pseudonymous nature of public addresses, the irreversibility of confirmed Transaction (Tx)s, the absence of jurisdictional boundaries, and the general lack of regulatory oversight collectively create conditions that are structurally advantageous to fraudsters [3]. Once funds are transferred on-chain, there is no central authority capable of reversing Txs or freezing the receiving address, tracking funds across borders requires coordination between jurisdictions that may have incompatible legal frameworks or limited technical capacity.

The scale of cryptocurrency-related crime has grown drastically in recent years. According to the FBI Internet Crime Complaint Center (IC3), losses attributed to cryptocurrency fraud reached $9.3 billion in 2024, representing a 66% increase over the previous year, with investment fraud accounting for the majority of reported losses [6]. As shown in **Figure 2.1**, the amount of loses increased in 2025 to a record $11,366 billion across 181,565 complaints, representing a 22% increase over the previous year [6]. These reported volumes can be considered conservative,

[1] https://connectedpapers.com/ [2] https://litmaps.com/

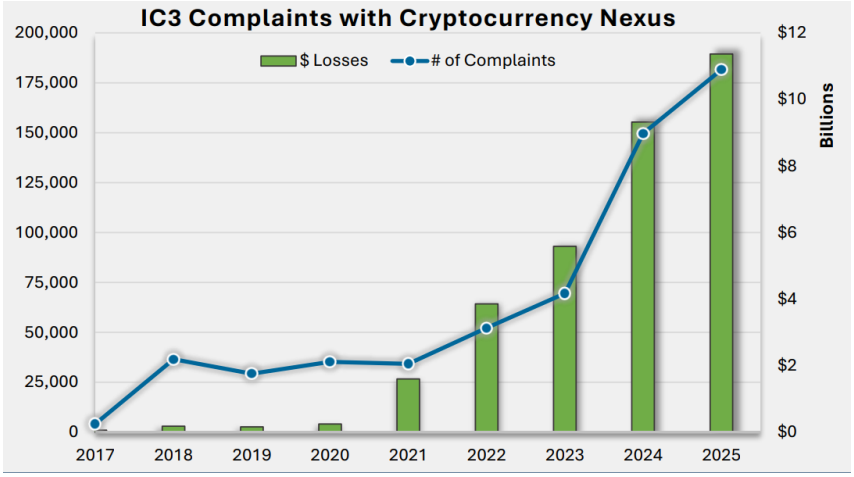


Figure 2.1: Comparison of total financial losses and volume of complaints related to cryptocurrency fraud (2017-2025). Adapted from FBI IC3 [6].

as a substantial proportion of victims do not report incidents to law enforcement due to stigma, lack of awareness, or limited confidence that reports will yield results or recover lost funds.

The diversity of fraud typologies observed on blockchain networks further complicates detection efforts. The most prevalent forms of fraud include:

Phishing attacks and address poisoning: Fraudsters impersonate legitimate entities such as exchanges or project teams to deceive retail users into revealing private credentials or approving malicious Txs. Phishing is among the most cost-effective and scalable attack types, requiring minimal technical infrastructure relative to the potential return. Address poisoning is a related technique in which attackers send low-value Txs from addresses visually resembling a victim's known contacts, exploiting copy-paste behavior to redirect future transfers to attacker-controlled wallets [11].

Ponzi schemes: These schemes promise sustained and abnormally high returns on investment, funding early investor payouts through the capital of later participants rather than legitimate profits. Collapse is inevitable once new investment flows decreases. The programmability of Ethereum smart contracts has made the deployment of on-chain Ponzi schemes particularly accessible, as fraudulent contracts can be deployed at minimal cost and dissolve automatically once conditions are met [12].

Initial Coin Offering (ICO) fraud: ICOs emerged as a mechanism to jumpstart capital for new blockchain projects by issuing tokens to early investors. They have been routinely exploited, with projects either nonexistent or abandoned after fundraising [13].

Exit scams and rug pulls: Project developers cultivate the appearance of a legitimate investment over an extended period before abruptly draining liquidity pools or treasuries and disappearing. The sustained legitimacy of the project prior to the exit event makes prospective detection particularly difficult [14].

2.2 Blockchain and Ethereum Background

A blockchain is a distributed ledger technology that records Txs across a network of computers, linking data in sequential blocks through cryptographic hashes. Each block contains a reference to the hash of its predecessor, forming a chain such that retroactively altering any record would require recomputing all subsequent blocks and overcoming the consensus mechanism of the network, which makes tampering computationally infeasible in practice [15]. This property of practical immutability is central to the trust model of public blockchains, as participants need trust only in the protocol itself, rather than in any individual node.

Ethereum specifically, is a decentralized blockchain platform first proposed in 2013 by Vitalik Buterin [2] and later formalized in its technical specification [16]. Unlike Bitcoin, which was designed primarily as a peer-to-peer digital currency, Ethereum operates as a general-purpose programmable blockchain, hosting a global virtual machine known as Ethereum Virtual Machine (EVM), capable of executing arbitrary code in the form of smart contracts [5]. Smart contracts are self-executing programs stored on-chain whose logic is enforced deterministically by every participating node, without the need for a trusted intermediary. This programmability distinguishes Ethereum from earlier networks and underlies its adoption in Decentralised Finance (DeFi), token standards, and Decentralised Autonomous Organisations (DAOs). Unfortunately, this programmability has also been particularly exploited for fraud. The ease and low cost of deploying smart contracts allows malicious actors to launch, operate and abandon schemes at scale, contributing to a disproportionate concentration of Ponzi scheme and exit scam activity on the Ethereum network compared to simpler blockchains [12].

Ether (ETH), the native token of the network, functions as a utility token used to compensate validators for computational resources consumed when performing operations, rather than serving solely as a monetary instrument. The computational cost on Ethereum is denominated in *gas*, a unit that represents the complexity of individual operations and helps ensure that the network remains resistant to denial-of-service attacks.

The validation of Txs was originally achieved by Proof of Work (PoW), in which miners competed to solve computationally intensive cryptographic puzzles to earn the right to append new blocks. In September 2022, the network transitioned to Proof of Stake (PoS), where validators are probabilistically selected in proportion to their staked ETH [17]. This transition reduced Ethereum's energy consumption by an estimated 99.95% while preserving the network's security guaranties, and is widely considered one of the most significant technical milestones in the history of public blockchain networks[17].

2.3 Classical Fraud Detection

Conventional fraud detection methods used in centralized financial systems typically rely on user-identifiable attributes such as account-holder information, verified Tx histories, and central

records accessible to investigators. However, public blockchains impose different constraints for fraud detection. User identities are hidden behind pseudonymous cryptographic addresses, which provide no direct link to real-world identity without off-chain information, and the data available on-chain is relatively sparse, usually restricted to Tx value, timestamp, gas price, and gas used.

Classical methods remain important in fraud detection as they are relatively simple to implement, computationally efficient, and often easier to interpret than more complex models. For this reason, they are commonly used as baseline methods before moving on to more expressive approaches. In addition, they continue to be widely adopted in financial applications where transparency and implementation cost remain important practical considerations. Before the rise of graph-based approaches, fraud detection was largely dominated by rule-based systems, which were costly to maintain and often insufficiently adaptable to evolving fraud patterns. The introduction of traditional Machine Learning (ML) methods increased flexibility and enabled models to learn from historical examples rather than relying solely on manually defined rules. Later, Deep Learning (DL) methods extended this trend by enabling the use of richer data representations and more complex decision boundaries [18]. These methods continue to occupy an important place in the literature as they provide practical solutions for many fraud detection settings and serve as benchmarks for more advanced models.

Among classical methods, Logistic Regression (LR), Random Forest (RF), Support Vector machines (SVM), and gradient boosting algorithms have been especially common in fraud detection. These models support the use of richer feature sets and can learn non-linear relationships that are not easily captured by rule-based systems [19]. These methods also offer a useful balance between predictive performance, computational cost, and interpretability. In the context of blockchain fraud detection, recent studies show that methods such as eXtreme Gradient Boosting (XGBoost) and RF are frequently used as references [20–22]. These models typically train and infer quickly, require less data than many DL approaches, and can be deployed relatively easily in production environments. However, they still depend heavily on feature engineering, which limits their ability to capture complex interactions between entities. They are also sensitive to class imbalance and therefore, often require dedicated preprocessing or resampling strategies before training.

XGBoost, in particular, is a ML model based on gradient boosting that is widely used in classification and regression tasks. Its main advantage lies in the iterative combination of decision trees, where each tree corrects the errors of the previous, producing a robust final model with strong predictive performance and reduced risk of overfitting [23]. Its effectiveness comes from an objective function that combines the prediction error with a regularization term, allowing the model to jointly optimize accuracy and complexity [24]. In fraud detection problems, XGBoost is often used as a baseline because it handles tabular data effectively and can produce strong results even with moderately imbalanced datasets [25]. It also offers a degree of interpretability through feature importance measures, which makes it attractive for financial applications [26].

More recently, DL models such as autoencoders, convolutional neural networks, and recur-

rent neural networks have also been applied to fraud detection, especially in tasks involving sequential patterns or more complex data representations. These models can learn feature representations automatically, identify non-linear relationships, and exploit information that may be difficult to encode manually. For this reason, they are particularly useful when multiple sources of information are available or when the underlying fraud patterns are too complex for simpler models to capture. At the same time, DL methods are often difficult to interpret, require more computational resources, and may depend on substantial amounts of clean data to achieve strong performance [27].

2.4 Graph-Based Fraud Detection

Fraud on public blockchains is often relational in nature, which means that the behavioral signatures of fraudsters emerge not only from individual Tx attributes, but also from the structural patterns formed by the flow of funds between multiple addresses and contracts over time [28]. As a result, tabular ML models such as gradient-boosted trees are not naturally suited to capture these structural dependencies, since they treat each observation independently and ignore the network context.

Graph-based fraud detection emerged as a natural extension in domains where interactions between entities are central to the task [29]. Rather than treating each observation as isolated, graph-based methods represent relations between entities, Txs, and other network elements, enabling contextual analysis of observed behavior. In blockchain settings, this is particularly useful because the data are inherently relational [19, 30]. In this context, a graph may contain nodes representing wallets or Txs, each with attributes such as account type, Tx history, wallet-owner information, timestamps or other metadata relevant for fraud detection. Edges can represent Txs and may carry attributes such as amount transferred, while the direction of each edge reflects the flow of funds.

The literature suggests that graph-based methods are particularly effective when fraud is characterized not only by the properties of a single Tx or account, but also by how entities interact with each other [31]. This includes Tx chains, the local neighborhood structure, and connectivity patterns that may be difficult to detect in tabular representations.

2.5 Graph Neural Networks

GNN are a family of DL models designed to operate directly on graph-structured data. They learn node representations by iteratively aggregating information from local neighborhoods through a message-passing mechanism [32, 33]. At each layer, a node representation is updated by collecting and transforming the representations of its neighbors, allowing information to propagate through the graph. This enables GNNs to capture relational patterns that are not visible to methods operating on flat feature vectors. In the context of Ethereum fraud detection,

the blockchain Tx history can be naturally mapped to a graph in which wallet addresses correspond to nodes and Txs correspond to directed edges, with node and edge attributes derived from on-chain data. This representation allows models based on GNN to take advantage of the full relational context of each address.

Within the broader fraud detection literature, several GNN architectures have been explored, including Graph Attention Networks [34], Graph Convolutional Networks [35], and heterogeneous variants such as Relational Graph Convolutional Networks [35], which are particularly useful when the graph contains multiple nodes or edge types. A study by Kanezashi *et al.* [30] compared multiple GNNs applied to Ethereum Txs and found that heterogeneous GNNs outperformed homogeneous GNNs, suggesting that a richer graph structure can improve fraud detection performance when the underlying Tx network contains diverse relational information.

Graph SAmple and aggreGatE (GraphSAGE), was developed as an inductive GNN architecture capable of generating embeddings for unseen nodes by leveraging neighborhood information, as illustrated in **Figure 2.2**. This makes GraphSAGE particularly suitable for dynamic or evolving graphs, where new nodes appear over time and the model must generalise beyond the nodes seen during training.

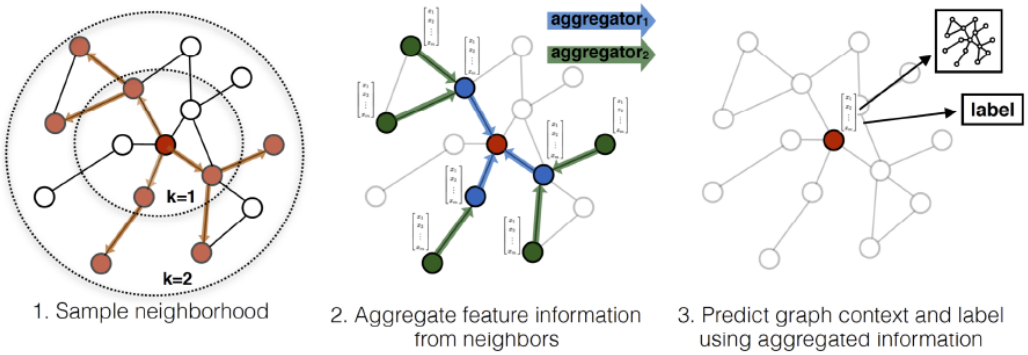


Figure 2.2: Illustration of the neighbourhood sampling and aggregation process implemented by GraphSAGE. Adapted from Hamilton *et al.* [8].

In layer k, the representation of a node v is updated using its previous representation $h_v^{(k-1)}$ and the aggregated information from its neighborhood $\mathcal{N}(v)$. Formally, this can be expressed as:

$$h_{\mathcal{N}(v)}^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\right\}\right),$$

followed by the update:

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \left[h_v^{(k-1)} \,\|\, h_{\mathcal{N}(v)}^{(k)}\right]\right),$$

where $\|$ denotes concatenation, $W^{(k)}$ is the learnable weight matrix in layer k, and σ is a non-linear activation function. The aggregation function can take different forms, such as mean pooling, max pooling, or Long Short-Term Memory (LSTM)-based aggregation, depending on the variant used [8]. This approach allows the model to capture structural patterns and relevant neighborhood context without requiring fixed embeddings for all nodes in advance, making it particularly suitable for large graphs and dynamic graphs with unseen nodes [8].

Although the literature demonstrates the strong potential of GNNs for fraud detection, it is important to note their limitations. Generalization across datasets, robustness to noisy or inconsistent labels, and sensitivity to class imbalance are still open challenges that require further research. The study of dynamic graphs, in which nodes and edges are continuously added over time, is another area where the literature remains comparatively limited [19]. As such graphs are highly relevant in online and financial contexts, the development of GNN models better suited to these settings would be valuable [19].

2.6 Class Imbalance in Graph Learning

Imbalanced datasets are one of the main challenges in fraud detection, since the number of legitimate financial Txs typically exceeds the number of fraudulent ones. This imbalance can dilute the minority class and make it difficult for models to learn discriminative representations. The problem is especially relevant in graph-based settings, where fraudulent nodes could be embedded within neighborhoods dominated by legitimate activity, further complicating the learning process [36].

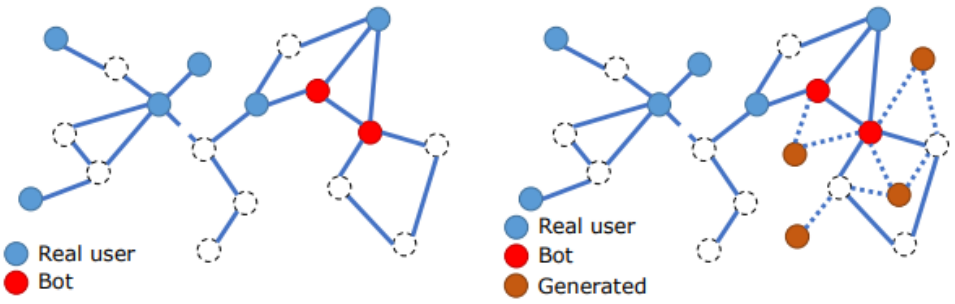
Classical approaches to class imbalance handling include resampling, class weighting, and decision-threshold adjustment. However, in relational data these strategies are not always sufficient because they may ignore structural dependencies between nodes and edges. In response to this limitation, specialized methods have emerged for imbalanced graph learning, including modifications to message passing, structural balancing strategies, and loss functions adapted to the minority class. Despite these developments, the area remains active and several open questions remain, especially in real-world applications such as blockchain fraud detection [36, 37].

GraphSMOTE is a framework designed to address imbalanced node classification in GNNs. It constructs an intermediate embedding space and performs oversampling in that space, generating synthetic representations for minority-class nodes [9] (**Figure 2.3**). In essence, GraphSMOTE extends the Synthetic Minority Over-sampling Technique (SMOTE) idea to relational data. Rather than generating new instances directly in the original feature space, the method first learns latent representations of nodes using a GNN, typically GraphSAGE, to obtain an embedding space in which nodes of the same class are closer together and the graph structure is preserved. Synthetic minority nodes are then generated by interpolating between a node and its nearest neighbor from the same class, which produces samples that are more plausible than

simple duplication. Given a minority-class node v and its nearest neighbor $nn(v)$, the synthetic embedding can be defined as:

$$h_{v'} = (1 - \lambda)h_v + \lambda h_{nn(v)}, \qquad \lambda \in [0, 1].$$

Since these synthetic nodes do not originally have graph connections, the method also includes an edge generator, that predicts links between the synthetic nodes and the remaining nodes in the graph. The resulting augmented graph is then used to train the final classifier, with the aim of improving the recognition of the minority-class while preserving the topological structure of the network [9].



(a) Bot detection task (b) After over-sampling

Figure 2.3: Representation of the graph classification task before and after the generation of synthetic nodes to balance the minority class. Adapted from Zhao *et al.* [9].

2.7 Evaluation in Imbalanced Settings

In fraud detection, commonly used evaluation metrics include precision, recall, F1-score, Area Under the Receiver Operating Characteristic Curve (AUC-ROC), and Area Under the Precision-Recall Curve (AUC-PR). Among these, recall is particularly important as the primary goal of a fraud detection system is often to minimize false negatives. Failing to identify fraudulent activity can lead to financial loss, security breaches, or further abuse of the system [38]. Although false positives are inconvenient and may increase the workload of analysts or users, they are generally preferable to missed fraud, since flagging legitimate activity is often less harmful than allowing malicious behavior to go undetected.

AUC-PR is generally more informative than AUC-ROC in imbalanced settings as it directly reflects the trade-off between precision and recall and is more sensitive to performance of the minority-class [39]. By contrast, AUC-ROC may remain high even when fraud detection is low, due to the dominant influence of the majority class in the false positive rate. Accuracy, although easy to interpret, requires careful scrutiny in imbalanced datasets, since high accuracy may simply reflect correct classification of the majority class. In blockchain fraud detection, this limitation is particularly important as the operational cost of false negatives is often much higher

than that of false positives. For these reasons, accuracy should be treated as a supplementary measure rather than the primary optimization target in imbalanced fraud detection problems [39].

2.8 Limitations and Gaps

Although the literature has advanced considerably, several gaps are still present. First, many Ethereum fraud studies still rely on a narrow range of datasets, which reduces external validity and makes comparison between papers difficult. Classical ML baselines remain important, but cannot fully exploit the relational structure of blockchain data, limiting their effectiveness for coordinated fraud. GNN studies have shown strong promise, yet label imbalance and sparse fraudulent neighborhoods continue to weaken performance, especially when the minority class is highly underrepresented.

Another gap is that the literature often evaluates models in idealized settings rather than addressing the practical difficulties of Ethereum fraud detection, such as scarce labels, changing fraud behavior, and the need for robust minority-class recall. These limitations suggest that there is room for a GNN-based method that is explicitly designed for imbalanced Ethereum fraud classification and evaluated using metrics that better reflect rare-event detection. This dissertation aims to contribute to the reduction of that gap by studying a graph-based approach that combines Ethereum Tx structure with imbalance-aware learning.

Chapter 3

Case Study on Ethereum transactions

This chapter describes the Ether (ETH) Transaction (Tx) dataset used in this dissertation, including its source, time range, and label distribution, and presents an exploratory analysis of the data to better understand its structure, class imbalance, temporal patterns, and feature relationships. Together, these sections motivate the subsequent graph-based modeling choices and provide the empirical basis for the preprocessing and graph construction steps described in the following chapters.

3.1 Dataset Description

The used BCCC-DeFiFraudTrans-2025 dataset was provided by York University[3], which was introduced by Lashkari *et al.* [40] and the description of features of the dataset can be consulted in detail in **Appendix A**. This dataset comprises 1,048,021 annotated ETH Txs, of which 88,793 (8.5%) are labeled as fraudulent and 959,228 (91.5%) as legitimate. At the wallet level, the dataset contains 4,324 distinct addresses, of which 1,762 are involved in fraudulent Txs (40.7%) and 2,562 are not associated with fraud (59.3%). The Txs span the period from 2015 to 2022, covering the early adoption and subsequent growth of the ETH blockchain, as shown in **Figure 3.1**.

The data was collected from Etherscan[4], a widely used blockchain explorer that provides transparent access to ETH Tx data. In this dataset, each Tx is associated with 79 features extracted by the DeFiTransLyzer V1.0 analyzer and grouped into wallet-level and Tx-level descriptors [40].

Wallet-level features consist of aggregated statistics such as mean, median, standard deviation, variance, skewness, minimum, maximum, range, coefficient of variation, and mode computed over properties such as gas, transferred value, nonce, error counts, unique interacted wallets, and the total number of Tx associated with a wallet. Tx-level features include gas usage and prices, block-related attributes, log and event characteristics, status and error information,

[3] https://yorku.ca/research/bccc/ucs-technical/cybersecurity-datasets-cds/

[4] https://etherscan.io/

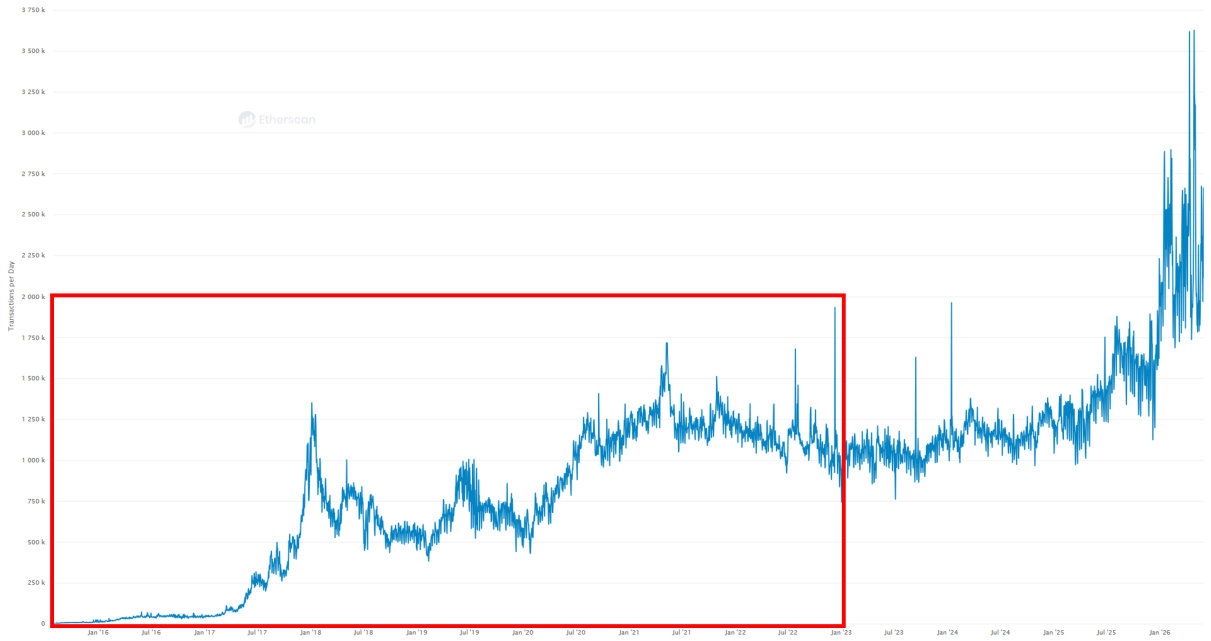


Figure 3.1: ETH Txs from 2015 to 2026. The red square indicates the sampling window used in this dissertation. Adapted from Etherscan[4].

token transfer activity, and structural indicators such as contract creation. The analyzer also flattens the nested logs and computes summary statistics to enrich the contextual information available for each Tx.

The original authors collected the Tx labels from Etherscan[4] and subsequently refined them through anomaly detection, consistency checks, and duplicate removal to improve reliability [40]. To better understand the type of fraud represented in the dataset, an Etherscan[4] Application Programming Interface (API) key was used to retrieve additional wallet information and complement the original data. This process provided the sender and receiver wallet addresses, the Tx timestamp, and the tag assigned to each wallet by Etherscan[4]. After an analysis of these tags it was observed that the dominant wallet label was “Phish / Hack”, which appeared 1,735 times and accounted for 99.03% of all tagged cases. The second most frequent tag was “Ponzi”, representing 0.514% of cases, while the remaining tag distribution is shown in **Figure 3.2**. This suggests that the dataset is mainly composed of phishing or hacking-related fraud, including addresses that deceive users into sending funds, approve malicious Txs, or receive stolen funds through compromised credentials or scam contracts. Since the main goal of this dissertation is to use graphs to identify fraud patterns, collecting `to` and `from` wallet addresses and `timestamp` was essential for constructing a comprehensive and chronologically ordered graph representation.

In the Ethereum blockchain, the `nonce` feature is a progressively increasing Tx counter assigned to each wallet and is increased for each outgoing Tx to enforce ordering. Abnormal nonce patterns, such as very low values indicating fresh accounts or very high values indicating automated senders, can be informative indicators of fraudulent activity.

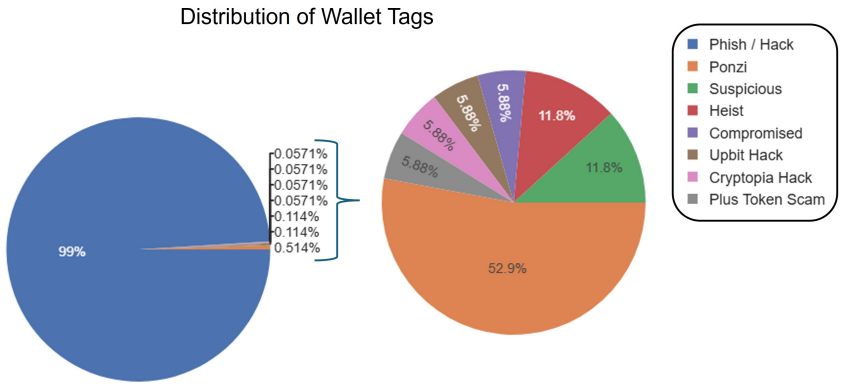


Figure 3.2: Distribution of tags associated with wallets involved in the ETH Txs. The left pie chart shows the overall tag distribution, while the right pie chart shows the breakdown of the remaining tags excluding “Phishing/Hack”.

3.2 Data Quality and Preprocessing

The dataset was first examined for missing values, duplicate rows, outliers, and redundant features. Timestamp features were decomposed and zero-variance features were removed to reduce redundancy and improve modeling efficiency.

The dataset did not contain duplicated rows, although some rows had several Tx-level values missing. As only 54 rows were affected, corresponding to 0.005% of the dataset, these rows were removed rather than imputed, since a reliable imputation strategy would be difficult to perform and justify. In addition, for a total of 28,649 rows, the `to_address` field could not be extracted. After further investigation on Etherscan[4], this appears to occur when the initiating address interacts with a smart contract, which the Etherscan[4] API does not expose in a directly usable form. Imputing these missing values would risk introducing an artificial structure into the graph, so the entries were excluded.

To reduce redundancy, features with zero variance were removed, as they provide no discriminative information. The following variables were excluded:

- `length_transaction_hash`
- `log_removed`
- `length_from`
- `length_to`
- `gas_per_log_event`

- log_index
- token_transfer_amount
- normalized_token_transfer
- is_contract_creation
- status
- message
- Error!

Next, perfectly or highly correlated features were identified in order to reduce multicollinearity and redundancy. A Pearson correlation threshold of $|r| \geq 0.95$ was used, and in each correlated pair, the feature with the lowest variance was removed. This threshold was selected as it allows redundant variables to be discarded while remaining as enough to preserve potentially useful information [41]. This step was particularly important because several of the wallet-level descriptors extracted by DeFiTransLyzer V1.0 [40] encode overlapping aspects of Tx behavior. As a result, the following features were removed:

- gas_prices.gas_prices_standard_deviation
- number_of_from_address
- number_of_to_address
- values.values_average
- nonce.nonce_maximum_val
- values.values_maximum_val
- cumulativeGasUsed.cumulativeGasUsed_median
- gas_prices.gas_prices_range_value
- error_rate
- cumulativeGasUsed.cumulativeGasUsed_maximum_val
- gas_used.gas_used_maximum_val
- gas_prices.gas_prices_maximum_val

After preprocessing, the shape of the dataset was reduced from the original (1048021, 96) to (1019318, 72).

The raw `timestamp (UTC)` feature was decomposed into `date`, `day`, `hour` and `weekday` so that the models could capture temporal patterns, such as increased fraudulent activity at specific times or on specific days. The distribution of these temporal variables is explored further in **Section 3.3**. To preserve temporal ordinality, a linear time index was also created by normalizing the Unix epoch timestamp, providing a continuous representation of time suitable for gradient-based models. In addition, as `hour` is cyclical (with 00 and 23 being semantically close despite being numerically distant) it was encoded using sine and cosine transformations:

$$\texttt{hour_sin} = \sin\left(\frac{2\pi \cdot \texttt{hour}}{24}\right), \quad \texttt{hour_cos} = \cos\left(\frac{2\pi \cdot \texttt{hour}}{24}\right) \tag{3.1}$$

This transformation maps each hour to the unit circle, ensuring that temporal proximity is represented correctly and preventing the model from treating 00 as an artificial discontinuity in feature space [42, 43].

3.3 Exploratory Data Analysis (EDA)

The dataset is substantially imbalanced, particularly at the Tx level, where 91.3% of Txs are labelled as legitimate and only 8.7% are marked as fraudulent, which is consistent with the imbalance typically observed in datasets of this type [44]. This imbalance motivates the use of metrics and modeling strategies that are robust to class skew.

After analyzing the timestamp distribution, it became apparent that the majority of Txs were collected in 2017, while the samples from 2015-2016 contained no fraud cases, as shown in **Figure 3.3**. This temporal concentration implies that a random split would be inappropriate as it could introduce leakage from future patterns into earlier training data. For this reason, the modeling strategy used in this dissertation follows a temporally ordered split, where older Txs are used for training and more recent Txs are reserved for validation and testing.

The heatmaps (**Figure 3.4**) of Tx counts by hour and day show that legitimate activity is spread more evenly across the month and throughout the 24-hour cycle with some concentration around American working hours (13 to 21 UTC). In contrast, fraudulent Txs occur in much smaller volumes and cluster in a few specific hour-day combinations namely day 14 between 18 and 19 UTC and day 17 at 13 UTC, producing isolated regions of high intensity against a sparse background, as can be seen in **Figure 3.4 A**. A similar pattern emerges when aggregating by weekday and hour. Legitimate Txs show relatively consistent traffic across the week, with mild peaks during typical business hours, while fraudulent Txs appear sporadic and concentrated in a few hours on specific days namely mondays at 13, 18 and 19 UTC. This suggests that fraud may occur in short bursts rather than as a continuous process, which supports the inclusion of temporal features and motivates the use of time-aware models **Figure 3.4 B**.

The distribution of numeric features was also examined using summary statistics, including counts, means, standard deviations, quartiles, and ranges. The large number of observations

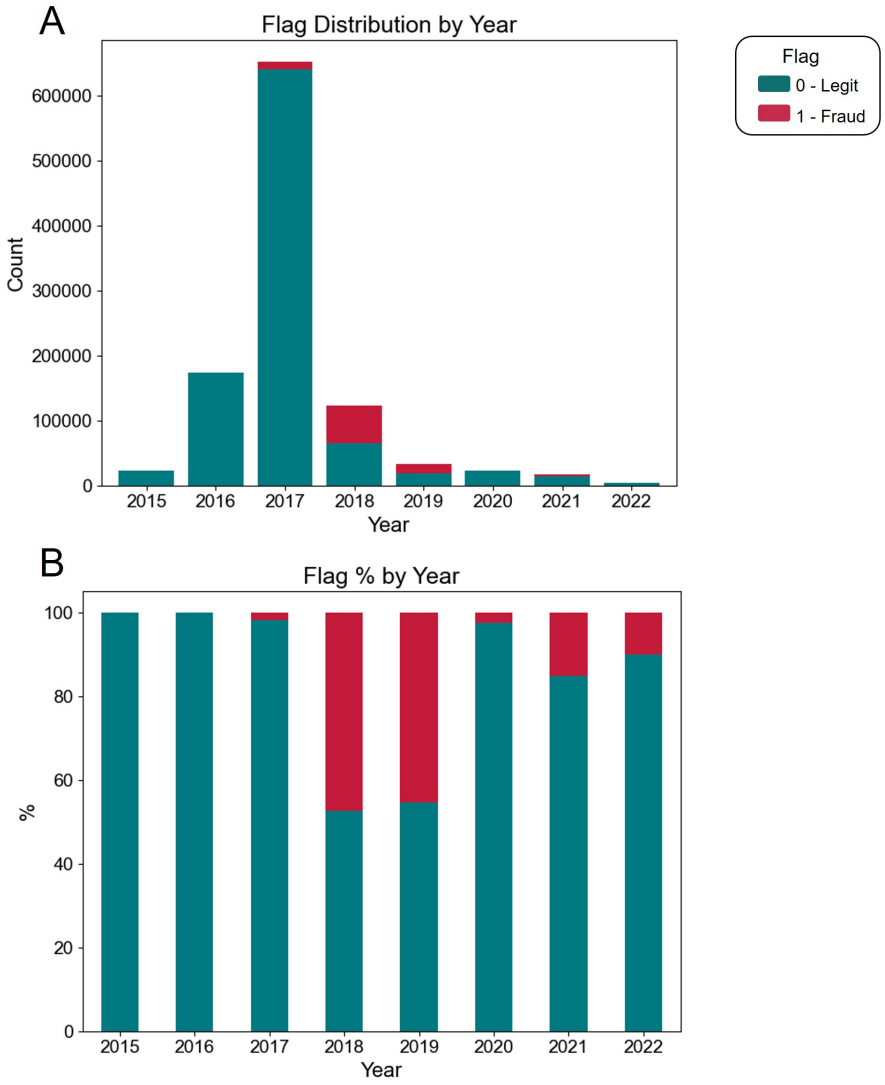


Figure 3.3: Flag distribution across the sampled years. **A** - Absolute counts of fraudulent and non-fraudulent Txs per year. **B** - Percentage composition of fraudulent and non-fraudulent Txs per year.

made it possible to assess both the distributional shape and tail behaviour. Continuous variables related to gas and value, such as `gas_used`, `gas_efficiency`, `total_gas_cost`, `effective_gas_price`, `value`, and `cumulative_gas_used`, show large gaps between their means and medians, as well as ranges spanning several orders of magnitude, indicating heavy-tailed behavior. In contrast, temporal variables such as `hour`, `day`, and `unix_timestamp`, together with cyclical encodings and binary indicators such as `event_activity_flag`, `address_is_sender`, `address_is_receiver`, and `chain_id`, exhibit compact ranges and a more regular spread.

To quantify extremal behavior, univariate outliers were identified using the standard Interquartile Range (IQR) rule, flagging observations lying outside $[Q_1 - 1.5\ \text{IQR},\ Q_3 + 1.5\ \text{IQR}]$ for each numerical feature. This analysis revealed substantial numbers of high-end outliers in several

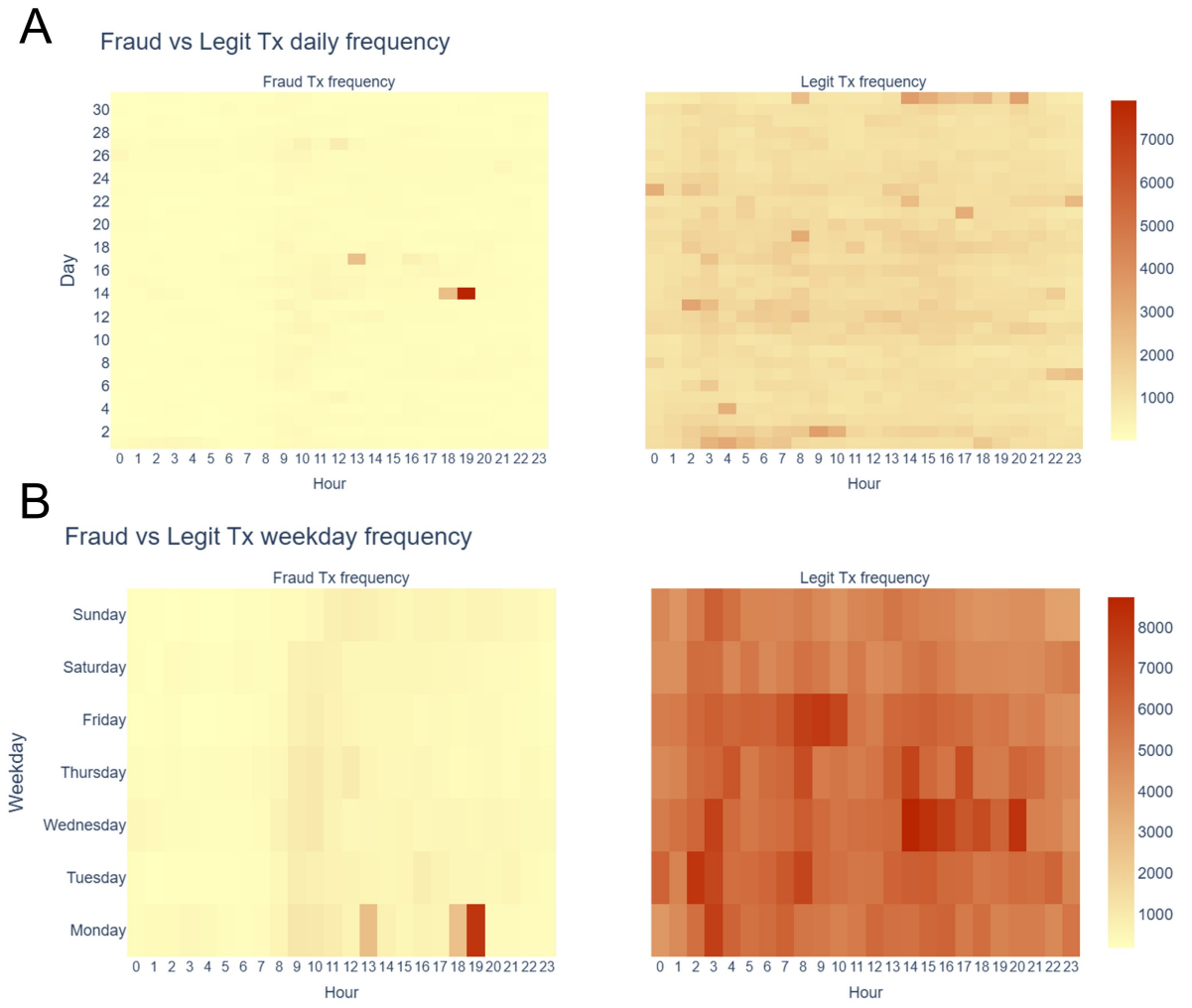


Figure 3.4: Heatmap showing Tx frequency by hour and day of month **(A)** and by hour and weekday **(B)**.

key variables, including `gas_used`, `value`, `total_gas_cost`, `effective_gas_price`, and count-like measures such as `number_of_errors`, `number_of_unique_from_address`, and `number_of_unique_to_address`. These outliers correspond to rare but very large Txs, gas usage values, or address activity levels and contribute to long right tails in the distributions.

Skewness coefficients were also computed to characterize the asymmetry of each distribution. Most gas and wallet-related features are strongly right-skewed, with skewness values often exceeding 30 and, in some cases, reaching several hundred, as seen for variables such as `value`, `total_gas_cost`, and `effective_gas_price`. A smaller set of features shows mild or near-zero skewness, including `hour`, `day`, the sinusoidal encodings `hour_sin` and `hour_cos`, and the main binary indicators. The only feature exhibiting strong negative skewness is `gas_price_ratio`.

3.3.1 Class-conditional distributions

Given the considerable right skew and heavy tails observed in the gas- and value-related variables, a $\log(1+x)$ transformation was applied to a subset of continuous features, including `gas_used`, `gas_efficiency`, `total_gas_cost`, `effective_gas_price`, `value`, `number_of_errors`, and the address-activity counts. **Figures 3.5, 3.6** and **3.7** presented in this section show violin plots comparing the marginal distributions of selected linear and log-transformed features between fraudulent Txs (flag = 1, crimson) and legitimate Txs (flag = 0, teal) when the linear representation is too skewed, making analysis of the data difficult.

The most pronounced differences appear in features related to operational behavior. `number_of_errors` (log-transformed) and `error_rate` (log-transformed) shows a clear rightward shift for fraudulent cases, with both higher medians and heavier upper tails, suggesting that fraudulent activity is associated with increased execution failures (**Figure 3.6**). This multimodal structure in the fraud class is informative, as it suggests the presence of distinct subgroups within the fraudulent Txs that differ in their error behavior, potentially reflecting different types of attack strategies or exploitation patterns.

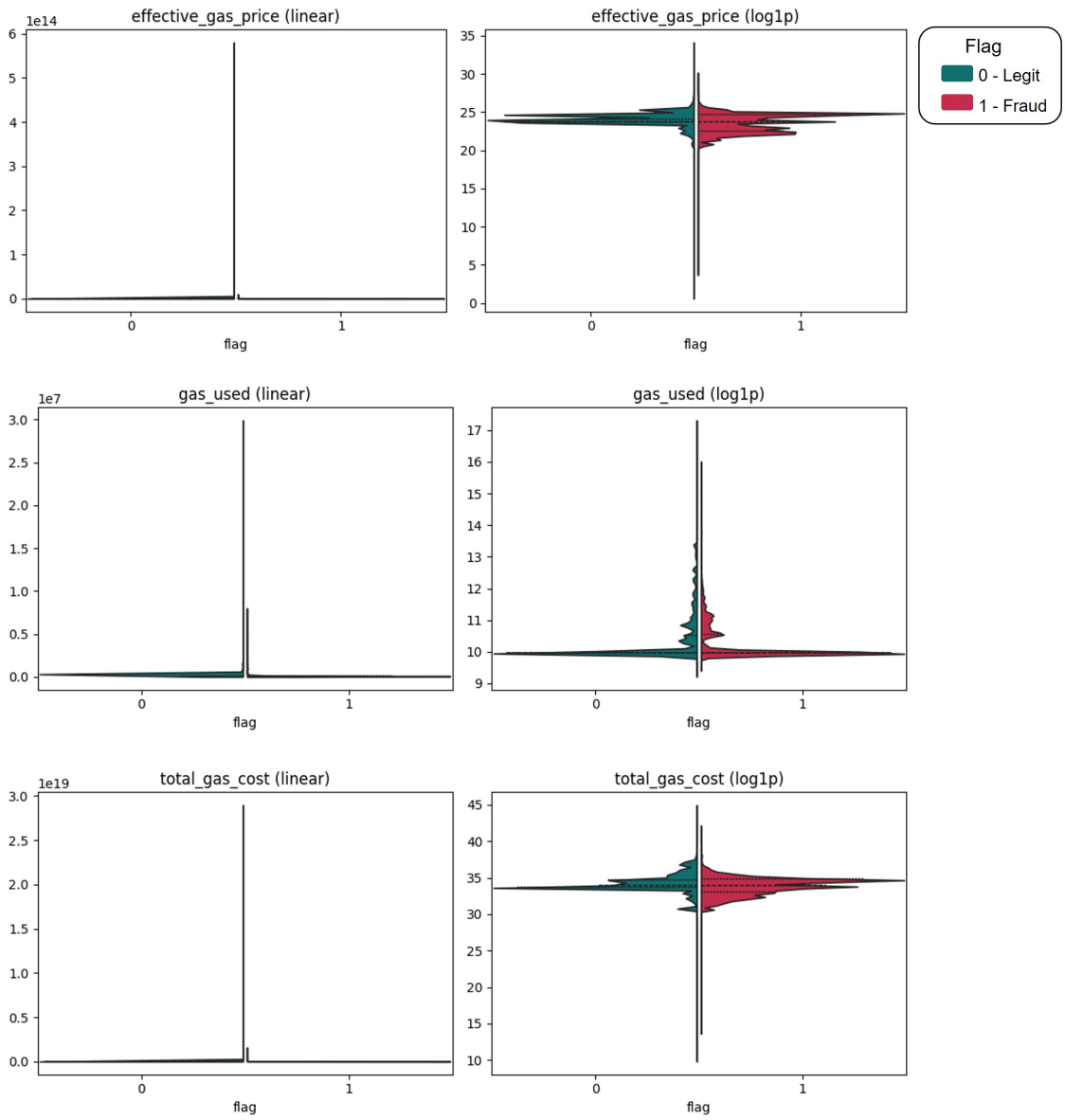


Figure 3.5: Split violin plots of right-skewed numeric Tx features. The left panel displays the raw linear scale and the right panel displays the $\log(1+x)$-transformed scale, applied to mitigate the influence of extreme values and improve distributional visibility. The internal lines within each violin indicate the quartiles of the respective distribution, with the central line marking the median. Flag 0 corresponds to legitimate Txs and flag 1 corresponds to fraudulent Txs.

Similarly, `cumulative_gas_used` is higher and more dispersed for fraudulent Txs, indicating greater computational costs. The `effective_gas_price` (log-transformed) seen in **Figure 3.5** distributions is shifted upward for fraudulent cases, suggesting that attackers may prioritize Tx inclusion by paying higher gas fees during congested periods. The `value` feature also concentrates at higher values for fraud cases, consistent with targeted high-value transfers rather than the broader range seen in legitimate activity. The legitimate class clusters tightly around a narrow range, while the fraudulent class is again more dispersed and occupies a wider

region of the transformed scale. In particular, `effective_gas_price` and `total_gas_cost` reveal that fraudulent Txs are not uniformly distributed across the log scale but instead show a pronounced concentration at certain gas price levels alongside a longer tail, which may reflect adversarial behavior involving unusually high gas fees to front-run or prioritize malicious Txs. For `cumulative_gas_used`, the fraud class shows a distinct rightward shift in the upper quartile compared to the legitimate class, indicating that fraudulent wallets tend to be associated with blocks of higher overall gas consumption.

The `gas_efficiency` metric, defined in **Equation** (3.2), and `cumulative_gas_used`, shown in **Figure 3.7**, also differs across classes. Fraudulent Txs exhibit a higher median efficiency and a longer upper tail, which may reflect either automated execution strategies or exploitation of favorable gas-market conditions.

$$\texttt{gas_efficiency} = \frac{value_{tx}}{total_gas_cost} = \frac{value_{tx}}{gas_used \times effective_gas_price} \tag{3.2}$$

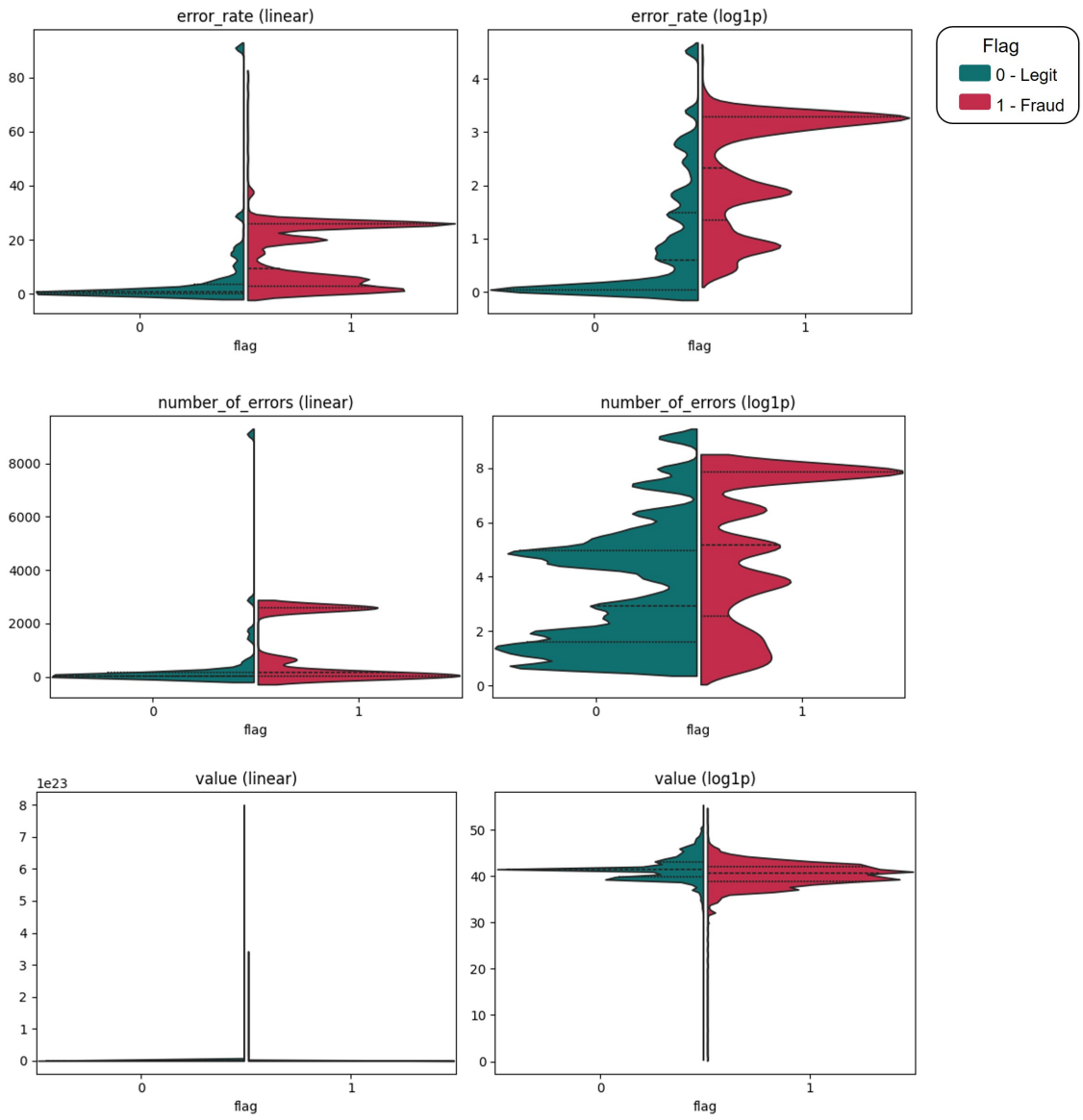


Figure 3.6: Split violin plots of right-skewed numeric Tx features. The left panel displays the raw linear scale and the right panel displays the $\log(1+x)$-transformed scale, applied to mitigate the influence of extreme values and improve distributional visibility. The internal lines within each violin indicate the quartiles of the respective distribution, with the central line marking the median. Flag 0 corresponds to legitimate Txs and flag 1 corresponds to fraudulent Txs

Tx `value` (**Figure 3.6**) after log transformation, shows the heaviest overlap between classes compared to the other features, with both groups sharing a dominant peak around the same log scale range. This suggests that the Tx value alone may be a weak discriminator, though its upper tail may still carry a marginal signal when combined with other features.

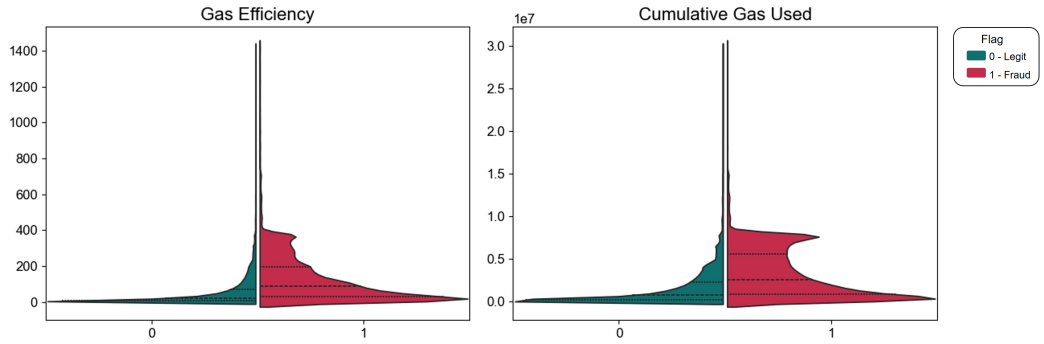


Figure 3.7: Split violin plots of gas-related features on the raw linear scale. The internal lines within each violin indicate the quartiles of the respective distribution, with the central line marking the median. Flag 0 corresponds to legitimate Txs and flag 1 corresponds to fraudulent Txs.

3.3.2 Correlation matrix

A bivariate analysis of the raw numeric features was also performed. **Figure 3.8** shows the Pearson correlation matrix, revealing several strong structural relationships driven by blockchain mechanics as well as moderate associations that may be useful for fraud prediction.

The most obvious pattern is the near-perfect correlation between `unix_timestamp` and `block_number` ($\rho = 0.999$), reflecting their deterministic relationship in sequential blockchain data and possibly also the sampling strategy, since a large proportion of the fraud cases were collected later in time. Block-level features appear to form a tight cluster, `cumulative_gas_used` correlates strongly with both `index` ($\rho = 0.822$) and `gas_efficiency` ($\rho = 0.829$), while `block_number` shows moderate correlation with `index`, `gas_efficiency`, and `cumulative_gas_used`. These relationships arise naturally from block construction dynamics, where later Txs within a block accumulate more gas usage and may exhibit different efficiency levels.

Moderate correlations also appear among activity-related metrics. `num_transaction` is positively associated with address diversity ($\rho \approx 0.37$-0.39 for sender and receiver counts) and with `number_of_errors` ($\rho = 0.35$), indicating that higher activity contexts tend to produce more errors and more unique addresses.

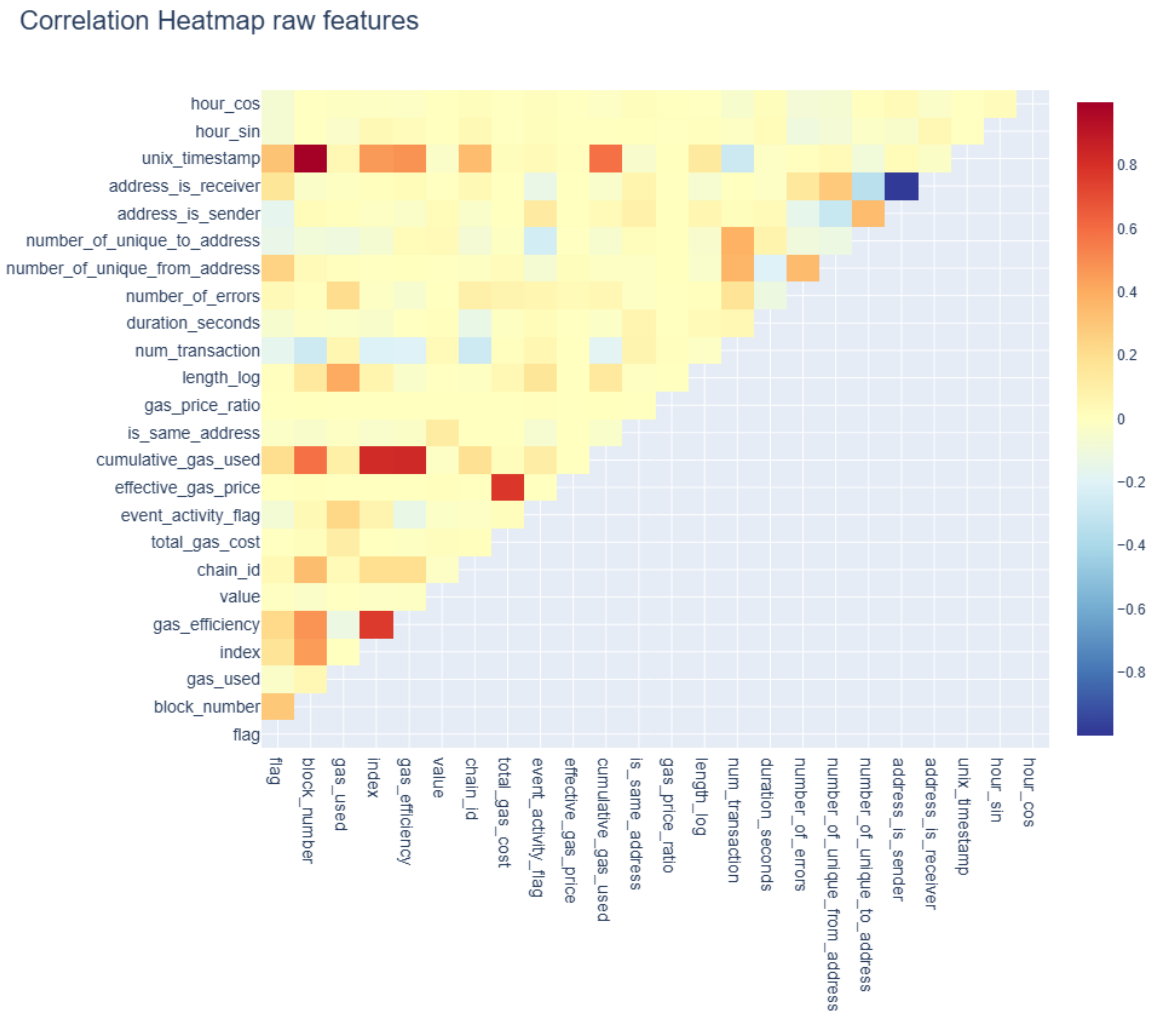


Figure 3.8: Pearson correlation matrix of raw numeric features.

The fraud indicator `flag` shows its strongest raw correlations with block structure ($\rho = 0.315$ with `unix_timestamp` and $\rho = 0.296$ with `block_number`) and with sender address diversity ($\rho = 0.252$), while it is negatively associated with Tx volume ($\rho = -0.164$ with `num_transaction`). Gas-related metrics show only weak direct correlations with `flag` ($\rho < 0.23$).

To complement the correlation matrix, a correlation network is shown in **Figure 3.9**. This network represents feature relationships using edges for $|\rho| \geq 0.25$. The threshold was selected so that the `flag` feature would also appear in the network, although it is only slightly correlated with features such as `block_number` and `unix_timestamp`. A dense clique (a fully connected subset of nodes in a graph) emerges among the block-structural features (`block_number`, `cumulative_gas_used`, `gas_efficiency`, and `index`), connected by thick edges reflecting $\rho > 0.45$. Address-diversity metrics form a secondary hub, while the fraud indicator `flag` acts as a connector between block structure and behavioral signals such as Tx volume and address counts.

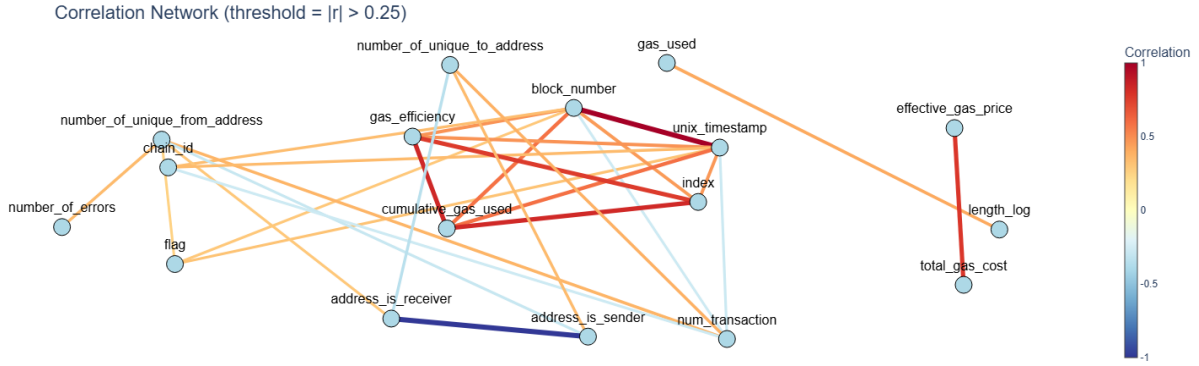


Figure 3.9: Correlation network based on the raw feature matrix. The proximity of features indicates stronger interaction, while edge color indicates the sign of the correlation. The threshold is $|\rho| \geq 0.25$.

Chapter 4

Methodology

This chapter describes the end-to-end methodology developed to detect fraudulent activity in Ethereum-based Transaction (Tx)s. The pipeline takes into account temporal integrity, fair comparability between model families, and leakage prevention at every stage of the experimental process. After describing the overall pipeline structure, the chapter details the Expanding-Window Cross-Validation (EW-CV) scheme, the construction of the directed bipartite heterogeneous graphs, the tabular baseline models, the Graph SAmple and aggreGatE (GraphSAGE) architecture, and the Embedding-Space SMOTE (Embed-SMOTE) variant.

4.1 Pipeline Overview

In **Figure 4.1**, it is possible to see an illustration of the experimental pipeline used in this dissertation, from raw Ethereum Tx data to graph construction, baseline modeling, Graph Neural Network (GNN) training, embedding-space oversampling, and final evaluation. The pipeline was designed to ensure a fair comparison between tabular and graph-based approaches while preserving temporal order and preventing information leakage.

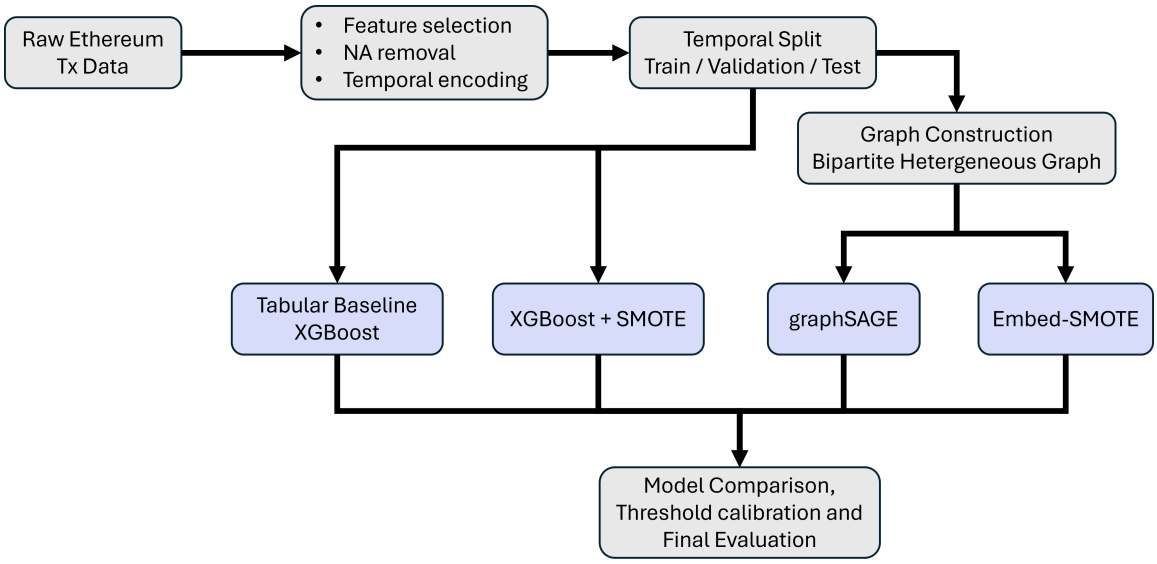


Figure 4.1: End-to-end experimental pipeline.

4.1.1 Adaptations from Literature Review

The literature review led to several methodological choices in this dissertation. First, GraphSAGE is selected as the core GNN architecture due to its inductive capability and suitability for unseen nodes [8]. Second, cyclical sine-cosine encodings are used for timestamp features to reduce boundary artifacts in temporal variables [42]. Third, random oversampling in the original feature space is avoided in favor of graph-aware imbalance strategies such as Synthetic Minority Oversampling Technique (SMOTE) variants [9]. Finally, Area Under the Precision-Recall Curve (AUC-PR) is treated as a primary evaluation metric because Area Under the Receiver Operating Characteristic Curve (AUC-ROC) can be overly optimistic in highly imbalanced settings [39].

4.1.2 Cross-validation fold definition

Standard k-fold cross-validation is not appropriate for time-dependent data, as random shuffle may allow future observations to influence the prediction of past ones [45, 46]. To address this limitation, an EW-CV scheme was adopted [45]. Three folds were used, selected according to the temporal granularity of the dataset and the available fraud-active years. The years 2015 and 2016 were excluded from the split because they contained no fraud cases, including them would not contribute to fraud characterization and could weaken the fraud signal by introducing a large proportion of non-fraudulent observations from an early period. This excluded period contained 182,276 Txs, corresponding to 17.97% of the total entries.

The folds were defined as follows:

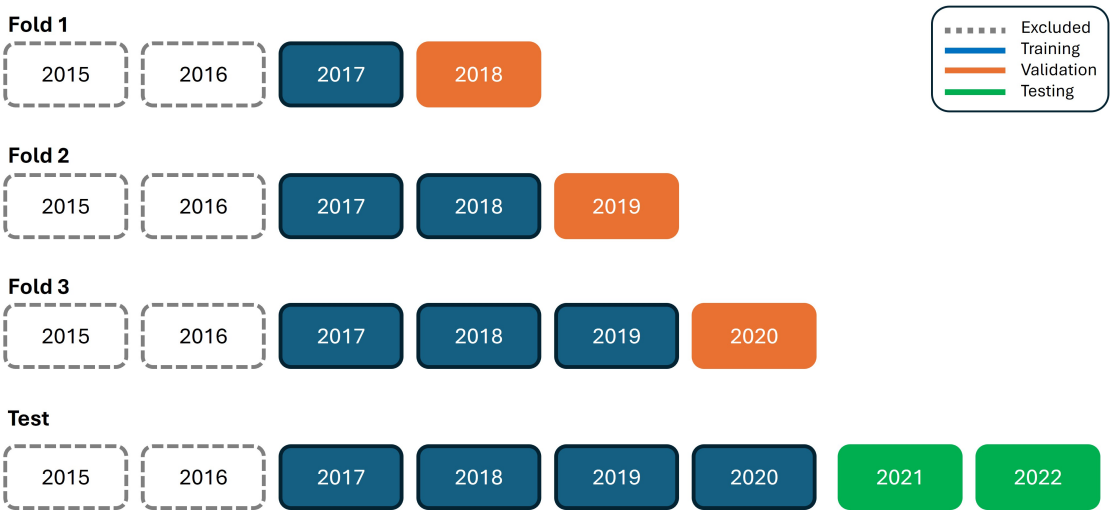


Figure 4.2: EW-CV protocol.

The years 2021 and 2022 were grouped as the final test period as they represented the most recent portion of the dataset and together provided a sufficiently large number of Txs for evaluation. Each fold expands the historical training window by one year, thereby simulating sequential deployment in a realistic fraud detection setting. This structure also exposes temporal distribution shift, as the prevalence of fraud varies substantially across folds, ranging from 1.86% to 20.75% as seen in **Figure 3.3**.

4.2 Bipartite Graph Construction

To capture the relational structure of Ethereum Txs, directed bipartite heterogeneous graphs were constructed using PyTorch Geometric's `HeteroData` format. The graph contains two node types, wallets and Txs, and two directed edge types representing the flow of funds. Each Tx node is connected to exactly one sender wallet and one receiver wallet ($wallet \xrightarrow{\text{from sender}} tx$) and one receiver wallet ($tx \xrightarrow{\text{to receiver}} wallet$), yielding $2 \times N_{tx}$ directed edges for N_{tx} Txs. A schematic example is shown in **Figure 4.3**.

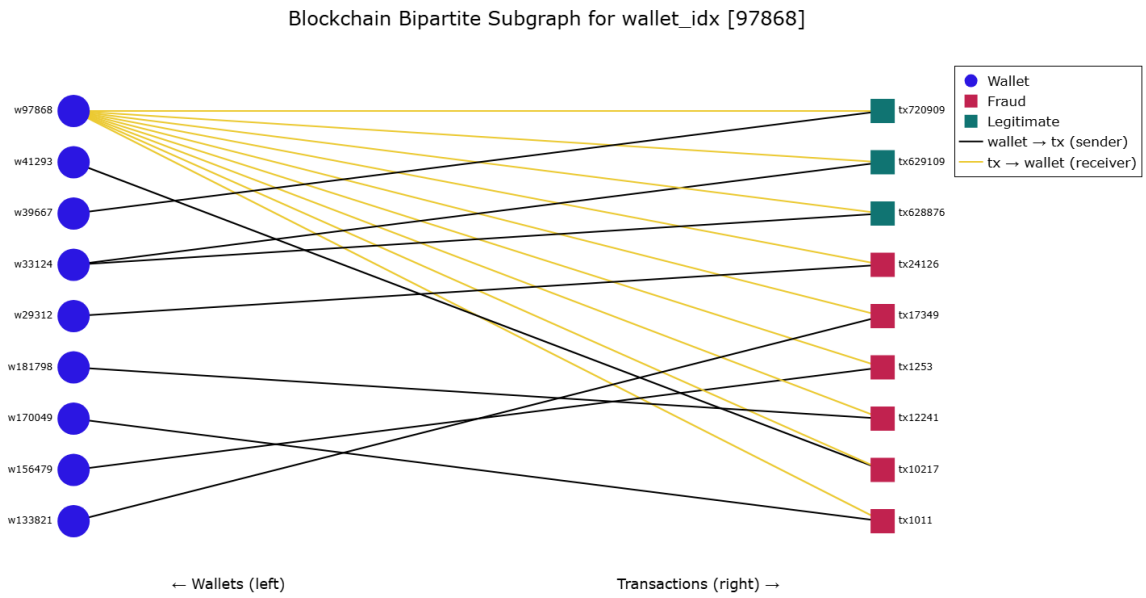


Figure 4.3: Bipartite graph schema. Wallets connect bidirectionally to Txs through sender and receiver relations. Node features encode intrinsic and structural statistics.

This bipartite formulation avoids direct wallet-to-wallet edges and instead routes all connectivity through Tx nodes. As a result, the model can jointly reason about the properties of individual Txs and the behavioral patterns of the wallets involved. Representing Txs as first-class nodes, rather than as edges between wallets, preserves their rich feature space, including gas usage, value, temporal features, and event flags, and allows the GNN to learn directly from Tx-level attributes during message passing. The node indices are local to each fold, so each graph is built independently without sharing a global index space between folds.

A separate graph is constructed for each cross-validation fold following the EW-CV scheme. A final held-out test graph covering 2021-2022 is constructed once and kept untouched until model selection is complete. **Table 4.1** summarizes the size of each bipartite subgraph.

The graph construction procedure follows an inductive logic consistent with GraphSAGE as each fold contains both training and validation Txs, with binary masks distinguishing the two splits during training. This design allows the model to generalize to wallet nodes unseen during training, which is an important property in blockchain settings where new addresses are added continuously.

The node features are computed separately for each node type. Tx features encode gas consumption, transferred value, temporal patterns through cyclical encodings, and behavioral flags. Wallet features are derived exclusively from training-periods Txs to prevent leakage and include sender and receiver statistics such as degree, Tx volume, and activity duration, as well as PageRank computed on the training-period Tx graph as a measure of structural centrality. All feature scaling is fit on training data only and then applied to validation nodes using transform-

only normalization.

Wallet nodes aggregate both intrinsic features and structural statistics computed from observed Txs:

- **Sender perspective**: out-degree, total/average/max value sent, unique receivers, average gas metrics, activity span, and inter-Tx timing.
- **Receiver perspective**: in-degree, total/average/max value received, unique senders, and activity span.
- **Network centrality**: PageRank score ($\alpha = 0.85$) computed on the directed Txs graph.
- **Ratio features**: in/out-degree ratio and net value asymmetry (received funds minus sent funds).

4.2.1 Wallet Node Features

The wallet nodes are represented by feature vectors constructed from three complementary sources. Wallet-level attributes and aggregated behavioral statistics are included where available, structural features are computed from training-periods Txs only and include in-degree, out-degree, total, average, and maximum transferred value, number of unique counterparties, gas consumption patterns, temporal activity spans, and average inter-Tx intervals. Lastly, a graph-theoretic feature is incorporated through PageRank [47], calculated on the directed Tx graph using a damping factor of $\alpha = 0.85$. Two derived ratios are also appended: the in-to-out degree ratio and value asymmetry, defined as received value minus sent value. All wallet features are standardized using parameters computed on training-period wallets only and then applied to validation wallets via transform, thereby preventing leakage of future information.

4.2.2 Tx Node Features

Each Tx node is represented by a feature vector that captures gas consumption metrics, monetary value, chain identifier, temporal signals, and cyclical encodings of hours, days, months, and weekdays. The fraud label is reserved exclusively as the node-level classification target and is excluded from the feature matrix to avoid label leakage. Edge attributes encode Tx value to capture the magnitude of economic flow. Tx nodes were represented by 21 features. These included gas-related variables, value, chain metadata, and cyclical temporal encodings. Two additional derived features were included: an in-to-out degree ratio and a value asymmetry measure, defined as the received value minus the sent value.

All Tx features were standardized using parameters computed on the training nodes only. The fitted scalers were then applied to the validation nodes using transform-only normalization, ensuring that the validation period did not influence the feature distributions used during training.

The same rule was applied to the final test graph, where the model could observe the training context but not the held-out test period during preprocessing.

The resulting graphs were large but sparse enough to require mini-batch graph learning rather than full-batch training. The fold-level graph sizes are reported in **Table 4.1**. The edge count is exactly twice the number of unique Txs in each graph, since every Tx contributes one sender edge and one receiver edge. This makes the graph structurally consistent across folds while allowing the number of wallets and Txs to grow as the temporal window expands. The increase in node and edge counts reflects the EW-CV design, as more years are added to the training context, the graph becomes larger and more informative while remaining manageable for neighborhood-sampled GraphSAGE training.

Each fold graph was serialized to disk together with metadata containing the node mappings, fitted scalers, and label encoder. This allowed the graph to be reused directly during model training without rebuilding it for every experiment, while preserving the exact preprocessing configuration used during construction.

Table 4.1: Fold-level graph statistics for the heterogeneous Tx graphs

Graph	Wallet nodes	Tx nodes	Edges
Fold 1	147,558	713,332	1,426,664
Fold 2	151,725	734,253	1,468,506
Fold 3	155,463	757,541	1,515,082
Final test graph	157,511	778,351	1,556,702

4.3 Baseline Model: eXtreme Gradient Boosting (XGBoost)

XGBoost was selected as the baseline classifier for this fraud detection project due to its robustness, scalability to large datasets, and strong empirical performance on tabular data with mixed feature types and class imbalance [48].

The XGBoost baseline was implemented using a temporally rigorous and leakage-aware modelling pipeline. With the binary `flag` serving as the target variable, the classifier was trained on the processed feature set. Before training, observations were sorted chronologically according to the `unix_timestamp` feature, which was then dropped to prevent leakage of future information. The dataset was divided into three folds as described in **Section 4.1.2**. To improve robustness, each fold configuration was repeated across ten random seeds, and the best seed was selected according to mean validation AUC-PR, which served as the main selection metric due to the class imbalance in the dataset.

A second XGBoost-based classifier was developed which combined XGBoost with SMOTE to assess whether synthetic minority oversampling could improve fraud detection under the same experimental conditions. The final minority ratio was set at 25%, which was the best-

performing ratio among the tested values {10, 15, 20, 25, 30, 40, 50}. Oversampling was applied exclusively to the training partition of each fold to prevent synthetic samples from contaminating validation or test evaluations. Both models were built on the same preprocessed feature set and evaluated using an identical temporal validation framework to ensure a fair comparison.

Both models used the same XGBoost configuration, a binary logistic objective, AUC-PR as the training metric, maximum depth of 6, learning rate of 0.1, subsampling and column subsampling rates of 0.8, minimum child weight of 5, gamma of 0.1, L1 regularization of 0.1, and L2 regularization of 1.0. Up to 500 boosting rounds were permitted during cross-validation, with early stopping after 30 rounds without improvement in validation AUC-PR. For each run, the classification threshold was tuned on the validation fold by maximizing the F1-score over the precision-recall curve, and the final threshold applied at the test time was taken as the mean of the thresholds from the best-performing seed.

In both cases, validation and test data remained untouched during resampling and threshold selection, preserving the integrity of model evaluation. After cross-validation, each model was retrained on the full training data with the chosen configuration and evaluated on the test set.

4.4 GraphSAGE for Heterogeneous Graphs

To incorporate relational information that cannot be captured by tabular Tx features alone, a GraphSAGE [49] model was adapted to the bipartite heterogeneous Tx graphs evaluated under the EW-CV protocol rather than the static benchmark graphs used in the original work. The model was formulated as an inductive node-classification task in which the Tx nodes were classified as fraudulent or legitimate by aggregating information from their local neighborhoods, while the wallet nodes provided structural context through graph connectivity. This inductive formulation was appropriate for the fraud detection setting as new wallets and Txs appear continuously over time, so the model had to generalize to previously unseen nodes rather than rely on a fixed embedding table. The GraphSAGE backbone was implemented in PyTorch Geometric using the `to_hetero` wrapper, which converts a homogeneous message-passing architecture into a heterogeneous model with relation-specific parameters for each type of edge.

The architecture consisted of two stacked `SAGEConv` layers with a hidden dimension of 256, Rectified Linear Unit (ReLU) activations, and dropout of 0.3. The resulting Tx embeddings were passed to a two-layer Multi-Layer Perceptron (MLP) classifier with the structure `Linear(256 → 64) → ReLU → Dropout(0.3) → Linear(64 → 2)` to produce binary fraud predictions. Mean aggregation was used across relation types, which allowed the model to combine local neighborhood information while remaining computationally tractable on large graphs.

Because the Tx graphs were too large for full-batch training, learning was carried out with mini-batch neighborhood sampling using `NeighborLoader`. For each target Tx node, up to 25 neighbors were sampled on the first hop and 10 on the second hop for each edge type,

with a batch size of 1,024 Tx nodes. This sampling strategy reduced memory requirements and supported inductive inference, since unseen validation and test nodes could be processed through sampled local neighborhoods without appearing during model fitting.

Class imbalance was handled using weighted cross-entropy loss, with class weights recomputed separately for each fold from the training labels only. This prevented leakage from future periods and ensured that the loss function reflected the class distribution observed in the current training window rather than in the full dataset. Optimization used Adam with a learning rate of 10^{-3}, while a `ReduceLROnPlateau` scheduler adjusted the learning rate according to validation AUC-PR.

The selection of the model followed the same EW-CV design used for the tabular baselines to preserve comparability between model families. Validation AUC-PR was used as the primary selection and early-stopping metric. After training, the classification threshold was tuned on the validation set by maximizing the F1-score over the precision-recall curve, and the mean threshold across the best seed's folds was carried forward to the final evaluation.

To preserve the integrity of the test evaluation, the held-out 2021-2022 graph was kept locked throughout cross-validation and opened only after model selection was complete. During cross-validation, `NeighborLoader` operated only on training and validation nodes. After selecting the best seed, the final GraphSAGE model was retrained from scratch on the full development graph covering 2017-2020 and evaluated exactly once on the held-out test set. This procedure ensured that both model selection and threshold tuning remained completely independent of the final test results.

4.5 Addressing Class Imbalance: Embed-SMOTE

Imbalanced classes are particularly challenging for GNNs, as the majority class can dominate the loss function and bias the model toward predicting legitimate Txs. To mitigate this issue, an embedding-space oversampling strategy was added to this work. The proposed approach is inspired by GraphSMOTE [9] and by the broader idea of generating synthetic minority examples in a learned embedding space. This model builds on the general idea introduced by Ando and Huang *et al.* [10], in which synthetic minority examples are generated in a learned embedding space rather than the raw input space. The present implementation is adapted to an inductive, heterogeneous, bipartite Tx graph trained with mini-batch neighborhood sampling, and therefore does not reproduce the full structural augmentation mechanism of GraphSMOTE [9]. Instead, it applies SMOTE-style interpolation directly in the learned embedding space of Tx nodes.

A faithful implementation of full GraphSMOTE was not practical in this setting for several reasons. The model is trained with `NeighborLoader` mini-batches, so synthetic nodes and predicted edges would exist only within a sampled subgraph and would not persist across batches. The fraud graph is also heterogeneous and bipartite, while the original GraphSMOTE

formulation assumes a homogeneous graph, which would require a more complex relation-specific edge-generation mechanism. Lastly, this project uses an inductive temporal evaluation setup in which future nodes remain unseen during training, while GraphSMOTE was originally designed for a transductive full-graph setting. For these reasons, the model is best described as an Embed-SMOTE adaptation rather than a full GraphSMOTE implementation.

Architecturally, the model uses a heterogeneous GraphSAGE backbone with two stacked `SAGEConv` layers transformed through PyTorch Geometric's `to_hetero` wrapper, allowing relation-specific message passing across the two edge types in the bipartite graph. The hidden dimension is 256, the embedding dimension before classification is 64, and dropout of 0.3 is applied to reduce overfitting. A two-layer MLP classifier then maps Tx embeddings to the binary fraud prediction task.

The forward pass is explicitly decomposed into an encode stage and a classify stage so that synthetic minority embeddings can be inserted between graph encoding and final classification. During each training mini-batch, the Tx-node embeddings of the minority class are extracted and used as the basis for synthetic oversampling. A k-nearest neighbors model with $k = 5$ is fitted on the fraud embeddings in the current batch, and synthetic embeddings are created by linear interpolation between a selected fraud embedding and one of its nearest minority neighbors. This follows the core SMOTE principle of creating new minority samples as combinations of nearby minority instances rather than by simple duplication.

The oversampling ratio was set to 15%, which was the best-performing value among the tested ratios {10, 15, 20, 25, 30, 40, 50}. Augmentation was applied only to training embeddings and never to validation or test data. If too few minority examples were available in a batch for standard interpolation, the implementation fell back to a small Gaussian perturbation of the available minority embedding to avoid failure cases in sparse batches. Synthetic embeddings were concatenated with the real training embeddings and passed to the classifier head together with their fraud labels.

Class imbalance was handled through a dual strategy combining embedding-space oversampling with class-weighted cross-entropy loss. The class weights were recomputed separately for each fold from the training labels only, ensuring that the loss reflected the class distribution observed in the current training window. Training used Adam with a learning rate of 10^{-3}, batch size 1024, dropout 0.3, and early stopping with patience 10 monitored on validation AUC-PR. This is consistent with the broader evaluation design of the project, where AUC-PR is preferred over threshold-dependent metrics for model selection under strong class imbalance.

The model follows the same EW-CV scheme as the other models in the project. The procedure is repeated across ten random seeds, and the best seed is selected by mean validation AUC-PR across folds rather than by the single best run. Classification thresholds are tuned only after training using validation predictions, rather than being updated during training. As in the other models, the test graph remains locked until cross-validation and model selection are complete.

4.6 Random Seed Strategy and Reproducibility

To ensure robust and reproducible results, all models were trained across ten random seeds, following a Fibonacci-based sequence {8, 13, 21, 34, 55, 89, 144, 233, 377, 610} to avoid arbitrary selection bias. In the XGBoost models, the random seed manages three sources of stochasticity, namely, row subsampling, column subsampling, and tie-breaking during tree construction. In the GraphSAGE model, the seed additionally controls weight initialization, mini-batch neighborhood sampling, training batch ordering, and the dropout mask applied during forward passes. In models employing SMOTE, the same seed value is passed to both the oversampling procedure and the initialization of the model, ensuring that the synthetic training distribution and the starting configuration of the model are consistently paired between runs. For each seed, the model is evaluated across all three EW-CV folds, and the seed yielding the highest mean AUC-PR across folds is selected for final training and test evaluation.

This multi-seed approach provides an estimate of variance due to random initialization rather than the learning algorithm itself. The inclusion of this approach also allows for reproducibility of results while also selecting the best seed based on mean cross-validation performance rather than a single fold with a favorable configuration, thus strengthening the validity of the final held-out test evaluation [50].

4.7 Computational Resources and Software Environment

All experiments in this dissertation were conducted on a local workstation, equipped with an AMD Ryzen 7 5800H processor running at 3.20 GHz, 16.0 GB of installed RAM, and an NVIDIA GeForce RTX 3060 Laptop GPU with 6 GB of dedicated memory. The system also includes integrated AMD Radeon graphics and uses a 64-bit Windows 11 operating system on an x64-based architecture. These specifications provided sufficient computational capacity for graph construction, feature preprocessing, model training, and repeated temporal cross-validation experiments. A full GraphSMOTE implementation was also explored, but the combination of temporally split fold-specific subgraphs, mini-batch graph processing, and synthetic graph augmentation exceeded the available memory on the local workstation.

The implementation was developed in Python 3.12.5 using a scientific computing and Machine Learning (ML) stack that included PyTorch 2.4.0 with CUDA 11.8 support, PyTorch Geometric 2.7.0, XGBoost 3.2.0, scikit-learn 1.8.0, pandas 3.0.1, NumPy 1.26.4, and imblearn 0.14.1. This software environment supported both tabular and graph-based modeling, as well as the imbalance-handling procedures used in the experiments.

Chapter 5

Results

This chapter presents the experimental results and their interpretation. It begins with the selection of the statistical test used for inferential comparisons, followed by a presentation of the Expanding-Window Cross-Validation (EW-CV) and final holdout test results across all four models evaluated. The results are then discussed in terms of the relative contributions of graph structure, class imbalance handling, and temporal generalization to fraud detection performance. The chapter also includes a comparison against prior work on Ethereum fraud detection to contextualize the findings within the broader literature.

5.1 Statistical test selection

As a preliminary step, the Shapiro-Wilk test was applied to the paired differences between models to check whether the assumption of normality was satisfied. This assumption is relevant as it determines whether a parametric or non-parametric inferential test is more appropriate. The results indicated that, among the evaluated metrics (Area Under the Receiver Operating Characteristic Curve (AUC-ROC), Area Under the Precision-Recall Curve (AUC-PR), fraud F1-score, fraud recall, and fraud precision), only AUC-ROC and AUC-PR were approximately normally distributed for comparisons between the eXtreme Gradient Boosting (XGBoost) baseline $\leftrightarrow$ XGBoost + Synthetic Minority Over-sampling Technique (SMOTE), while fraud recall was the only metric that appeared approximately normal for the comparisons between the XGBoost baseline $\leftrightarrow$ Graph SAmple and aggreGatE (GraphSAGE), and between the XGBoost baseline $\leftrightarrow$ Embedding-Space SMOTE (Embed-SMOTE). Since the majority of paired differences did not satisfy the normality assumption, the paired Wilcoxon signed-rank test was selected for the inferential analysis [51].

5.2 Models results

The EW-CV results revealed a clear separation between the tabular and graph-based approaches (**Table** 5.1). Across the three expanding-window folds and the repeated random seeds, the XGBoost baseline showed the weakest and least stable performance, particularly in the third fold. XGBoost + SMOTE achieved only a marginal improvement over the baseline and retained a similar degree of instability across folds. In contrast, both GraphSAGE and Embed-SMOTE achieved consistently high validation performance throughout the temporal folds, with GraphSAGE obtaining the strongest overall cross-validation scores as can be seen in **Figure 5.1** and **Figure 5.2**.

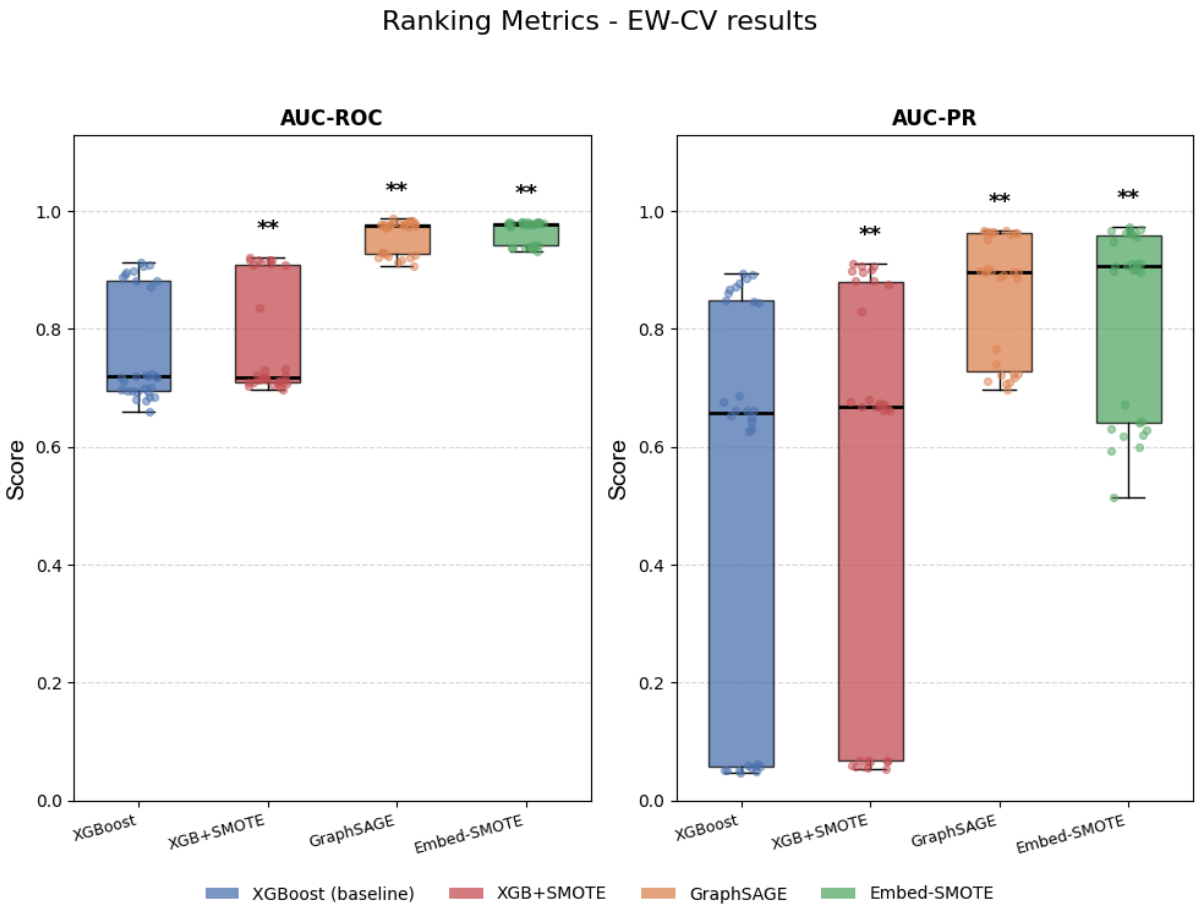


Figure 5.1: Distribution of ranking metrics across the evaluated models over the 10 random-seed validation runs. Jittered points show per-split values, and double asterisks (**) indicate statistically significant differences ($p \leq 0.01$) relative to the XGBoost baseline after the Bonferroni correction ($\alpha = 0.0167$).

The paired Wilcoxon signed-rank tests demonstrated that graph-based models yielded statistically significant improvements over the XGBoost baseline. After Bonferroni correction ($\alpha = 0.0167$), GraphSAGE significantly outperformed the baseline across all metrics evaluated. Embed-SMOTE also improved from the baseline on all metrics, although the magnitude of the

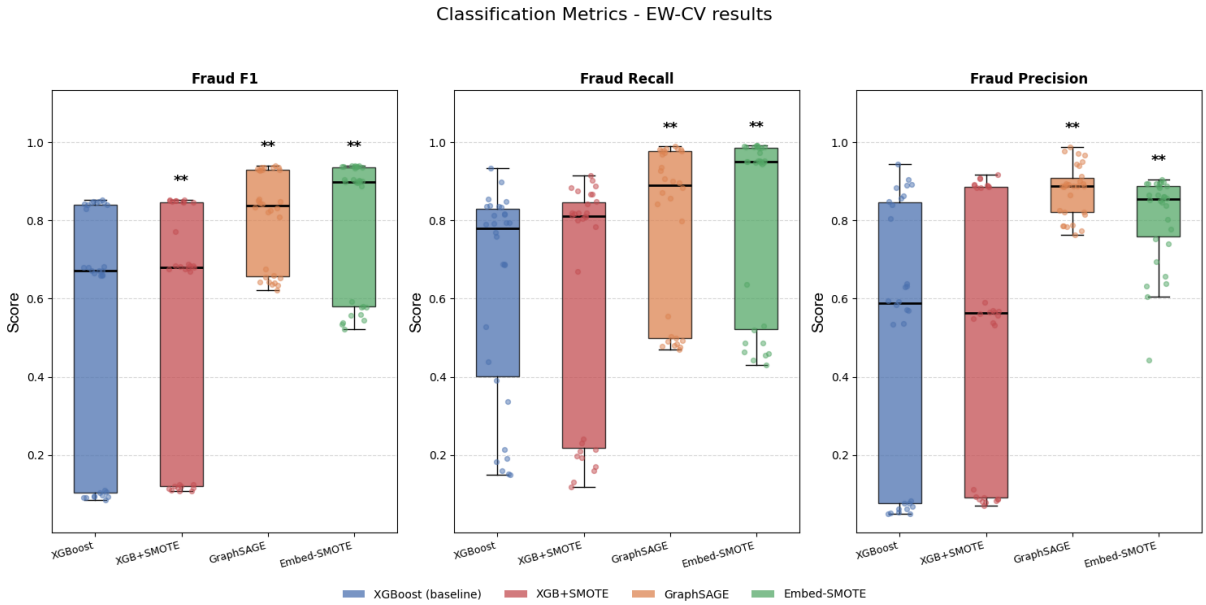


Figure 5.2: Distribution of fraud-class classification metrics across the evaluated models over the 10 random-seed validation runs. Jittered points show per-split values, and double asterisks (**) indicate statistically significant differences ($p \leq 0.01$) relative to the XGBoost baseline after the Bonferroni correction ($\alpha = 0.0167$).

gains was consistently smaller than that observed for GraphSAGE. By contrast, XGBoost + SMOTE produced statistically significant improvements only in AUC-ROC, AUC-PR, and fraud F1-score, while differences in fraud recall and fraud precision were not statistically significant.

Table 5.1: Cross-validation performance for all models.

Model	Fold 1 AUC-PR	Fold 2 AUC-PR	Fold 3 AUC-PR	Mean AUC-PR $\pm$ SD	Fold 1 F1	Fold 2 F1	Fold 3 F1	Mean F1 $\pm$ SD
XGBoost baseline	0.6551	0.8691	0.0534	0.5259 $\pm$ 0.3515	0.6702	0.8443	0.0962	0.5369 $\pm$ 0.3252
XGBoost + SMOTE	0.6693	0.8888	0.0615	0.5399 $\pm$ 0.3562	0.6806	0.8421	0.1157	0.5461 $\pm$ 0.3171
GraphSAGE	0.8956	**0.9633**	**0.7219**	**0.8603 $\pm$ 0.1041**	0.8366	0.9337	**0.6458**	**0.8054 $\pm$ 0.1222**
Embed-SMOTE	**0.9049**	0.9630	0.6159	0.8280 $\pm$ 0.1562	**0.8998**	**0.9384**	0.5588	0.7990 $\pm$ 0.1740

On the final holdout test set visible in **Table 5.2**, the overall ranking was similar to that observed in validation results (**Table 5.1**). The tabular baselines performed modestly, with XGBoost + SMOTE slightly improving upon the XGBoost baseline. By contrast, the graph-based methods clearly outperformed the tabular approaches, with GraphSAGE achieving the strongest overall results and Embed-SMOTE also delivering very strong performance.

The class-specific metrics further confirm the advantage of graph-based approaches. GraphSAGE achieved the most favorable balance between fraud recall and fraud precision on the holdout set. Embed-SMOTE also exhibited strong minority-class performance, suggesting a slightly more recall-oriented decision profile. In contrast, both XGBoost variants displayed substantially lower fraud-class performance, particularly in terms of fraud F1-score, which remained consistently below that of graph-based models.

Table 5.2: Final test performance for all models.

Model	AUC-ROC	AUC-PR	Fraud F1	Fraud Recall	Fraud Precision
XGBoost baseline	0.7918	0.3630	0.3596	0.3237	0.4045
XGBoost + SMOTE	0.8005	0.3718	0.3669	0.3376	0.3977
GraphSAGE	**0.9894**	**0.9315**	**0.8395**	**0.9842**	**0.7318**
Embed-SMOTE	0.9724	0.8397	0.7944	0.9762	0.6697

The paired Wilcoxon tests (**Figure** 5.1 and **Figure** 5.2) showed that GraphSAGE significantly outperformed the XGBoost baseline across all the main metrics, with large paired effect sizes indicating that the observed differences were not only statistically significant but also practically substantial. Embed-SMOTE also improved from the baseline, although the corresponding effect sizes were smaller than those associated with GraphSAGE. For XGBoost + SMOTE, the gains were more limited.

Taken together, the cross-validation, holdout test, and inferential results consistently indicate that graph-based message passing provides the strongest predictive advantage for this fraud-detection task. Embed-SMOTE constitutes a substantial improvement over the tabular baseline, but GraphSAGE remains the most effective model overall. The XGBoost-based approaches, even when combined with SMOTE, were unable to match the performance achieved by graph-based models under the adopted temporal evaluation pipeline.

5.3 Comparison with prior work

Although the present study was not designed primarily as a direct benchmark against previous work, the results are broadly consistent with the literature on Ethereum fraud detection. Prior studies have shown that Transaction (Tx)-level and graph-based representations can be highly effective in blockchain security tasks, particularly when a structural or behavioral context is incorporated into the model. The strong performance of GraphSAGE in this dissertation aligns with the work of Lashkari *et al.* [40] and reinforces the view that relational modeling is especially valuable in decentralized financial settings. In their work, the proposed Advanced Genetic Algorithm outperformed the GraphSAGE as it obtained 0.95 in Recall, 0.96 in Precision, and 0.96 in F1-score in the final evaluation, while the tested Graph Neural Network (GNN) had comparable results as it obtained 0.83 in Recall, 0.93 in Precision, and 0.87 in F1-score.

5.4 Discussion

The results indicate that fraud detection in this Ethereum setting benefits substantially from modeling relational structure rather than relying exclusively on tabular Tx attributes. Across both the EW-CV and final holdout evaluation, graph-based models consistently outperformed

the XGBoost baselines, with the GraphSAGE model achieving the strongest overall performance. This pattern suggests that the predictive signal relevant to fraud is not fully captured by the engineered Tx, wallet features and attribute patterns alone, but also in the context of the surrounding neighborhood and the propagation of information throughout the graph.

The limited gains obtained by XGBoost + SMOTE further support this interpretation. Although oversampling improved some ranking metrics relative to the baseline, the improvement was modest and did not significantly improve the overall behavior of the classifier. This indicates that class imbalance, is not the principal limitation of the tabular approach in this task. Instead, the results suggest that the main bottleneck lies in the representational capacity of the input space itself. Synthetic resampling in the original feature space can help a classifier better accommodate the minority class, but it does not introduce the relational information that appears to be critical to distinguish fraudulent from legitimate Txs in this dataset.

GraphSAGE produced the best results across all reported metrics, which is consistent with the hypothesis that message passing can exploit local neighborhood dependencies that are invisible to independent tabular models. In this setting, the model likely benefits from aggregating information from surrounding wallet and Tx nodes, thereby learning patterns that reflect collective behavior rather than isolated records. The large margin by which GraphSAGE exceeded tabular baselines is also worth mentioning as it was observed not only in ranking metrics such as AUC-ROC and AUC-PR (**Figure 5.1**), but also in fraud-specific measures such as recall, precision, and F1-score (**Figure 5.2**). This combination indicates that the model is not only ranks Txs well in aggregate, but is also effectively detects fraudulent cases at the class level.

Embed-SMOTE also achieved strong performance, and its results are valuable for understanding the effect of imbalance handling in embedding space. Its improvement over the tabular baselines suggests that oversampling can remain beneficial when applied after a learned representation has been formed. However, its consistent underperformance relative to GraphSAGE implies that embedding-space augmentation is complementary rather than substitutive. The embedding-based synthetic samples may help the classifier better cover sparse regions of the minority class, but they do not fully replicate the expressive power of explicit neighborhood aggregation. This is an important finding because it suggests that, in this context and dataset, representation learning should be treated as the primary mechanism, while oversampling should be viewed as a secondary refinement.

By using EW-CV, the experiments preserved the chronological structure of the data and reduced the risk of leakage from future information into earlier folds. This is especially important in blockchain fraud detection, where Tx patterns can change over time and where models trained on randomly shuffled data can easily obtain overly optimistic estimates. The instability observed in the XGBoost folds, particularly the degradation in later temporal validation, suggests that the tabular baseline is more sensitive to distribution shift. In contrast, the graph-based models remained substantially more stable, which indicates a better temporal generalization under a realistic evaluation protocol. Running each model across ten random initializations makes it possible to distinguish robust effects from potentially lucky runs. The fact that GraphSAGE

remained superior under repeated seeds indicates that the improvement is not an artifact of a single favorable initialization. This shows that the observed gains are reproducible and not merely a consequence of variance in model training.

From a domain perspective, the findings are aligned with the structure of blockchain fraud itself. Fraudulent behavior in Ethereum-like Tx networks is rarely isolated and often appears through interactions between wallets, temporal activity patterns, and local network topology. A model that only inspects each Tx independently may therefore miss crucial relational clues. The results support the view that fraudulent entities can be better characterized through their graph neighborhoods, which may encode coordinated behavior, repeated interaction patterns, or structural anomalies. This helps explain why GraphSAGE, which is designed to learn from neighborhood aggregation, performed so strongly.

The comparison between GraphSAGE and Embed-SMOTE is also informative from a methodological point of view. GraphSAGE achieved the best balance between recall and precision, suggesting that it was able to identify fraudulent Txs while maintaining comparatively fewer false positives. Embed-SMOTE, by contrast, exhibited slightly more recall-oriented behavior. In a fraud detection setting, this trade-off matters as higher recall reduces the risk of missing fraudulent activity, but lower precision can increase the operational burden caused by false alarms. The fact that GraphSAGE remained superior in both ranking and fraud-class metrics suggests that structural learning provided a cleaner and more discriminative representation than synthetic latent augmentation alone.

Overall, the results support the idea that graph structure provides substantial additional information for Ethereum fraud detection beyond tabular features alone. The results also demonstrate that oversampling in the original feature space offers only limited benefit, whereas embedding-space augmentation is more promising but still secondary to explicit graph learning. Lastly, a temporally evaluation protocol reveals that graph-based models are not only more accurate, but also more stable under realistic distribution shifts. Taken together, these findings suggest that graph neural networks are a strong methodological choice for blockchain fraud detection, particularly when the goal is to capture relational behavior and maintain robustness over time.

Chapter 6

Conclusions

This chapter closes the dissertation by summarizing the main findings, stating the contributions of the work, and identifying its limitations and directions for future research. The central question motivating this study was whether graph-based representations provide a significant advantage over tabular models for Ethereum fraud detection under a temporally faithful evaluation protocol. The results, methodology, and their broader implications are reflected here to situate the work within the field and to identify what remains to be done.

6.1 Main findings

This work aimed to investigate Ethereum fraud detection through classical Machine Learning (ML) and Graph Neural Network (GNN)-based approaches, with a particular focus on class imbalance and temporally ordered evaluation. All models were assessed using Expanding-Window Cross-Validation (EW-CV) and a temporally separated final test set.

RQ 1: The GNN-based models outperformed both tabular baselines across the temporal validation experiments and final test set. Graph SAmple and aggreGatE (GraphSAGE) achieved the strongest overall results, while Embedding-Space SMOTE (Embed-SMOTE) also substantially outperformed the tabular models. These findings suggest that the graph representation captures relational information that is unavailable when Transaction (Tx)s are treated as independent tabular observations.

RQ 1.1: GNN-based models produced substantial gains in fraud-class detection. GraphSAGE achieved the strongest balance of fraud recall, precision, and F1-score, while also obtaining the highest Area Under the Precision-Recall Curve (AUC-PR). This indicates that relational message passing provided useful information for identifying fraudulent Txs beyond that captured by tabular features alone.

RQ 1.2: Embed-SMOTE performed strongly, indicating that embedding-space oversampling can be beneficial when combined with graph representation learning and class-weighted

training. However, it did not outperform standard GraphSAGE. eXtreme Gradient Boosting (XGBoost) with Synthetic Minority Over-sampling Technique (SMOTE) produced only modest improvements over the XGBoost baseline, suggesting that graph representation learning contributed more to performance than tabular oversampling alone.

Overall, within the scope of this case study, the findings indicate that graph-based representations are more effective than purely tabular feature-based models for Ethereum fraud detection under temporally ordered evaluation. In particular, GraphSAGE delivered the best holdout performance, while the embedding-based approach also improved substantially over the XGBoost baselines.

6.2 Contributions

The development of this dissertation contributes to the field of blockchain fraud detection as it formulates Ethereum fraud detection as a bipartite graph learning problem, where wallets and Txs are modeled as distinct node types connected by Tx edges. This design enables the model to capture structural dependencies between actors and Txs, going beyond traditional tabular representations. The work also introduces a temporally faithful evaluation framework based on EW-CV and an untouched holdout test set. By preserving chronological order and rebuilding each fold using only past information, the evaluation more closely reflects the conditions of real-world fraud detection and reduces temporal leakage.

This dissertation also contributes evidence that GNN are better suited than conventional tabular baselines for this task under class imbalance. In particular, the comparison between GraphSAGE, the Embed-SMOTE, and XGBoost with SMOTE shows that graph representation learning offers a more substantial benefit than oversampling alone.

6.3 Limitations and future work

Despite the strong empirical results obtained by the graph-based models, several limitations should be acknowledged. This study is based on a single Ethereum dataset, which limits the extent to which the findings can be generalized to other blockchain ecosystems, token standards, or fraud typologies. Although the temporal evaluation protocol increases realism, the results still reflect the characteristics of a specific data source and sampling strategy, which may not transfer directly to other chains or deployment environments. The graph construction process itself introduces methodological choices that may influence the final performance. Decisions about node representation, edge definition, feature aggregation, and fold-specific preprocessing can all affect the information available to the model. Although the inductive EW-CV design reduces leakage and aligns with a realistic temporal setting, it also means that changes in the graph structure across folds may compromise comparability and make the task more challenging in

later periods.

The class imbalance in the data remains substantial, particularly in the later temporal folds. Although the graph-based models handled this imbalance better than the tabular baselines, the rarity and temporal variability of fraudulent Txs can still affect model calibration and threshold selection.

Another limitation is interpretability. GNNs provide superior predictive performance, but their decisions are less transparent than those of simpler tabular models. This can make it difficult to explain why a specific Tx was flagged as suspicious, a concern relevant to financial and security applications. A model that performs well statistically may still be difficult to trust or deploy without additional explanation mechanisms.

Future work could address these limitations in several directions. A natural extension would be to evaluate the proposed framework in additional blockchain datasets, spanning different ecosystems and fraud scenarios. This would allow for a stronger assessment of generalization and robustness across domains. Another useful direction would be to investigate alternative graph constructions, such as temporal graph formulations that more explicitly model the evolution of wallet interactions over time.

It would also be valuable to explore more advanced imbalance-handling strategies in conjunction with graph learning. For example, embedding-space augmentation could be combined with class-aware sampling or cost-sensitive training to further improve performance for minority class. Since Embed-SMOTE already showed promising results, refining this approach may yield additional gains, particularly if synthetic samples are generated in a manner that better preserves graph neighborhood structure. Applying a full version of graphSMOTE with edge generator would also optimize the potential results. Such extensions would bring the work closer to practical applicability and provide a more complete picture of its usefulness in real-world fraud monitoring systems.

Bibliography

[1] Satoshi Nakamoto. Bitcoin p2p e-cash paper. *Cryptography Mailing list*, 03 2008. Accessed: 2026-01-28.

[2] Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform, 2013.

[3] Chainalysis. The 2025 crypto crime report: The rising role of cryptocurrency in all forms of crime and how its transparency is creating unique opportunities for investigation. Technical report, Chainalysis, 02 2025. Industry report.

[4] Cryptocom Research and Insights team. Crypto market sizing 2025, 01 2026. Accessed: 2026-02-02.

[5] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. Ethereum White Paper, 2014. Accessed: 2026-03-10.

[6] Federal Bureau of Investigation, Internet Crime Complaint Center. 2025 internet crime report. Annual report, Federal Bureau of Investigation, Washington, D.C., 2025.

[7] Hiroki Kanezashi, Toyotaro Suzumura, Xin Liu, and Takahiro Hirofuchi. Ethereum fraud detection with heterogeneous graph neural networks, 2022.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[9] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, page 833–841, New York, USA, 2021. Association for Computing Machinery. ISBN: 9781450382977. doi:10.1145/3437963.3441720.

[10] Shin Ando and Chun-Yuan Huang. Deep over-sampling framework for classifying imbalanced data, 2017.

[11] Jiajing Wu, Qi Yuan, Dan Lin, Wei You, Weili Chen, Chuan Chen, and Zibin Zheng. Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP:1–11, 09 2020. doi:10.1109/TSMC.2020.3016821.

[12] Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In *2018 World Wide Web Conference*, pages 1409–1418, 04 2018. doi:10.1145/3178876.3186046.

[13] Dirk Zetzsche, Ross Buckley, Douglas Arner, and Linus Föhr. The ico gold rush: It's a scam, it's a bubble, it's a super challenge for regulators. *SSRN Electronic Journal*, 01 2017. doi:10.2139/ssrn.3072298.

[14] Pengcheng Xia, Haoyu wang, Bingyu Gao, Weihang Su, Zhou Yu, Xiapu Luo, Chao Zhang, Xusheng Xiao, and Guoai Xu. Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange, 2021.

[15] Esteban Landerreche and Marc Stevens. On immutability of blockchains. In *Proceedings of 1st ERCIM Blockchain Workshop 2018*. European Society for Socially Embedded Technologies (EUSSET), 2018. doi:10.18420/blockchain2018_04.

[16] Daniel Davis Wood. Ethereum: A secure decentralised generalised transaction ledger. In *Semantic Scholar*, 2014.

[17] EU Blockchain Observatory and Forum. Ethereum Merge Trend Report. Trend report, EU Blockchain Observatory and Forum, 04 2023.

[18] Pedro María Preciado Martínez, Ricardo Francisco Reier Forradellas, Luis Miguel Garay Gallastegui, and Sergio Luis Náñez Alonso. Comparative analysis of machine learning models for the detection of fraudulent banking transactions. *Cogent Business & Management*, 12(1):2474209, 2025. doi:10.1080/23311975.2025.2474209.

[19] Soroor Motie and Bijan Raahemi. Financial fraud detection using graph neural networks: A systematic review. *Expert Systems with Applications*, 240:122156, 2024. ISSN: 0957-4174. doi:https://doi.org/10.1016/j.eswa.2023.122156.

[20] Tehreem Ashfaq, Rabiya Khalid, Adamu Sani Yahaya, Sheraz Aslam, Ahmad Taher Azar, Safa Alsafari, and Ibrahim A. Hameed. A machine learning and blockchain based efficient fraud detection mechanism. *Sensors*, 22(19), 2022. ISSN: 1424-8220. doi:10.3390/s22197162.

[21] Joao Crisostomo, Fernando Bacao, and Victor Lobo. Machine learning methods for fraud detection within ethereum blockchain—a review. *Blockchain: Research and Applications*, page 100469, 2026. ISSN: 2096-7209. doi:https://doi.org/10.1016/j.bcra.2026.100469.

[22] Rawan Ghnemat and Hatem Mosa. Blockchain-based fraud detection: A systematic review of ethereum network applications. *Cluster Computing*, 28(16), November 2025. ISSN: 1386-7857. doi:10.1007/s10586-025-05797-w.

[23] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN: 9781450342322. doi:10.1145/2939672.2939785.

[24] Huseyin Ahmetoglu and Resul Das. A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions. *Internet of Things*, 20: 100615, 2022. ISSN: 2542-6605. doi:https://doi.org/10.1016/j.iot.2022.100615.

[25] Sergiy Obushnyi, Denis Virovets, Andrii Ramskyi, Maksym Zhytar, and Pavlo Skladannyi. Variational autoencoders for detecting anomalous and fraudulent transactions in financial systems. In *Proceedings of the Workshop on Digital Economy Concepts and Technologies (DECaT 2025)*, volume 4029 of *CEUR Workshop Proceedings*, pages 110–118, 2025.

[26] Antonis Papasavva, Samantha Lundrigan, Ed Lowther, Shane Johnson, Enrico Mariconti, Anna Markovska, and Nilufer Tuptuk. Applications of ai-based models for online fraud detection and analysis. *Crime Science*, 14(1), 2025. ISSN: 2193-7680. doi:10.1186/s40163-025-00248-8.

[27] Huajie Huang, Bo Liu, Xiaoyu Xue, Jiuxin Cao, and Xinyi Chen. Imbalanced credit card fraud detection data: A solution based on hybrid neural network and clustering-based undersampling technique. *Applied Soft Computing*, 154:111368, 2024. ISSN: 1568-4946. doi:https://doi.org/10.1016/j.asoc.2024.111368.

[28] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics, 2019.

[29] Dawei Cheng, Yao Zou, Sheng Xiang, and Changjun Jiang. Graph neural networks for financial fraud detection: a review. *Frontiers of Computer Science*, 19(9), January 2025. ISSN: 2095-2236. doi:10.1007/s11704-024-40474-y.

[30] Hiroki Kanezashi, Toyotaro Suzumura, Xin Liu, and Takahiro Hirofuchi. Ethereum fraud detection with heterogeneous graph neural networks, 2022.

[31] Ahmad Asiri and K. Somasundaram. Graph convolution network for fraud detection in bitcoin transactions. *Scientific Reports*, 15(1):11076, 2025. doi:10.1038/s41598-025-95672-w.

[32] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[33] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[34] Collin Arnold Kabwama, Pius Businge, Curthbert Jeremiah Malingu, Jude Innocent Atuhaire, Ian Asiimwe Ankunda, Joram Gumption Ariho, Brian Mugalu, and Denis Musinguzi. Graph attention networks for credit card fraud detection: A relational learning approach. *World Journal of Advanced Research and Reviews*, 26(3):2580–2585, 2025. doi:10.30574/wjarr.2025.26.3.2400.

[35] Qiuwu Sha, Tengda Tang, Xinyu Du, Jie Liu, Yixian Wang, and Yuan Sheng. Detecting credit card fraud via heterogeneous graph neural networks with graph attention. *arXiv preprint*, arXiv:2504.08183, 2025. arXiv:2504.08183 [cs.LG]. doi:10.48550/arXiv.2504.08183.

[36] Yihong Ma, Yijun Tian, Nuno Moniz, and Nitesh V. Chawla. Class-imbalanced learning on graphs: A survey, 2023.

[37] Abdullah Alchihabi, Hao Yan, and Yuhong Guo. Overcoming class imbalance: Unified GNN learning with structural and semantic connectivity representations. In *New Perspectives in Graph Machine Learning*, 2025.

[38] Giuseppe Fenza, Salvatore Froncillo, Claudio Stanzione, et al. Enhancing fraud detection through cascading machine learning models and clustering techniques. In *ITASEC*, 2024.

[39] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3): 1–21, 03 2015. doi:10.1371/journal.pone.0118432.

[40] Arash Habibi Lashkari, Sepideh Hajihosseinkhani, Joshua Duarte, Isabella Lopez, Ziba Habibi Lashkari, and Sergio Rios-Aguilar. Advanced genetic algorithm and penalty fitness function for enhancing defi security and detecting ethereum fraud transactions. *Blockchain: Research and Applications*, page 100376, 2025. ISSN: 2096-7209. doi:https://doi.org/10.1016/j.bcra.2025.100376.

[41] A. M. Siam, P. Bhowmik, and M. P. Uddin. Hybrid feature selection framework for enhanced credit card fraud detection using machine learning models. *PLoS One*, 20(7):e0326975, jul 2025. doi:10.1371/journal.pone.0326975.

[42] NVIDIA Corporation. Financial fraud detection: Data preprocessing, dec 2025.

[43] Tanisha Mahajan, Gaurav Singh, and Glenn Bruns. An experimental assessment of treatments for cyclical data. In *2021 Computer Science Conference for CSU Undergraduates (CSCSU 2021)*. California State University, Monterey Bay, 2021. Accessed: 2026-03-02.

[44] Zhexian Gu and Omar Dib. Enhancing fraud detection in the ethereum blockchain using ensemble learning. *PeerJ Computer Science*, 11:e2716, 2025. doi:10.7717/peerj-cs.2716.

[45] Vítor Cerqueira, Luís Torgo, and Igor Mozetic. Evaluating time series forecasting models: An empirical study on performance estimation methods. *CoRR*, abs/1905.11744, 2019.

[46] Steven C. Hespeler, Pablo Moriano, Mingyan Li, and Samuel C. Hollifield. Temporal cross-validation impacts multivariate time series subsequence anomaly detection evaluation, 2025.

[47] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[48] Yumeng Tian. Financial fraud detection based on an xgboost and lstm fusion model: A comparative study on the enhancement of time-series features. *Advances in Economics, Management and Political Sciences*, 170:101–111, 2025. doi:10.54254/2754-1169/2025.LH23993.

[49] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. doi:10.48550/arXiv.1706.02216.

[50] Nicholas Williams and Alejandro Schuler. Improving reproducibility by controlling random seed stability in machine learning based estimation via bagging, 2026.

[51] B. Freidlin, W. Miao, and J.L. Gastwirth. On the use of the shapiro-wilk test in two-stage adaptive inference for paired data from moderate to very heavy tailed distributions. *Biometrical Journal*, 45(7):887–900, 2003. doi:https://doi.org/10.1002/bimj.200390056.

Appendix A

This appendix presents the feature descriptions for the BCCC-DeFiFraudTrans-2025 [3] dataset introduced by Lashkari *et al.* [40], together with the additional features engineered during the development of this project. The first table summarizes the Transaction (Tx)-level attributes, while the second table details the wallet-level attributes.

Table A.1: BCCC dataset Tx features

Category	Feature	Description
Address Related	length_from_address	The length of the "from" address in the Tx
	length_to_address	The length of the "to" address in the Tx
	from_is_same_as_to	Boolean indicating if the "from" and "to" addresses are the same
	from	Sender wallet address
	to	Receiver wallet address
	address_is_sender	Boolean indicating if the flagged address is the sender of funds
	address_is_receiver	Boolean indicating if the flagged address is the receiver of funds
Gas Related	gas_used	The amount of gas used for the Tx
	effective_gas_price	The price per unit of gas paid in the Tx
	total_gas_cost	The total cost of gas used (gas_used * effective_gas_price)
	gas_per_log_event	Gas used per log event in the Tx
	cumulative_gas_used	Total gas used in the block up to this Tx
	gas_price_ratio	Ratio of gas price to some baseline or previous price
	gas_efficiency	A measure of Tx efficiency based on gas usage
Log Related	log_count	Number of log entries in the Tx
	log_removed	Indicates whether any logs were removed
	log_index	Index of the log within the Tx
	length_log	Length of the log data
Block Related	value	The value transferred in the Tx
	index	Tx index in the block
	chain_id	Identifier of the blockchain network
	block_number	The block number in which the Tx is recorded
	coming_from_contract	Boolean indicating if the Tx originated from a contract
	length_transaction_hash	Length of the Tx hash
	hash	Unique cryptographic identifier used to trace and verify a blockchain Tx
Tx status Related	event_activity_flag	Flag indicating specific event activity
	message	Message associated with the Tx
	status	Status of the Tx (e.g., success, fail)
	Error!	Any errors returned by the Tx execution

Category	Feature	Description
Token Related	token_transfer_amount	Amount of tokens transferred in the Tx
	normalized_token_transfer	Normalized value of tokens transferred
Time Related	timestamp_UTC	The exact date and time when the Tx was recorded, expressed in UTC as YYYY/mm/dd hh:mm
	unix_timestamp	The Tx time converted into Unix time, measured as the number of seconds elapsed since 1 January 1970 UTC
	sin_hour	Sine transformation of the hour of the day, used to encode the cyclical nature of time
	cos_hour	Cosine transformation of the hour of the day, used to encode the cyclical nature of time

Table A.2: BCCC dataset Wallet features

Category	Feature	Description
Error Related	number_of_errors	Total number of errors encountered
	error_rate	Proportion of Txs that resulted in errors
Gas related	gas_used_summation	Sum of gas used across Txs
	gas_used_average	Average gas used per Tx
	gas_used_median	Median gas used per Tx
	gas_used_standard_deviation	Standard deviation of gas used
	gas_used_maximum_val	Maximum gas used in a single Tx
	gas_used_minimum_val	Minimum gas used across Txs
	gas_used_variance	Variance of gas used across Txs
	gas_used_range_value	Range between maximum and minimum gas used
	gas_used_skewness	Skewness of the gas used distribution
	gas_used_mode	Mode of the gas used across Txs
	gas_used_coefficient_of_variation	Coefficient of variation of gas used
	gas_prices_summation	Sum of gas prices across Txs
	gas_prices_average	Average gas prices per Tx
	gas_prices_median	Median gas prices per Tx
	gas_prices_standard_deviation	Standard deviation of gas prices
	gas_prices_maximum_val	Maximum gas price in a single Tx
	gas_prices_minimum_val	Minimum gas prices across Txs
	gas_prices_variance	Variance of gas prices across Txs
	gas_prices_range_value	Range between maximum and minimum gas prices
	gas_prices_skewness	Skewness of the gas prices distribution
	gas_prices_mode	Mode of the gas prices across Txs
	gas_prices_coefficient_of_variation	Coefficient of variation of gas prices
	cumulativeGasUsed_summation	Sum of cumulative gas used across Txs
	cumulativeGasUsed_average	Average cumulative gas used per Tx
	cumulativeGasUsed_median	Median cumulative gas used per Tx
	cumulativeGasUsed_std	Standard deviation of cumulative gas used
	cumulativeGasUsed_max_val	Maximum cumulative gas used in a single Tx
	cumulativeGasUsed_min_val	Minimum cumulative gas used across Txs
	cumulativeGasUsed_variance	Variance of cumulative gas used across Txs
	cumulativeGasUsed_range_value	Range between maximum and minimum cumulative gas used
	cumulativeGasUsed_skewness	Skewness of the cumulative gas used distribution

Category	Feature	Description
	cumulativeGasUsed_mode	Mode of the cumulative gas used across Txs
	cumulativeGasUsed_cov	Coefficient of variation of cumulative gas used
Values Related	values_summation	Sum of values transferred across Txs
	values_average	Average value transferred per Tx
	values_median	Median value transferred per Tx
	values_standard_deviation	Standard deviation of values transferred
	values_maximum_val	Maximum value transferred in a single Tx
	values_minimum_val	Minimum value transferred across Txs
	values_variance	Variance of values transferred across Txs
	values_range_value	Range between maximum and minimum values transferred
	values_skewness	Skewness of the values transferred distribution
	values_mode	Mode of the values transferred across Txs
	values_coefficient_of_variation	Coefficient of variation of values transferred
Nonce related	nonce_summation	Sum of nonce values used across Txs
	nonce_average	Average nonce used per Tx
	nonce_median	Median nonce used per Tx
	nonce_standard_deviation	Standard deviation of nonce used
	nonce_maximum_val	Maximum nonce used in a single Tx
	nonce_minimum_val	Minimum nonce used across Txs
	nonce_variance	Variance of nonce used across Txs
	nonce_range_value	Range between maximum and minimum nonce used
	nonce_skewness	Skewness of the nonce used distribution
	nonce_mode	Mode of the nonce used across Txs
	nonce_cov	Coefficient of variation of nonce used
Count related	number_of_from_address	Total number of “from” addresses used
	number_of_unique_from_address	Number of unique “from” addresses
	number_of_to_address	Total number of “to” addresses involved
	number_of_unique_to_address	Number of unique “to” addresses
	num_transaction	Total number of Txs
	duration	Time duration from the first to the last Tx