RESEARCH CENTRE FOR
INFORMATION SYSTEMS
ENGINEERING (LIRIS)

Experimental Analysis of Graph Neural Networks for Credit Card Fraud Detection

Ruben Boeckx
r0792465

Thesis submitted to obtain the degree of

Master of Information Management

Promoter: Prof. Dr. Wouter Verbeke
Daily Supervisor: Bruno Deprez

Academic year: 2023-2024

Abstract. As financial fraud continues to evolve, there is an interest in novel detection techniques that can leverage the inherent network structure of financial transactions. Traditional machine learning models, while effective, may not fully capture the intricate relationships between entities in a transaction network. Graph neural networks, with their ability to process and learn from graph-structured data, offer a promising approach to address this limitation. This master's thesis conducts an experimental analysis of graph neural networks for credit card fraud detection, exploring their potential to enhance current detection methods. Using a dataset of 1.3 million credit card transactions, we construct a heterogeneous graph and implement two graph neural network architectures - Graph Attention Network and GraphSAGE. These are benchmarked against industry standard models including Random Forest and Logistic Regression. By comparing the performance of these models, we aim to assess whether graph-based approaches can match or surpass traditional methods in detecting fraudulent transactions. This thesis contributes to the growing body of research on applying graph-based methods to financial fraud detection.

Keywords: Graph Neural Networks · Fraud Detection · Network Analytics · Node Classification · Heterogeneous Graph Learning

Acknowledgement. Writing a master's thesis is a significant challenge that requires dedication, perseverance, and support from various quarters. As I reflect on this demanding journey, I am filled with gratitude for all those who have stood by me, offering their assistance and encouragement throughout the past year.

First and foremost, I would like to thank Wouter Verbeke and Bruno Deprez. I am grateful to Professor Verbeke for introducing me to the interesting yet challenging topic of graph neural networks. Equally deserving of my gratitude is Bruno Deprez, the daily supervisor of this master's thesis. His support and availability have been crucial to the progress of this work. Bruno's prompt responses to my questions, his constructive feedback, and his willingness to arrange meetings whenever necessary have been a constant source of motivation and clarity.

My heartfelt thanks go to my friends, who have been a source of support and a welcome distraction during the more challenging periods of this journey.

Last but certainly not least, I owe an immeasurable debt of gratitude to my family, particularly my father, who has supported me throughout my entire academic journey.

Table of Contents

List of Abbreviations and Symbols

Abbreviations

ANN	Artificial Neural Network
AUC	Area Under Curve
CNN	Convolutional Neural Network
ECB	European Central Bank
FPR	False Positive Rate
GAT	Graph Attention Network
GCN	Graph Convolutional Neural Network
GNN	Graph Neural Network
ML	Machine Learning
PRC	Precision-Recall Curve
ReLU	Rectified Linear Unit
RNN	Recursive Neural Network
ROC	Receiver Operating Characteristic Curve
SEPA	Single Euro Payments Area
SMOTE	Synthetic Minority Over-Sampling Technique
SNA	Social Network Analysis
STAGN	Spatial-Temporal Attention-Based Graph Network
TPR	True Positive Rate

1 Introduction

Payment card fraud remains a pervasive global problem. In 2021, the total value of transactions using payment cards issued in SEPA amounted to €5.40 trillion, with fraudulent activity totaling €1.53 billion, accounting for 0.028% of the total [22]. It is important to note that this figure pertains solely to the number of fraudulent transactions that have been uncovered, suggesting that the actual percentage could be even higher. Recently, there has been encouraging progress regarding this issue. According to the most recent publication of the ECB on payment card fraud [22], the incidence of card fraud experienced a significant decrease in 2021, with a notable 11.2% drop compared to the previous year. As a result, the proportion of fraudulent transactions in relation to the total value of transactions also declined significantly in 2021, reaching 0.028% (down from 0.036% in 2020). This represents the lowest recorded percentage of fraud since data collection commenced in 2008. While these statistics indicate a positive trend, it is crucial to interpret them cautiously. The decrease in fraud rates could be attributed to various factors, including improved security measures, increased awareness among consumers, or potentially, more sophisticated fraudulent activities that evade detection. Despite this progress, the absolute value of fraudulent transactions remains substantial at €1.53 billion. This underscores the ongoing need for vigilance and continuous improvement in fraud detection methods. As fraudsters adapt their techniques [6], it remains crucial to develop and refine robust fraud detection models to identify and prevent as many fraudulent transactions as possible.

Fraud detection methods have evolved significantly over time, driven by the increasing complexity of fraudulent activities and the growing volume of transactions. Historically, the classic approach to fraud detection was expert-based, relying on the experience, intuition, and business knowledge of fraud analysts [6]. This method, while valuable for its insights, has significant limitations in the modern financial landscape. The traditional expert-based approach proves time-consuming and inefficient when faced with the massive volume of transactions occurring in today's digital economy. Manual detection

methods simply cannot keep pace with the speed and scale of modern financial transactions, creating a need for more automated and sophisticated solutions. This need has been met by the rise of machine learning in fraud detection. The dramatic increase in transaction volumes has both necessitated and facilitated the implementation of machine learning algorithms, thanks to the abundance of available data. Financial institutions have increasingly turned to these advanced techniques to detect fraudulent transactions more efficiently and accurately. Machine learning methods offer several advantages for card fraud detection:

- Ability to analyze large volumes of transaction data quickly;
- Capacity to identify complex patterns that might be imperceptible to human analysts;
- Adaptability to new fraud patterns through continuous learning.

Some commonly used machine learning techniques in the industry include neural networks, decision trees, and logistic regression [12]. While machine learning has revolutionized fraud detection, it is important to note that the expertise of human analysts remains crucial. The most effective fraud detection systems often combine machine learning algorithms with human oversight, leveraging the strengths of both automated analysis and expert judgment [6].

Recent research has explored the application of social network analysis to fraud detection, since fraud activities often involve a social/group dynamic [58,6]. Criminal networks often exhibit interconnectedness among fraudsters, who frequently engage in similar illicit activities, utilize common resources, or even adopt shared identities through identity theft. This interconnectedness stems from fraudsters operating within shared contexts, leading to a higher degree of network connectivity among them [59]. This phenomenon aligns with the sociological concept of homophily, often summarized by the phrase 'birds of a feather flock together.' Homophily describes the tendency of individuals to associate with others who share perceived similarities [47]. The network effect resulting from homophily enables the application of 'guilt-by-association' techniques in fraud detection. These methods infer a node's classification based on the

labels of closely connected nodes within the network [39]. In fraud detection scenarios, a network displaying strong homophily indicates that fraudulent nodes have a higher likelihood of being connected to other fraudulent nodes. Similarly, legitimate nodes are more likely to be connected to other legitimate nodes. This pattern enhances the effectiveness of network-based fraud detection methods.

This thesis focuses on developing two distinct graph neural network architectures that can operate on heterogeneous graphs and two common industry baseline models, all aimed at flagging transactions as fraudulent or non-fraudulent. Our goal is to determine if these network learning approaches can match or surpass the performance of conventional credit card fraud detection models.

We begin with a literature review, examining current machine learning techniques in the field and their evaluation metrics used. We will define graph neural networks, explain their functionality, and describe the architectures we will implement. The review will also address common challenges in developing credit card fraud detection models, informing our approach throughout the research. Chapter 3 outlines our experimental methodology. We will describe the dataset used, highlight key preprocessing steps, and explain the graph construction process. This chapter will also cover model specifications, hyperparameter tuning, and evaluation metrics.

Following this, we will present our results on the test set in Chapter 4 and provide a discussion of our findings. In Chapter 5 we outline the potential limitations of our experimental setup. The thesis concludes with a summary of our experimental analysis, discussing the implications of our results for credit card fraud detection and suggesting potential areas for future research.

The Python code developed for this experimental analysis, including data preprocessing, graph construction, and visualization scripts, is publicly available in a GitHub repository[1] for transparency and reproducibility.

[1] https://github.com/Ruben-Boeckx/Experimental-Analysis-of-GNNs-for-CCF

2 Literature Review

Chapter 2 begins with an exploration of current machine learning techniques employed in credit card fraud detection. We then define graph neural networks, describe their core working mechanisms, and briefly explain the various GNN types to be used in our experimental analysis. Additionally, we examine previous applications of GNNs in the broader field of fraud detection. The chapter concludes by addressing the challenges commonly encountered in credit card fraud detection.

2.1 Machine learning for credit card fraud detection

The application of machine learning to credit card fraud detection and prevention has seen a surge in research over recent decades [8,27,50]. This critical domain of financial security has witnessed the exploration, implementation, and rigorous testing of a wide spectrum of algorithms and methodologies, each aiming to bolster the efficacy of fraud detection systems [3].

Studies conducted by Mittal and Tyagi [45], Ali et al. [3], and Minastireanu and Mesnita [43] have examined the most effective and used machine learning algorithms for identifying credit card fraud. They consistently demonstrate a preference for supervised machine learning techniques in addressing credit card fraud, likely due to these algorithms' ability to effectively learn and discern patterns that distinguish legitimate transactions from fraudulent ones.

While unsupervised methods have their merits, supervised learning's capacity to extract insights from historical data with known outcomes often proves more effective in the context of fraud detection [46].

Among the reviewed studies, artificial neural networks, support vector machines, random forest, logistic regression, and naïve Bayes emerged as the most commonly utilized machine learning techniques in the industry. Support Vector Machines and Artificial Neural Networks are highlighted as the most prevalent machine learning algorithms for fraud detection. Ali et al. [3] specifically notes that credit

card fraud emerges as the primary type of financial fraud effectively addressed through machine learning methodologies. This likely stems from the vast amounts of data generated by credit card transactions, which these algorithms can efficiently analyze to identify fraudulent patterns.

Evaluating the effectiveness of the employed models is another crucial component for building successful detective models. Yet there is a notable absence of standardized assessment criteria specifically tailored for machine learning methodologies in this domain [30]. Mittal and Tyagi [45] and Ali et al. [3] explore various performance metrics used to gauge the efficacy of different machine learning approaches. Mittal and Tyagi [45] highlights that traditional metrics such as overall accuracy and the standard confusion matrix may not adequately capture the fraud detection rate due to the inherent class imbalance in fraud datasets. They advocate for metrics that give equal weight to both classes. To address these challenges, researchers commonly employ a range of metrics including: balanced accuracy, precision, F1-score and recall. These metrics provide a more comprehensive view of model performance, especially in imbalanced datasets typical of fraud detection. The importance of choosing appropriate evaluation metrics will be revisited in greater detail in Subsection 3.6.

However, it is interesting to note that none of these literature reviews did include the use of graph neural networks. The use of artificial neural networks was addressed but still, it is crucial to point out that graph neural networks possess distinct functionalities. Unlike graph neural networks, artificial neural networks do not examine the graph's structure in order to unveil concealed patterns indicative of fraudulent activities.

2.2 Graph Neural Network

Graph Neural Networks are a significant breakthrough in the field of deep learning, resulting from the combination of classical graph theory and modern neural networks. Due to this innovative combination, GNNs are utilized across various fields ranging from social network analysis [62] and molecular chemistry [18,20] to recommendation systems [7,23].

The graph neural network approach was first proposed by Scarselli et al. [53]. They noted that traditional methods often relied on manually extracting features from graph metrics such as degree, PageRank, or betweenness centrality. However, these approaches frequently failed to capture all relevant structural information within the graph. GNNs were developed to address this limitation by incorporating more comprehensive structural data. Scarselli et al. [53] based their GNN model on an extension of recursive neural networks. Their framework posits that each node in a graph possesses an internal state, which is determined by a combination of the node's own attributes, the properties of its neighboring nodes, and the states of those neighbors. The output for any given node is then computed as a function of its internal state and inherent properties. The process of determining node states in this model involves iterative updates, reminiscent of the methodology employed in RNNs. This iterative approach allows the GNN to propagate information throughout the graph structure effectively

2.2.1 Graph

Graph neural networks are designed to process and analyze data structured as graphs. A graph or network $G = (V, E)$ consists of a set of nodes V and a set of edges E connecting the nodes. An edge is typically defined by its incident nodes; for example, the edge between nodes $i, j \in V$ is denoted as $e(i, j) \in E$. In directed graphs, the edge direction is important, whereas in undirected graphs, edges have no inherent direction, implying $e(i, j) \in E \Leftrightarrow e(j, i) \in E$. A common example is a social network, where nodes represent individuals and edges represent social connections such as friendships, communications, or collaborations [47]. GNNs leverage the intrinsic

connectivity and features of these graphs to perform various tasks, including node classification, link prediction, and graph classification [53,29].

We elaborate further on two important network variants: homogeneous graphs and heterogeneous graphs.

2.2.1.1 Homogeneous graph
A homogeneous graph $G = (V, E)$ represents the standard concept of a graph with a single node type. The set V contains all nodes in the graph, while the edge set $E \subseteq V \times V$ defines connections between these nodes. Every node $v \in V$ is of the same type, which simplifies the graph structure and analysis. This formulation is suitable for modeling systems where all entities are of the same nature, such as social networks where nodes represent individuals.

2.2.1.2 Heterogeneous graph
A heterogeneous graph $G = (V, E, Q, \phi)$ extends the concept of a standard graph by introducing different node types. Every node $v \in V$ has exactly one node type $q \in Q$, where Q is the set of all possible node types. The mapping function $\phi : V \to Q$ links each node to its associated node type. This formulation allows for more complex relationships and richer representations in graph-structured data, enabling models to capture type-specific interactions and features. Unlike homogeneous graphs, heterogeneous graphs can represent diverse entities and their relationships within a single structure.

2.2.2 Core working mechanism of a GNN

Message passing serves as the core mechanism of graph neural networks, operating through an iterative process where nodes aggregate information from their local neighborhoods. As iterations progress, each node's embedding captures information from increasingly wider areas of the graph. After k iterations, a node's embedding contains information from its k-hop neighborhood, encompassing data from all nodes reachable within k steps [29].

The information captured in these embeddings is twofold. First, there is structural information about the graph topology, such as the

degrees of nodes in the neighborhood, centrality measures, and local clustering coefficients [29]. This structural information can reveal important patterns in the graph's architecture. Second, the embeddings encode feature-based information, aggregating and combining the features of neighboring nodes [29]. These features could be any attributes associated with the nodes or edges. The GNN learns to combine these features in meaningful ways, potentially uncovering complex relationships and interactions within the graph. This dual encoding of structural and feature-based information allows GNNs to capture rich, multi-faceted representations of graph elements, enabling them to perform well on a wide range of graph-based machine learning tasks.

This local feature-aggregation process is analogous to how convolutional neural networks operate on images, where CNNs aggregate information from spatially-defined patches. In contrast, GNNs aggregate information based on graph-defined neighborhoods. This methodology enables GNNs to learn representations that effectively capture both local and broader graph structures and features, thereby making them potent tools for analyzing and learning from graph-structured data.

2.2.3 Types of GNNs

Over the past years several GNNs have gained prominence. The Graph Convolutional Network is a pioneer, enabling the fusion of local node information and its neighbors [36]. GraphSAGE [28], on the other hand, employs a sampling strategy to generate embeddings by sampling a fixed-size neighborhood around each node. More recently, the Graph Attention Network [60] has introduced attention mechanisms to weigh neighbor contributions, allowing for a more dynamic and adaptive aggregation of information.

In the following subsections, we will briefly overview the two different graph neural networks used in our experimental analysis.

2.2.3.1 GraphSAGE

GraphSAGE (Sample and Aggregate) is a prominent type of graph neural network introduced by Hamilton et al. [28]. Similar to other GNN architectures like Graph Convolutional Networks and Graph Attention Networks, GraphSAGE uses inductive learning, allowing it to generalize its learned representations to new, unseen nodes [28]. The primary innovation of GraphSAGE is its neighborhood sampling approach. Unlike GCNs, which utilize the entire set of neighbors for each node, GraphSAGE uniformly samples a fixed-size set of neighbors to aggregate information [65]. This sampling strategy provides significant advantages, particularly in scalability and generalizability. By limiting the number of neighbors, GraphSAGE reduces computational complexity, enhancing its efficiency for large-scale graphs. Furthermore, this method enables GraphSAGE to handle dynamic graphs and adapt to unseen nodes more effectively, increasing its applicability in real-world scenarios where graph structures may continuously change.

The sampling process in GraphSAGE is designed to select neighbors that have a high similarity to the target node, ensuring that the sampled neighbors can effectively represent and enrich the information of the current node [42]. Specifically, for each node v, a fixed-size set of neighbors $\mathcal{N}(v)$ is uniformly sampled. Mathematically, this can be represented as:

$$\mathcal{N}(v) = \text{Sample}(\{u \mid (v, u) \in E\}, K)$$

where K is the number of neighbors to sample, and E represents the edges in the graph.

After sampling, GraphSAGE aggregates the information from the selected neighbors using a chosen aggregation function. Let $h_u^{(k)}$ represent the embedding of neighbor u at layer k. The aggregation can be expressed as:

$$h_{\mathcal{N}(v)}^{(k)} = \text{Aggregate}(\{h_u^{(k-1)} \mid u \in \mathcal{N}(v)\})$$

where Aggregate can be a mean, LSTM, or pooling function. This aggregated information is then combined with the target node's own

features to generate its embedding. This combination can be written as:

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

where σ is a non-linear activation function, $W^{(k)}$ is a learnable weight matrix for layer k, and Concat represents the concatenation operation. This process is performed iteratively, with each layer of the network capturing information from a wider neighborhood. These steps enable GraphSAGE to generate meaningful node embeddings that can be used for various downstream tasks such as node classification, link prediction, and clustering [28].

2.2.3.2 Graph Attention Network

Graph Attention Networks [60] are a type of neural network architecture designed to work with graph-structured data. Unlike previous methods that relied on graph convolutions, GATs leverage a concept from transformers called masked self-attention. This allows GATs to focus on specific features of neighboring nodes, giving them a more nuanced understanding of the local graph structure. The authors argue that this attention mechanism allows the network to implicitly assign different weights to neighboring nodes, even within neighborhoods of varying sizes, without the need for expensive matrix operations [60]. This makes GATs computationally efficient and well-suited for large graphs.

This attention mechanism offers several advantages over traditional methods. GATs can implicitly assign different weights to neighboring nodes within a single attention layer, eliminating the need for separate weighting schemes or expensive matrix operations [60]. Specifically, the attention coefficients α_{ij} between node i and its neighbor j are computed as:

$$e(h_i, h_j) = \text{LeakyReLU}\left(a^T \cdot [Wh_i \| Wh_j]\right)$$

where $e(h_i, h_j)$ represents the unnormalized attention score, $W \in R^{d' \times d}$ is a weight matrix applied to the node features, $a \in R^{2d'}$ is the attention mechanism's learnable weight vector, and $\|$ denotes concatenation. The Leaky ReLU activation function is applied to

introduce non-linearity. The normalized attention coefficients α_{ij} are then obtained using the softmax function:

$$\alpha_{ij} = \text{softmax}_j(e(h_i, h_j)) = \frac{\exp(e(h_i, h_j))}{\sum_{k \in \mathcal{N}(i)} \exp(e(h_i, h_k))}$$

where $\mathcal{N}(i)$ represents the neighborhood of node i. The node features are then updated by computing a weighted average of the transformed features of the neighbors, scaled by the attention coefficients:

$$h_i' = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \cdot W h_j \right)$$

where σ is a non-linear activation function such as ReLU.

GATs are also well-suited for inductive learning tasks, where the model encounters unseen graphs during testing, as they do not require upfront knowledge of the entire graph structure [60]. This is a significant advantage over spectral methods, which often struggle with unseen graph instances. These steps enable GATs to effectively learn representations from graph-structured data while handling varying neighborhood sizes and focusing on relevant node features.

2.2.4 Fraud detection using GNNs

There has been a rise in research focusing on the use cases of GNNs for fraud detection [35,13,42]. This growing interest can be attributed to GNNs' ability to directly extract features from neighboring nodes in graph-structured datasets. This characteristic makes GNNs particularly compelling for fraud detection, as many domains naturally lend themselves to graph-structured data representation.

In the domain of credit card fraud, Cheng et al. [13] developed a GNN method for detecting fraudulent transactions in a social network setting. They propose a spatial-temporal attention-based graph network for detecting fraudulent transactions. This is a specific type of graph neural network with an attention mechanism. Their research illustrates how the detection model is constructed and trained, followed by extensive experiments conducted on a real-word card transaction dataset. The results of these experiments show that the STAGN performs better than the other baselines that are adopted in the industry. It demonstrates its superiority in detecting suspicious transactions, mining spatial and temporal fraud hotspots, and uncover fraud patterns.

Jiang et al. [35] applied GNNs in the field of fraud detection. The authors designed a GCN based anomaly scheme for insider threat detection and fraud detection. The behaviour of users and connection relationships are translated into a graph and are then used to train the robust anomaly detection model using the GCN-algorithm. However, when converting user interactions into a graph structure, many isolated nodes emerge due to infrequent communication between users. Consequently, constructing an adjacency matrix for the GCN model based solely on direct user connections fails to capture the broader structural information of the network. That is why they designed a weighted function, which leverages users' connection relationships and the similarity of their behaviors to quantify the structural information of the network. The result shows that the updated adjacency matrix input could primarily improve the detection accuracy for insider threat detection compared to the baselines [35].

Liu et al. [42] proposed a new model that could comprehensively characterize not only the relationship of transactions but also the relationship of original transaction features and the dynamic changes of behaviors of users. To include all those relations, they designed a weighted multiple graph called Transaction Graph. Based on the Transaction Graph, GraphSAGE, together with an additional added attention mechanism, is used to train a detection model for credit card fraud [42]. They compared the results of their model to many baselines such as linear regression, GCN and also a normal GraphSAGE. The results of their model were very promising as they outperformed both GCN and GraphSAGE. This implies that their model has the capability to extract additional valuable data from the adjacent nodes [42].

GNNs have a wide range of applications beyond credit card fraud detection. They are versatile tools that can be utilized in various domains, such as for developing credit ratings [24]. Additionally, GNNs are effective in detecting fraud in industries like insurance and e-commerce [3]. In the e-commerce sector, GNN techniques are particularly valuable because existing fraud detection systems often suffer from performance deterioration and struggle to adapt to evolving fraud patterns. This limitation occurs because these systems typically rely on previously identified fraudulent activities as reference points for spotting other suspicious behaviors [64]. According to Zhang et al. [64], GNN methods outperform other industry-standard techniques, demonstrating superior performance in fraud detection in the e-commerce field.

We can see that previous research shows a strong tendency to develop novel implementations rather than relying on standard GNN methods. The examined authors often strive to enhance existing models to achieve superior performance across various fields. The reviewed studies indicate that GNN applications for classification problems in different industries generally yield better results compared to industry-specific baseline models.

2.3 Challenges in credit card fraud detection

2.3.1 Imbalanced dataset

In card fraud detection, datasets often exhibit class imbalance, with fraudulent transactions being the minority [22]. This poses challenges for machine learning methods to correctly classify non-fraudulent transactions while maintaining good performance for the minority class [44]. Resampling methods are used to balance datasets, aiming to improve minority class classification [21]. Resampling should only be applied to training data, as altering test and validation sets introduces bias [6].

Kotsiantis et al. [37] discusses several methods to balance a class distribution. The most commonly used techniques are undersampling, oversampling and SMOTE. Undersampling eliminates instances of the majority class while retaining all minority instances, either randomly or in a directed manner to minimize information loss. Oversampling increases minority class instances, typically through duplication. The Synthetic Minority Over-Sampling Technique (SMOTE) goes further by generating new synthetic samples using k-nearest neighbors. SMOTE selects examples from the k nearest neighbors of each minority example and creates new synthetic examples along the line between the minority example and its selected neighbors.

2.3.2 Time-evolving behavior of fraudsters and customers

Fraudsters continually adapt their methods, requiring fraud-detection systems to evolve correspondingly [19,38]. Despite the ongoing nature of this challenge, organizations can effectively reduce losses by investing in advanced detection and prevention mechanisms [6]. Not only does the behavior of fraudsters change over time, but the behavior of customers can change as well. For example when people spend more on holidays than in their daily life. This is what we call data drift. Data drift refers to the phenomenon where the input data change over time which can have a deteriorating effect on the accuracy of the model [9]. This change can be caused by various factors such as changes in the relationship between different variables, changes in the distribution of the target variable or changes in the distribution of the input features. These changes are respec-

tively called concept drift, prevalence drift and covariate drift [48]. All these changes lead to more variation in the model, which negatively impacts the performance. The models become less accurate and reliable when applied to new data [14].

2.3.3 Large scale data

Major credit institutions annually manage millions of transactions involving millions of clients and partners [19]. Credit card fraud detection datasets feature numerous instances and attributes [52], with an average of 25 attributes per card [1]. This high dimensionality allows diverse analyses but risks distraction from low-correlation attributes. The massive data scale also challenges processing times, especially with complex ML models [40].

Various ways exist to address these challenges, but unfortunately, they often come with a cost. One option could be to simply invest in higher processing power through the acquisition of faster processing chips or overclocking available processing chips, although this first option could quickly become expensive depending on the application. Alternatively, there exist ways to reduce the size of the data, resulting in lower runtimes [26]. However, data reduction techniques often carry the cost of having to work with lower-quality data, which should be balanced against the gain of having lower processing times.

2.3.4 The cost of fraud detection

The cost of misclassification is crucial in fraud detection evaluation [55]. The cost arises when a model tags a genuine transaction as fraudulent (false positive) or when the system fails to detect a fraudulent transaction (false negative) [1,63]. Certain sectors within fraud detection see considerable benefits in the development of cost-sensitive approaches that take these misclassification costs into account. While traditional fraud detection maximizes statistical measures like precision-recall, AUC, and F1-score [55,10], cost-sensitive classifiers aim to maximize financial gain [33,57].

As mentioned in Subsection 2.3.3, the scale of data used to predict fraud carries financial implications for those involved in the predic-

tion process. The costs of processing power and excessive processing times go hand in hand, and are usually quantifiable in monetary terms. Real-time detection is crucial as delayed detection may negate fraud detection gains [1,4]. Reducing processing times may require investments in faster hardware or cloud infrastructure.

Development costs are also significant. Fraud detection models require tailoring, thorough study, regular updates, and maintenance [49]. Various implementation costs must be considered in the overall equation of fraud detection systems.

3 Methodology

3.1 Dataset

For this experimental analysis, a credit card transactions dataset from Kaggle[2] is used. This is a simulated dataset containing legitimate and fraudulent transactions recorded between 01/01/2019 and 31/12/2020. The dataset consists of 1 296 675 unique transactions, of which only 0.58% are fraudulent, highlighting the significant class imbalance. Furthermore, the dataset exists out of 983 unique credit card users, and 693 unique merchants. This means that the dataset consists of a multitude of transactions from different credit card users with a lot of different, and potentially the same merchants. It is crucial to understand that credit card users solely engage in transactions with merchants, and not with each other. Each row in the dataset represents a unique transaction between a credit card user and a merchant.

Each transaction contains 22 attributes to describe it. These attributes provide information about the date of the transaction, the category in which the transaction falls (gas, groceries, ...), the coordinates of the credit card user and the merchant, and much more. These attributes will be assigned to their corresponding node during the construction of the graph and used as features during the node classification task.

3.2 Data preprocessing

The dataset has already undergone preprocessing up to a point where there are no missing or duplicate values present. This has facilitated the preprocessing process. Our primary tasks were to conduct feature engineering and address the significant class imbalance.

To get started, we assigned a distinct identifier number to each individual occurrence of a node type. This action would facilitate the identification of the different nodes for constructing the graph. As mentioned in Subsection 3.1, this resulted in the generation of 983

[2] https://www.kaggle.com/datasets/kartik2112/fraud-detection

unique identifier numbers for the credit card users, 693 unique identifier numbers for the various merchants, and 1 296 675 unique identifier numbers for the transactions.

Besides this important step we performed some basic feature engineering to create more meaningful inputs. We applied a logarithmic transformation on the transaction amount since these values can vary widely, with some being very small and others very large. Taking the logarithm can help normalize the data, bringing extreme values closer to the mean and reducing the range of values. Furthermore, we decided to derive the hour, weekday, month, and day of the month from the transaction date attribute. Understanding the typical timing of a cardholder's transactions, including hour, weekday, month, and day of month, empowers machine learning models to identify fraudulent activities that deviate from these established spending patterns. To finalize the preprocessing stage, we generated the age for each credit card user, and applied count encoding on the categorical variables since the graph neural network requires numeric inputs. It should be noted that certain features were excluded from the initial 22 features per transaction due to their lack of significance or potential for overfitting.

A final essential action we took was to oversample the minority class and undersample the majority class in the training set. This approach ensures that the graph neural network recognizes the importance of both classes during the training phase. We performed the train-test-split at the level of both credit card users and merchants, this means that the training set doesn't include credit card users that are also in the test set and vice versa. The same holds for the merchants. By keeping credit card users and merchants exclusive to either the training or test set, we effectively avoid any overlap that could lead to the model learning specific details rather than general patterns. This separation is critical as it ensures that the model's performance metrics are reflective of its ability to handle completely unseen data, providing a more accurate assessment of its generalization capabilities.

3.3 Transaction network

Following the approach proposed by Van Belle et al. [58], we will implement a tripartite network design comprising three distinct node types: credit card users, merchants, and transactions. This network design belongs to the category of heterogeneous graphs, as discussed in Subsection 2.2.1.2. This structure allows us to generate embeddings for each node type while maintaining a sparse network connectivity. The network will feature undirected and unweighted edges. In this configuration, credit card users serve as source nodes, transactions act as bridge nodes, and merchants function as destination nodes.

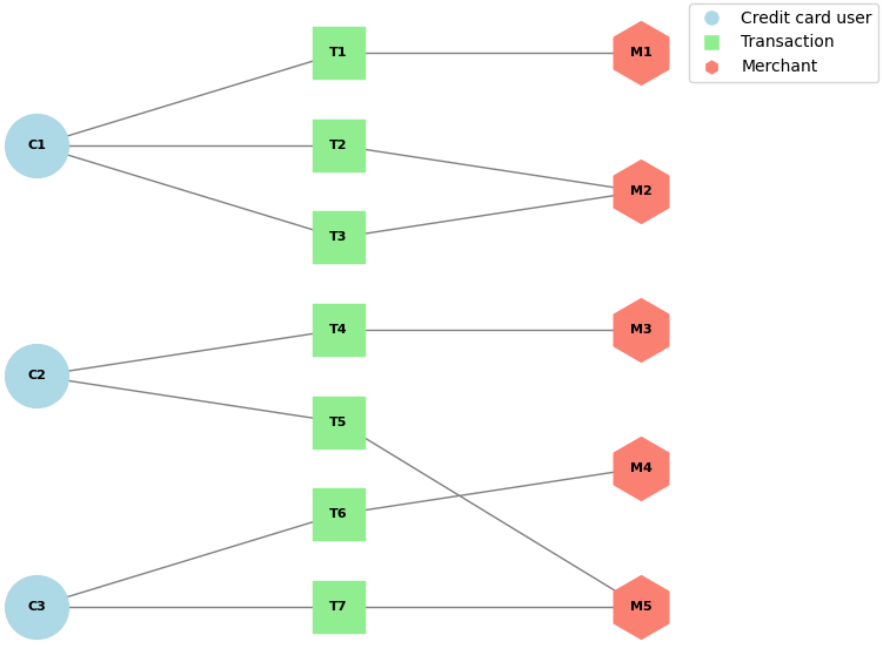

Fig. 1. Subgraph of transaction network showing relationships between users, transactions and merchants

Figure 1 illustrates a subgraph of the transaction network, comprising 3 credit card users, 7 transactions, and 5 merchants. This visualization underscores the network's structure: credit card users

interact exclusively with merchants through transactions, not with other credit card users. Each credit card user can engage in transactions with any number of merchants, and similarly, each merchant can transact with any number of credit card users. Importantly, the network maintains strict separation between node types, with credit card users and merchants only connecting to transactions, never directly to entities of their own type.

3.4 Model specifications

The GNN architectures previously discussed in Subsection 2.2.3 are all developed using the PyTorch Geometric Python library [25]. This powerful library, specifically designed for geometric deep learning on graphs and other irregular structures, provided a robust framework for our implementations.

When developing the GNNs, we prioritized flexibility to facilitate hyperparameter tuning. This approach allows for the exploration of various model configurations, enabling us to identify optimal settings for each architecture. The hyperparameters we considered are included in Table 1. To enhance the generalizability of our models and mitigate overfitting, we implemented dropout as a regularization technique. Dropout randomly deactivates a proportion of neurons during training, forcing the network to learn more robust and distributed representations [31].

All of our graph neural network models utilize the Rectified Linear Unit activation function [2]. We chose ReLU for its simplicity and effectiveness in addressing the vanishing gradient problem, which can occur in deep neural networks. ReLU's non-linear properties allow our graph neural networks to learn complex patterns in the graph data while maintaining computational efficiency.

3.5 Hyperparameter tuning

The majority of machine learning techniques are highly dependent on their hyperparameter values. This makes hyperparameter tuning a critical component for finding the optimal configuration of a model. For the hyperparameter tuning of our models, we constructed a hyperparameter grid which was used to test different values for the hyperparameters of the GNNs. The values in this grid are based on tuning ranges that are often observed in literature. The used hyperparameter grid is represented by Table 1.

Hyperparameter	Tuning range	Model (hyperparameter value)
Hidden dimensions	[64, 256]	GraphSAGE (64), GAT (64)
Embedding dimensions	[64, 256]	GraphSAGE (64), GAT (64)
Number of layers	[1, 3]	GraphSAGE (3), GAT (3)
Dropout rate	[0, 0.5]	GraphSAGE (0.5), GAT (0.5)
Number of heads	[1, 4]	GAT (1)
Learning rate	[0.0001, 0.1]	GraphSAGE (0.001), GAT (0.001)
Aggregator	[min, mean, max]	GraphSAGE (mean)
Batch size	[256, 4112]	GraphSAGE (2048), GAT (2048)
Number of epochs	[5, 250]	GraphSAGE (50), GAT (50)

Table 1. Hyperparameter grid used for fine-tuning the GNNs + obtained value between parentheses

Ideally, we would explore all valid configurations to determine the optimal hyperparameters for each graph neural network. This approach would involve evaluating a wide range of parameter combinations to identify the best settings for enhancing model performance. However, due to limited computational resources, this exhaustive search was not feasible. Instead, we focused our efforts on tuning key hyperparameters that significantly impact model performance. For each model, we adjusted the number of layers and the learning rate, as these parameters play a crucial role in the depth and training dynamics of the networks. For the GraphSAGE model, we tested various aggregators to find the most effective method for combining node features from neighbors. In the case of the GAT

model, we experimented with different numbers of attention heads to optimize its ability to capture complex relationships within the graph. By concentrating on these critical parameters, we aimed to balance the trade-off between thorough hyperparameter tuning and the constraints imposed by our computational resources. This approach allowed us to enhance the performance of our models within the practical limits of our environment.

3.6 Performance metrics

In the presence of an imbalanced class distribution, certain binary classification performance measures become unreliable. It has been shown that a class imbalance can exert a major impact on the value and meaning of accuracy and certain other well known performance metrics [10,15,34]. One can achieve a near perfect accuracy score by always predicting the majority class.

To assess the performance of our constructed models, we will be using the AUC-ROC and AUC-PR as performance metrics. We opted for the AUC-ROC because it is a commonly used metric in the research community. However, in the presence of a strong class imbalance this is not the most reliable metric since the FPR can remain low even when the number of false positives is high due to the large number of negatives [51]. This can result in an overly optimistic view of a model's performance. That is why we also opted to use the AUC-PR as evaluation metric. The AUC-PR provides a more informative view of model performance in tasks with a large skew in the class distribution since it focuses on correctly predicting positives by using precision and recall as its axes [16].

Note that we opted for threshold-independent evaluation metrics. These metrics allow us to assess model performance across all possible classification thresholds, rather than at a single chosen threshold. This approach saves time by eliminating the need to find the optimal threshold for each model, which is particularly beneficial given our limited access to computational resources. Additionally, it facilitates easier comparison between different models, regardless of their specific thresholds.

In the following two subsections we will provide a more in depth description of the used evaluation metrics.

3.6.1 Area Under Receiver Operating Characteristic Curve

The Area Under the Receiver Operating Characteristic Curve is a widely used, threshold independent metric for evaluating the performance of classification models. It provides a comprehensive measure of a model's ability to distinguish between classes across all possible classification thresholds.

To calculate the AUC-ROC, the Receiver Operating Characteristic curve is first plotted. This curve is created by graphing the TPR against the FPR at various threshold settings. The TPR represents the proportion of actual positive cases correctly identified, while the FPR represents the proportion of actual negative cases incorrectly classified as positive. The area under this curve is then computed to give the AUC-ROC value. The AUC-ROC score ranges from 0 to 1. A perfect classifier would have an AUC of 1, indicating it can completely separate the positive and negative classes. An AUC of 0.5 suggests the model's performance is no better than random guessing, as this would be equivalent to a diagonal line on the ROC plot. Scores below 0.5 imply the model is performing worse than random chance, which could indicate a fundamental flaw in the model or that the predictions need to be inverted.

3.6.2 Area Under Precision-Recall Curve

The Area Under the Precision-Recall Curve is another important threshold-independent metric used for evaluating classification models, particularly effective for imbalanced datasets. The AUC-PR measures the model's performance across all possible classification thresholds by plotting precision against recall. Precision is the ratio of true positives to all positive predictions, while recall is the ratio of true positives to all actual positive instances.

To calculate the AUC-PR, the Precision-Recall curve is first plotted by varying the classification threshold and computing precision and

recall at each point. The area under this curve is then calculated to give the AUC-PR score. Unlike the ROC curve, the Precision-Recall curve does not include true negatives in its computation, making it more informative for imbalanced datasets where the negative class is much larger than the positive class. The AUC-PR score ranges from 0 to 1, with higher values indicating better model performance. A perfect model would have an AUC-PR of 1, representing both high precision and high recall across all thresholds. The baseline for AUC-PR is not fixed at 0.5 (as it is for AUC-ROC) but is equal to the fraction of positive samples in the dataset. This makes AUC-PR especially useful for imbalanced datasets, where it can provide a more informative assessment of model performance than AUC-ROC.

4 Results and Discussion

Table 2 presents our models' results on the test data, with the best score for each performance metric highlighted in bold. We have included Random Forest and Logistic Regression as industry baseline models, given their widespread use in practical applications [5,43]. These methods also gained recognition in the past for their great performance across various domains, making them well-suited as baseline models for our analysis [41,56,61,54]. By including these established techniques in our analysis, we aim to provide a solid foundation for evaluating the performance of our developed graph neural networks. The Logistic Regression model is trained with L1 regularization, while the Random Forest model is configured with 200 estimators and a maximum depth of 30. These configurations were determined through a grid search. This approach does not only allow us to gauge the relative strengths of our methods but also contextualize our results within the broader landscape of machine learning applications in the field.

Model	**AUC-PR**	**AUC-ROC**
Random Forest	**0.9146**	0.9889
Logistic Regression	0.2577	0.8664
GAT	0.7844	**0.9913**
GraphSAGE	0.8263	0.9904

Table 2. Performance of the models on the testing data

The performance evaluation of the four machine learning models on the test set reveals interesting insights into their comparative strengths and limitations in the context of credit card fraud detection. The models were assessed using two evaluation metrics: AUC-PR and AUC-ROC, as discussed in Subsection 3.6. Both metrics provide valuable information about a model's ability to distinguish between fraudulent and legitimate transactions, with AUC-PR being particularly relevant for imbalanced datasets often encountered in fraud detection scenarios.

The Random Forest model demonstrates outstanding performance, leading in AUC-PR with a score of 0.9146 and achieving the second-highest AUC-ROC at 0.9889. This indicates that Random Forest excels in balancing precision and recall while maintaining excellent overall classification ability. Its high AUC-PR score, significantly better than the other models, suggests that it is particularly effective at identifying true positives without introducing many false positives. This outstanding performance can be attributed to Random Forest's ability to capture complex, non-linear patterns in the data through its ensemble of decision trees [32]. By combining multiple trees and considering various feature subsets, Random Forest can effectively model intricate relationships between transaction characteristics and fraud likelihood, making it particularly well-suited for detecting subtle patterns present in fraudulent activities.

The GAT model showcases strong performance, particularly in overall classification ability, as evidenced by its highest AUC-ROC score of 0.9913. This indicates that GAT excels at distinguishing between classes across various threshold settings, surpassing even the Random Forest in this metric by a small margin. However, its AUC-PR score (0.7844), while good, is lower than both Random Forest and GraphSAGE. This suggests that while GAT has excellent discriminative power, it may have some room for improvement in balancing precision and recall compared to Random Forest. The GAT's performance demonstrates the effectiveness of leveraging graph structure and attention mechanisms for fraud detection tasks. Its ability to focus on the most relevant neighboring nodes in the transaction network likely contributes to its strong performance.

GraphSAGE exhibits robust performance across both metrics, achieving the second-highest AUC-PR (0.8263) and a very high AUC-ROC (0.9904). This places it as a strong contender, closely trailing the Random Forest in overall performance. The model's high scores in both metrics indicate its effectiveness in maintaining a good balance between precision and recall while also providing strong overall classification. GraphSAGE's performance, similar to GAT, underscores the value of graph-based approaches for this task, showing that it can effectively leverage the graph structure of the data to outperform tra-

ditional methods like Logistic Regression and compete closely with the top-performing Random Forest. The slight performance difference between GraphSAGE and GAT suggests that the choice between these two graph-based models may depend on whether precision or overall classification is prioritized in the specific fraud detection application.

Logistic Regression significantly underperforms compared to the other models, with the lowest scores in both metrics (AUC-PR: 0.2577, AUC-ROC: 0.8664). The considerably low AUC-PR score is particularly concerning, indicating that the model struggles greatly with precision or recall, or both. Its AUC-PR is also drastically lower than that of Random Forest, highlighting a substantial performance gap. While its AUC-ROC is not as drastically low, it is still the worst among the four models. The poor performance of Logistic Regression indicates its inability to capture the likely non-linear relationships within the data, rendering it less suitable for this particular task compared to the other models. The model's linear decision boundary appears too simplistic to effectively identify the complex patterns characteristic of fraudulent transactions.

The superior performance of Random Forest and the graph neural networks in this fraud detection task underscores the importance of models that can capture non-linear patterns and complex feature interactions. Random Forest achieves this through its ensemble of decision trees, each of which model non-linear decision boundaries by default. In contrast, graph neural networks leverage structural information and node features, which can be highly indicative of fraud.

In conclusion, the Random Forest and GNNs prove to be the most effective for this fraud detection task. Random Forest slightly edges out in terms of precision-recall balance, while GAT leads in overall classification performance. The poor performance of Logistic Regression, especially in AUC-PR, suggests that the relationship between features and the target variable is likely non-linear and complex, further highlighting the superiority of the other models in handling this particular data.

5 Limitations

As was concluded in Section 4, the evaluated machine learning models provide valuable insights, but it's crucial to critically examine the limitations of this analysis and their potential impact on the results. While the performance metrics offer a useful starting point, several key factors warrant further consideration to provide a more comprehensive understanding.

A first limitation of this analysis might be the reliance on AUC-PR and AUC-ROC metrics, while informative, they may not capture the full picture of model performance. Especially in the context of imbalanced datasets typical in fraud detection. Both AUC-ROC and AUC-PR summarize performance into a single number, which obscure details about specific threshold performance. They provide an overall sense of model performance but do not convey how the model performs at a particular threshold or in specific operating conditions. This is a crucial consideration in real-world applications. Due to limited computational resources, we opted for this approach, yet these metrics still provide a robust view of the models' performance. Particularly, the AUC-PR curve is emphasized due to the class imbalance. Incorporating additional metrics such as precision, recall, F1-score at various thresholds, and confusion matrices would offer a more comprehensive evaluation. Furthermore, considering domain-specific metrics that align with the business impact of fraud detection could provide more actionable insights.

The generalizability of the results to other datasets or real-world scenarios requires careful consideration. The strong performance of certain models might be specific to this particular dataset and may not extend to other fraud detection contexts. However, it is worth noting that both GAT and GraphSAGE are designed with inductive capabilities as mentioned in Subsection 2.2.3. This allows them to generate embeddings for previously unseen nodes. This inductive nature potentially enhances their ability to generalize to new data and adapt to evolving fraud patterns. GAT's attention mechanism and GraphSAGE's neighborhood sampling approach enable these models to learn generalizable functions of node features and graph

structure, which could contribute to their robustness across different datasets. Nonetheless, to ensure the broader applicability of the findings, future studies should test the models on multiple, diverse datasets, particularly evaluating how well the inductive properties of GAT and GraphSAGE translate to varied fraud detection scenarios.

Fraudulent patterns often evolve over time, and models that perform well on historical data might quickly become outdated as new fraud techniques emerge. Going forward, the Random Forest might not necessarily remain the overall best model, and the GAT model could potentially outperform the GraphSAGE model in the future. Discussing the need for continuous monitoring and updating of models, and possibly including analysis on the models' performance across different time periods, would provide a more realistic view of their long-term effectiveness in combating fraud.

Furthermore, it is important to consider that the superior performance of the Random Forest may be attributed to the nature of our original dataset, which is tabular. For problems involving tabular data, ensembles of decision trees are often the top choice in many applications because they mitigate overfitting and are more robust in handling outliers [17,11]. In contrast, graph neural networks are designed to operate on graph-structured data, where relationships between data points are defined by edges in a graph. To utilize GNNs, we first had to construct a graph from the tabular dataset, a process that may not fully capture the intrinsic relationships and features inherent in the tabular format. Consequently, the GNNs might not have performed as well as the Random Forest, as they were not operating in their optimal data structure environment. Additionally, the impact of the train-test split is likely more severe for GNNs than for Random Forests. When we split the data, a node in the graph may lose its most influential neighbors, which can impact the learning process.

Conclusion

The objective of this master's thesis was to conduct an experimental analysis comparing the performance of graph neural networks and traditional machine learning models in identifying fraudulent transactions. We constructed a transaction network from the Kaggle dataset and developed both graph neural networks (GAT and GraphSAGE) and industry-standard baseline machine learning models (Random Forest and Logistic Regression) to perform binary (node) classification based on the transaction labels. We used the AUC-PR and AUC-ROC metrics to evaluate the performance of all models on the test set.

Our results show that while graph neural networks demonstrate strong performance, the Random Forest model achieved the highest overall scores. The GAT model performed comparably on AUC-ROC but lagged on AUC-PR. GraphSAGE also showed robust performance, outperforming the traditional Logistic Regression model.

These findings highlight the potential of graph-based approaches in fraud detection tasks. While they did not surpass the Random Forest model in this particular dataset, their strong performance suggests that GNNs can effectively leverage the structural information in transaction networks. The superior performance of Random Forest underscores the continued effectiveness of ensemble methods in handling complex, non-linear relationships in tabular data.

In conclusion, this experimental analysis contributes to the growing body of research on applying graph-based methods to financial fraud detection. While traditional methods like Random Forest remain highly effective, the promising results of GNNs suggest that further research and refinement of these approaches could yield valuable insights and improvements in fraud detection capabilities.

Based on the results of our work, we suggest the following topics for future work:

- Exploring other (more advanced) GNN architectures and their applications in credit card fraud detection remains a promising avenue for future research. In our study, we were constrained by the nature of our dataset, which necessitated the use of a heterogeneous graph structure. This limitation significantly narrowed the range of applicable GNN architectures, as only a minority of existing models are designed to effectively operate on heterogeneous graphs. Access to a dataset suitable for constructing a homogeneous graph could expand opportunities to explore the applicability of various other GNN architectures.
- Investigating the impact of different graph construction methods on model performance. By comparing graph construction methods and their impact on model performance metrics, we can identify the most effective approaches for representing fraud-related data in graph form.
- Conducting longitudinal studies to assess model performance over time as fraud patterns evolve. By implementing these longitudinal study practices, we can ensure that GNN-based fraud detection systems remain robust and effective in the face of constantly evolving fraud tactics
- Exploring interpretability techniques for GNNs to provide insights into their decision-making process. This can transform GNNs from "black box" models into more transparent and understandable tools for fraud detection.

List of Figures

List of Tables

References

[1] Abdallah, A., Maarof, M.A., Zainal, A.: Fraud detection system: A survey. Journal of Network and Computer Applications **68**, 90–113 (2016)

[2] Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 (2018)

[3] Ali, A., Abd Razak, S., Othman, S.H., Eisa, T.A.E., Al-Dhaqm, A., Nasser, M., Elhassan, T., Elshafie, H., Saif, A.: Financial fraud detection based on machine learning: a systematic literature review. Applied Sciences **12**(19), 9637 (2022)

[4] Attivilli, R., Arul Jothi, A.: Serverless stream-based processing for real time credit card fraud detection using machine learning. In: 2023 IEEE World AI IoT Congress (AIIoT). pp. 0434–0439. IEEE (2023)

[5] Awoyemi, J.O., Adetunmbi, A.O., Oluwadare, S.A.: Credit card fraud detection using machine learning techniques: A comparative analysis. In: 2017 international conference on computing networking and informatics (ICCNI). pp. 1–9. IEEE (2017)

[6] Baesens, B., Van Vlasselaer, V., Verbeke, W.: Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection. John Wiley & Sons (2015)

[7] Berg, R.v.d., Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263 (2017)

[8] Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C.: Data mining for credit card fraud: A comparative study. Decision support systems **50**(3), 602–613 (2011)

[9] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: Structure and dynamics. Physics Reports **424**(4), 175–308 (2006)

[10] Branco, P., Torgo, L., Ribeiro, R.P.: Relevance-based evaluation metrics for multi-class imbalanced domains. In: Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I 21. pp. 698–710. Springer (2017)

[11] Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning. pp. 161–168 (2006)

[12] Chaudhary, K., Yadav, J., Mallick, B.: A review of fraud detection techniques: Credit card. International Journal of Computer Applications **45**(1), 39–44 (2012)

[13] Cheng, D., Wang, X., Zhang, Y., Zhang, L.: Graph neural network for fraud detection via spatial-temporal attention. IEEE Transactions on Knowledge and Data Engineering **34**(8), 3800–3813 (2020)

[14] Cherif, A., Badhib, A., Ammar, H., Alshehri, S., Kalkatawi, M., Imine, A.: Credit card fraud detection in the era of disruptive technologies: A systematic review. Journal of King Saud University-Computer and Information Sciences **35**(1), 145–174 (2023)

[15] Daskalaki, S., Kopanas, I., Avouris, N.: Evaluation of classifiers for an uneven class distribution problem. Applied artificial intelligence **20**(5), 381–417 (2006)

[16] Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on Machine learning. pp. 233–240 (2006)

[17] Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems. pp. 1–15. Springer (2000)

[18] Do, K., Tran, T., Venkatesh, S.: Graph transformation policy network for chemical reaction prediction. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 750–760 (2019)

[19] Duman, E., Elikucuk, I.: Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization. In: Advances in Computational Intelligence. pp. 62–71. Springer Berlin Heidelberg, Berlin, Heidelberg

[20] Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems **28** (2015)

[21] Estabrooks, A., Jo, T., Japkowicz, N.: A multiple resampling method for learning from imbalanced data sets. Computational intelligence **20**(1), 18–36 (2004)

[22] European Central Bank: Report on card fraud in 2020 and 2021, https://www.ecb.europa.eu/pub/pdf/cardfraud/ecb.cardfraudreport202305~5d832d6515.en.pdf

[23] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: The world wide web conference. pp. 417–426 (2019)

[24] Feng, B., Xu, H., Xue, W., Xue, B.: Every corporation owns its structure: Corporate credit rating via graph neural networks. In: Chinese Conference on Pattern Recognition and Computer Vision (PRCV). pp. 688–699. Springer (2022)

[25] Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

[26] Freitas, A.A., Lavington, S.H.: Approaches to Speed Up Data Mining, pp. 89–108. Springer US, Boston, MA (2000). https://doi.org/10.1007/978-1-4615-5521-6_10

[27] Gyamfi, N.K., Abdulai, J.D.: Bank fraud detection using support vector machine. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). pp. 37–41. IEEE (2018)

[28] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)

[29] Hamilton, W.L.: Graph representation learning. Morgan & Claypool Publishers (2020)

[30] Heryadi, Y., Wulandhari, L.A., Abbas, B.S., et al.: Recognizing debit card fraud transaction using chaid and k-nearest neighbor: Indonesian bank case. In: 2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS). pp. 1–5. IEEE (2016)

[31] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)

[32] Ho, T.K.: Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. vol. 1, pp. 278–282. IEEE (1995)

[33] Höppner, S., Baesens, B., Verbeke, W., Verdonck, T.: Instance-dependent cost-sensitive learning for detecting transfer fraud. European Journal of Operational Research **297**(1), 291–300 (2022)

[34] Hossin, M., Sulaiman, M.N.: A review on evaluation metrics for data classification evaluations. International journal of data mining & knowledge management process **5**(2), 1 (2015)

[35] Jiang, J., Chen, J., Gu, T., Choo, K.K.R., Liu, C., Yu, M., Huang, W., Mohapatra, P.: Anomaly detection with graph convolutional networks for insider threat and fraud detection. In: MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM). pp. 109–114. IEEE (2019)

[36] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

[37] Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al.: Handling imbalanced datasets: A review. GESTS international transactions on computer science and engineering **30**(1), 25–36 (2006)

[38] Kou, Y., Lu, C.T., Sirwongwattana, S., Huang, Y.P.: Survey of fraud detection techniques. In: IEEE International Conference on Networking, Sensing and Control, 2004. vol. 2, pp. 749–754 Vol.2 (2004)

[39] Koutra, D., Ke, T.Y., Kang, U., Chau, D.H., Pao, H.K.K., Faloutsos, C.: Unifying guilt-by-association approaches: Theorems and fast algorithms. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II 22. pp. 245–260. Springer (2011)

[40] Kulatilleke, G.K.: Challenges and complexities in machine learning based credit card fraud detection (2022)

[41] Larivière, B., Van den Poel, D.: Predicting customer retention and profitability by using random forests and regression forests techniques. Expert systems with applications **29**(2), 472–484 (2005)

[42] Liu, G., Tang, J., Tian, Y., Wang, J.: Graph neural network for credit card fraud detection. In: 2021 International Conference on Cyber-Physical Social Intelligence (ICCSI). pp. 1–6. IEEE (2021)

[43] Minastireanu, E.A., Mesnita, G.: An analysis of the most used machine learning algorithms for online fraud detection. Informatica Economica **23**(1) (2019)

[44] Mishra, A., Ghorpade, C.: Credit card fraud detection on the skewed data using various classification and ensemble techniques. In: 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS). pp. 1–5. IEEE (2018)

[45] Mittal, S., Tyagi, S.: Performance evaluation of machine learning algorithms for credit card fraud detection. In: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). pp. 320–324. IEEE (2019)

[46] Nasteski, V.: An overview of the supervised machine learning methods. Horizons. b **4**(51-62), 56 (2017)

[47] Newman, M.: Networks. Oxford university press (2018)

[48] Rahmani, K., Thapa, R., Tsou, P., Chetty, S.C., Barnes, G., Lam, C., Tso, C.F.: Assessing the effects of data drift on the performance of machine learning models used in clinical sepsis prediction. International Journal of Medical Informatics **173**, 104930 (2023)

[49] Ryman-Tubb, N.F., Krause, P., Garn, W.: How artificial intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. Engineering applications of artificial intelligence **76**, 130–157 (2018)

[50] Sahin, Y., Bulkan, S., Duman, E.: A cost-sensitive decision tree approach for fraud detection. Expert Systems with Applications **40**(15), 5916–5923 (2013)

[51] Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PloS one **10**(3), e0118432 (2015)

[52] SamanehSorournejad, Zojaji, Z., Atani, R.E., Monadjemi, A.H.: A survey of credit card fraud detection techniques: Data and technique oriented perspective (2016)

[53] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE transactions on neural networks **20**(1), 61–80 (2008)

[54] Shah, K., Patel, H., Sanghvi, D., Shah, M.: A comparative analysis of logistic regression, random forest and knn models for the text classification. Augmented Human Research **5**(1), 12 (2020)
[55] Sorournejad, S., Zojaji, Z., Atani, R.E., Monadjemi, A.H.: A survey of credit card fraud detection techniques: Data and technique oriented perspective. CoRR **abs/1611.06439** (2016), http://arxiv.org/abs/1611.06439
[56] Statnikov, A., Wang, L., Aliferis, C.F.: A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC bioinformatics **9**, 1–10 (2008)
[57] Stolfo, S., Fan, W., Lee, W., Prodromidis, A., Chan, P.: Cost-based modeling for fraud and intrusion detection: results from the jam project. In: Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00. vol. 2, pp. 130–144 vol.2 (2000)
[58] Van Belle, R., Baesens, B., De Weerdt, J.: Catchm: A novel network-based credit card fraud detection method using node representation learning. Decision Support Systems **164**, 113866 (2023)
[59] Van Vlasselaer, V.: Fair: Forecasting and network analytics for collection risk management. (2015)
[60] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
[61] Whitrow, C., Hand, D.J., Juszczak, P., Weston, D., Adams, N.M.: Transaction aggregation as a strategy for credit card fraud detection. Data mining and knowledge discovery **18**, 30–55 (2009)
[62] Wu, Y., Lian, D., Xu, Y., Wu, L., Chen, E.: Graph convolutional networks with markov random field reasoning for social spammer detection. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 1054–1061 (2020)
[63] Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Third IEEE international conference on data mining. pp. 435–442. IEEE (2003)
[64] Zhang, G., Li, Z., Huang, J., Wu, J., Zhou, C., Yang, J., Gao, J.: efraudcom: An e-commerce fraud detection system via compet-

itive graph neural networks. ACM Transactions on Information Systems (TOIS) **40**(3), 1–29 (2022)

[65] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI open **1**, 57–81 (2020)