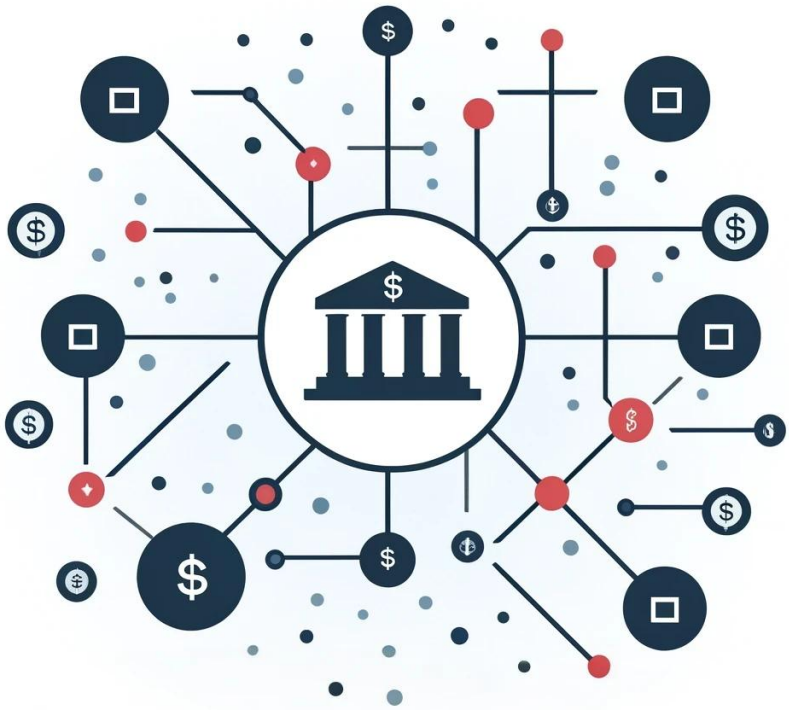
CHALMERS
UNIVERSITY OF TECHNOLOGY



Anti-Money Laundering with Unreliable Labels

Master's thesis in Electrical engineering

Jesper Bergquist

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2024

www.chalmers.se

Master's thesis 2024

Anti-Money Laundering with Unreliable Labels

Jesper Bergquist

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Anti-Money Laundering with Unreliable Labels

Jesper Bergquist

Supervisors: Johan Östman, AI Sweden & Edvin Callisen, AI Sweden
Examiner: Alexandre Graell I Amat, Department of Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: **Illustration of a transaction network as a graph.**

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Anti-Money Laundering with Unreliable Labels

Jesper Bergquist
Department Electrical engineering
Chalmers University of Technology

Abstract

This report examines the effectiveness of Graph Neural Networks (GNNs) in detecting money laundering activities using transaction data with unreliable labels. It explores how weakly supervised learning, specifically with GNNs, manages the challenges posed by missing and inaccurate labels in anti-money laundering (AML) systems. The study utilizes simulated transaction datasets to compare the performance of GNNs against traditional statistical models. Findings indicate that GNNs, due to their ability to process relational data structures, demonstrate superior adaptability and accuracy in scenarios with label deficiencies. This research provides effective strategies for enhancing anti-money laundering systems by employing GNNs to more effectively manage data challenges.

Keywords: GNN, AML, money laundering, machine learning, deep learning, graph neural networks.

Acknowledgements

I would like to extend omy heartfelt gratitude to my supervisors, Johan Östman and Edvin Callisen from AI Sweden, and Anton Chen from Handelsbanken, for their invaluable feedback, guidance, and support throughout the course of our thesis. Additionally, I would like to thank all the team members at AI Sweden and Handelsbanken who contributed their time, insights, and assistance, making this work possible. Your collective efforts and encouragement have been instrumental in the successful completion of our project. Jesper Bergquist, Gothenburg, June 2024

Contents

1

Introduction

Money laundering is the process of disguising funds acquired from illicit activities to make them appear as though they have been obtained through legitimate means. This practice is a fundamental operation for organized crime groups, involving funds from illegal activities such as corruption, gambling, embezzlement, and drug trafficking. According to estimates from the United Nations in 2009, as much as 1.6$ trillion dollars are laundered annually, which equates to about 2 to 5 % of total GDP [1]–[3]. Despite various anti-money laundering (AML) measures by banks and financial authorities, the scale of the problem is still large. Close to 40% of all criminal networks in the EU are involved in the trade of illegal drugs, and more than 80% of the criminal networks use legal business structures with two thirds partaking in corruption on a regular basis [4]–[6].

Current AML systems have demonstrated limited effectiveness in detecting illegal activities [7]–[9]. Using a data-driven approach in AML is a strategy that improves efficiency and scales effectively to handle the large volumes of transactions, but this presents multiple challenges. The intricacy of laundering schemes, and their attempt to be stealthy by behaving normally often makes them difficult to identify [10]. Additionally, banks generally only have access to data concerning their own customers' accounts and transactions, which provides a restricted perspective. The data available to banks may also be incomplete or contain errors, with potential inaccuracies in labeling, due to the lengthy investigations [11]. Achieving reliable labels for AML purposes is a difficult task, that is deeming if an account really is laundering money or not is hard, because banks are reliant on the financial authorities for feedback if a suspicious case was correct or not. These factors contribute to the obscurity

surrounding money laundering, complicating detection efforts.

Since transaction networks can be represented as graphs, AML can either be viewed as a node or a link classification problem. If it is viewed as a link classification problem, the approach is to find specific illicit transactions in the network, and if it is viewed as node classification problem, the approach is to find specific accounts partaking in illegal activities. The best approach depends on the situation, but node classification can be preferred because it is less computationally intensive, as there are fewer accounts than transactions in a financial institution. In this thesis, the focus will be on classifying accounts, hence, it will be viewed as a node classification problem, this choice is further motivated in section 5.0.3.

One common approach is using statistical machine learning models to try and capture the behavior of normal versus suspicious accounts [12]. As previously mentioned, a significant challenge in this domain is the limited number of labels, which can be treated as a weakly supervised problem. In weakly supervised settings statistical models often perform poorly because with few labeled data-points, they are not able to capture the nature of the data [13].

Graph neural networks (GNNs) are particularly promising in the context of financial anomaly detection. Recent research highlights the potential of GNNs to enhance anomaly detection capabilities in financial networks [14]–[16]. GNNs excel at processing graph-based structures, enabling them to uncover complex relationships within data. They facilitate the understanding of interactions within financial networks. This approach could significantly improve the detection of sophisticated money laundering schemes. Deploying GNNs on sparsely labeled and inaccurate data would present a weakly supervised scenario, where the models must learn from imprecise and incomplete information [17].

The main goal of this project is to evaluate how well different GNN models detect money laundering using simulated transaction data. We will compare these results with those from statistical models to understand the strengths and weaknesses of GNNs.

A key part of this research is to study how data quality affects GNN performance. We will create several datasets with specific errors, e.g., missing and incorrect labels,

to see how these issues impact both GNNs and statistical models. This will provide insights into how different data problems influence the accuracy and reliability of these models.

By achieving these objectives, the research aims to improve our understanding of the practical challenges and limitations of current AML technologies. To address this challenge, the study focuses on the following two main research questions:

1. **How does the presence of incomplete labels affect the performance of statistical models and GNN models in detecting money laundering activities?**

   This question aims to explore how each model type compensates for missing labels and the resulting effects.

2. **What is the impact of inaccurate labels on the performance of GNN and statistical models?**

   Specifically, how do incorrect labels in the dataset influence the performance and reliability of the models classifying money laundering accounts?

To the best of our knowledge, the specific impact of different types of transaction data noise in a weakly supervised setting, on GNNs and statistical models, has not been thoroughly investigated. The insights gained could help tailor GNNs to be more reliable and effective for practical use in detecting financial crimes.

The findings from this research could provide a foundation for future studies and help improve regulatory practices and financial security measures, leading to stronger defenses against financial crimes. Specifically, our work demonstrates the usability of GNNs in a weakly supervised scenario, showing their potential effectiveness in detecting complex patterns indicative of money laundering activities. This could significantly influence the adoption of GNNs for AML purposes within the financial sector.

By proving GNNs capability to operate efficiently with limited labeled data, our research encourages further development and integration of these models into real-world AML systems. As financial institutions and regulatory bodies begin to implement GNN-based approaches, we anticipate a substantial improvement in the accuracy and reliability of money laundering detection. This enhancement could

lead to more sophisticated monitoring systems that can uncover previously undetected laundering schemes, thereby increasing the amount of illicit activity that is identified and prevented.

2

Background

2.1 Money laundering

Money laundering is an illegal activity that impacts the global financial system by disguising illegally obtained funds as legitimate income. This process enables criminals to use their illegal profits in the legal system without being prosecuted and compromises the integrity of financial institutions, leading to social and economic consequences [2], [3].

The process of money laundering typically consists of three stages [18]. The first stage is placement, where illicit funds are introduced into the legitimate financial system. This may involve depositing large sums of cash into bank accounts or purchasing high-value assets such as real estate or luxury goods. The second stage is layering, where the launderer executes a series of complex transactions to obscure the origins of the illicit funds. This often includes transferring money between various accounts. The final stage is integration, wherein the laundered money is reintroduced into the economy, now appearing as legitimate.

Globally, money laundering poses a significant threat by facilitating other serious crimes, such as drug trafficking and terrorism. The United Nations Office on Drugs and Crime has reported estimates that up to 5 percent of global GDP is laundered each year, highlighting the vast scale and impact of this activity [3]. Effective measures against money laundering are vital for maintaining financial market integrity and ensuring national and global security. These measures include strict regulatory frameworks, thorough monitoring of financial transactions, and international cooperation [19].

2.2 Current AML practices

Traditional AML approaches have relied on rule-based systems [20]. These systems use predefined rules and thresholds to identify potentially suspicious activity. While somewhat effective, these methods can be rigid and may miss complex laundering schemes. These systems result in a false positive rate (FPR) of about 95-98% [21]. To improve detection, research has shifted towards learning-based methods, employing classical machine learning techniques such as logistic regression and support vector machines. These techniques analyze patterns and assess the likelihood of transactions being associated with money laundering [12]. However, they struggle with the complexity of money laundering tactics. In response to these limitations, there has been a shift towards employing advanced deep learning models [22]. With the vast amounts of transactional data available, these models can learn from a broader set of behavioral patterns where accounts are seen as dependent on their neighbours, providing a more effective tool in detecting laundering activities.

2.3 AMLSim

Machine learning models depend on the quality of the data used to train them. Therefore, it is crucial that the training data is of high quality and represents a broad spectrum of the target domain. This includes ensuring the data is accurate, complete, consistent, and diverse to avoid biases and ensure the model's robustness and generalizability across different scenarios. There is a shortage of such publicly available datasets in the field of AML [23]. One commonly used dataset to develop money-laundering detection methods is the Elliptic Data Set [24]. This dataset contains 203 769 nodes and 234 355 edges over a span of 98 weeks. The nodes represent a bitcoin transaction and edges represent a flow of bitcoins between one transaction and the other, with 2% of the nodes labelled as money-laundering, 21% are labelled as normal [25]. This dataset has several shortcomings. Firstly, the limited usage of only bitcoin transactions may not fully capture the complexity of real-world money laundering activities. Consecutively, the dataset has a large

imbalance of the classes, with only a small portion of the labelled nodes being illicit. This could cause biases whilst training machine learning models. Also, since the dataset only contains transactions, there is limited capabilities of capturing the behavior of specific accounts.

Instead, using a simulation tool gives full control of the properties of the data, in this thesis simulated data is generated by deploying a version of AMLSim, a simulation platform developed by the MIT-IBM Watson AI Lab to aid in the fight against money laundering [26]. The simulation platform is designed to generate synthetic datasets of financial transactions for research and development in AML technologies. AMLSim has been further enhanced by a team at AI Sweden as part of the project named *Federated Learning in Banking* [27]. This project is a collaboration between AI Sweden, Handelsbanken, and Swedbank.

In AMLSim, each node within the simulation represents a bank account, and edges between these nodes symbolize financial transactions. This structure allows AMLSim to create complex graphs that mimic the intricate networks through which money might be laundered in the real world.

AMLSim works by first defining the spatial part of the transaction graph, forming predefined agents in graph-topologies. These agents are then allowed to perform transactions over a discrete-time simulation, achieving the temporal aspect of the graph.

The platform operates by employing a multi-agent approach, where each agent (or node) behaves as an individual bank account. These agents interact by transferring funds among themselves, creating a dynamic graph of transactions. Importantly, AMLSim includes the capability to simulate illicit behaviors by programming a subset of these agents to engage in activities typical of money laundering (see Figure 3.4), based on observed patterns from real cases [28].

AMLSim provides a controlled environment for generating transaction data, making it a valuable resource for researchers. It facilitates a way to generate data to enable the testing of new AML methods, particularly those incorporating deep learning and graph analytics, under various scenarios and conditions. The data produced by AMLSim can be utilized to train algorithms for improved identification and

prediction of suspicious activities within extensive, complex financial networks.

2.4 Statistical machine learning models

Detecting suspicious activity in transaction graphs can be done with many different methods. One way of classifying money laundering is with the usage of statistical machine learning models. Several popular and widely used models include support vector machines (SVM), k-nearest neighbours (KNN), logistic regression, random forest, and XGBoost.

SVMs work by finding the optimal hyperplane that separates the two classes in the feature space. It is particularly useful with high-dimensional data [29].

KNN is an algorithm that classifies data based on the majority vote of the nearest neighbours in the feature space. KNNs are particularly useful with non-linear data since it does not make an assumption of the underlying distributions [30].

Logistic regression is a popular method used for binary classification. It models the probability of an outcome based on the input variables, making it suitable for money laundering detection where the goal is estimate a probability for an account being suspicious. It is a simple and interpretable model, making it easy to understand the decisions made [31].

Random forest is an ensemble learning method that trains several decision trees in parallel and merges their outputs to improve its performance. This is an effective model for money laundering detection due to its capability to handle large amounts of data and ability to find complex patterns in the data [32].

XGBoost is another ensemble method that builds a model by sequentially optimizing decision trees with respect to the loss function. It excels in handling unbalanced datasets, and is widely used in several fields [33]–[36]. It is known from previous work that XGBoost is one of the best methods for detecting money laundering in graph transaction data [15], [37]. This is something that will be challenged by graph neural networks in this thesis.

2.5 Graphs

A graph is composed of nodes (also called vertices) and edges (also called links) that connect pairs of nodes [38]. An *undirected graph* is defined by a set of *nodes* $\mathcal{V} = \{v_1, \ldots, v_N\}$ and a set of *edges* $\mathcal{E} = \{e_1, \ldots, e_M\}$, where an edge $e_i = \{v, u\}$ connects nodes v and u in $\mathcal{V}$. This implies that if there is an edge between node v and node u, traversal is possible in both directions. Formally, an undirected graph G is represented as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges, with each edge being an unordered pair of nodes $\{u, v\}$.

In contrast, a *directed graph* (or digraph) includes edges with specific directions, where an edge from node v to node u is represented as an ordered pair (v, u), indicating the direction from v to u [39].

In the context of this thesis, the nodes represent bank accounts, and an edge between two nodes represents a transaction between these entities. Furthermore, each node is associated with a *node feature vector* $x_i \in \mathbb{R}^n$ and a label, which contains features, the features and labels for this project is further explained in table 3.2.

The *graph structure* refers to the sets $\mathcal{V}$ and $\mathcal{E}$, while the term *graph* includes both the graph structure and the node feature vectors X.The neighbors of a node v are defined as the set $\mathcal{N}(v) = \{u \mid \{v, u\} \in \mathcal{E}\}$, which includes all nodes u connected to v by an edge. An example of an undirected graph with nodes A, B, C and D with feature vectors can be sen in Figure 2.1.

2.6 Graph neural networks

Current AML systems are usually rule based and have had limited success in detecting illegal activities. Recent research suggests that GNNs could enhance AML [14]–[16]. GNNs can process graph structures and reveal underlying complex relationships, and transaction networks can be represented as graphs, and thus GNN shows the promise to not only consider transactions in isolation, but to identify patterns and anomalies by considering the broader context of interactions in financial networks. This makes GNN a promising candidate to uncover complex money launder-

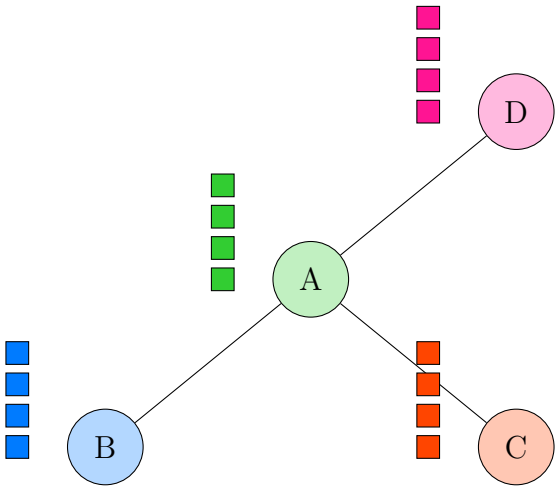


Figure 2.1: An undirected graph with nodes A, B, C, and D. Node A is connected to all other nodes, while nodes B, C, and D are only connected to node A. The squares represent feature vectors associated with each node.

ing schemes to further reduce the illegal practice.

GNNs are sophisticated models that harness the inherent structure of graph data to perform various tasks such as classifying nodes, edges, or entire networks. These models are particularly effective in domains where data relationships can be naturally represented as graphs, including social networks, molecular structures, and transaction networks [40].

A GNN iteratively updates node embeddings by integrating both their own features and the features of their neighboring nodes. This process is structured through layers, where each layer refines the node's representation by employing two key functions:

1. **Aggregation Function**: This function collects and combines the feature vectors of a node's neighbors into a single vector that summarizes the local neighborhood information.
2. **Updating Function**: This function then takes the aggregated information and the node's current features to produce a new, updated embedding.

Mathematically, the operation of a GNN across its layers can be encapsulated as follows:

$$\mathbf{h}_v^k = \text{Updating}^{(k-1)}(\mathbf{h}_v^{(k-1)}, \mathbf{m}_{\mathcal{N}(v)}^{(k-1)}) \tag{2.1}$$

$$\mathbf{m}_{N(v)}^k = \text{Aggregation}^{(k-1)}(\{\mathbf{h}_u^{(k-1)} : \forall u \in \mathcal{N}(v)\}) \tag{2.2}$$

Here, $\mathbf{h}_v^k$ represents the embedding of node v at the k-th layer, while $\mathbf{m}_{\mathcal{N}(v)}^{(k)}$ denotes the aggregated message derived from the embeddings of v's neighbors at the previous layer. These functions are designed to be differentiable, enabling the use of gradient-based learning techniques to optimize the network parameters [41].

As illustrated in Figure 2.2, the node A is updated by aggregating information from its neighboring nodes and information from itself. This visualization shows how the features from nodes B, C, and D are combined with information from the node A and used to refine the embedding of node A.

This flexible framework allows GNNs to learn to encode both node and topological features into the embeddings, making them powerful tools for predictive and analytic tasks on graph-structured data.

Different Graph Neural Networks employ various aggregation and updating functions. These functions are tailored to the specific architecture and objectives of the GNN model being used. The three different GNN models utilized in this project are the following:

2.6.1 Graph Convolutional Networks

Graph convolutional networks (GCNs) apply convolutional principles to graph-structured data, allowing each node in the graph to be represented by aggregating features from its immediate neighbors [42]. Developed by Thomas N. Kipf and Max Welling, GCNs use a layer-wise propagation rule based on the eigen-decomposition of graph Laplacians, simplifying the convolution operation in the spectral domain. By doing so, GCNs efficiently blend local node features with their neighborhood topology, resulting in a powerful representation that captures both individual and collective properties. This approach is particularly effective for datasets where the structural connections between entities are pivotal to their overall data representa-

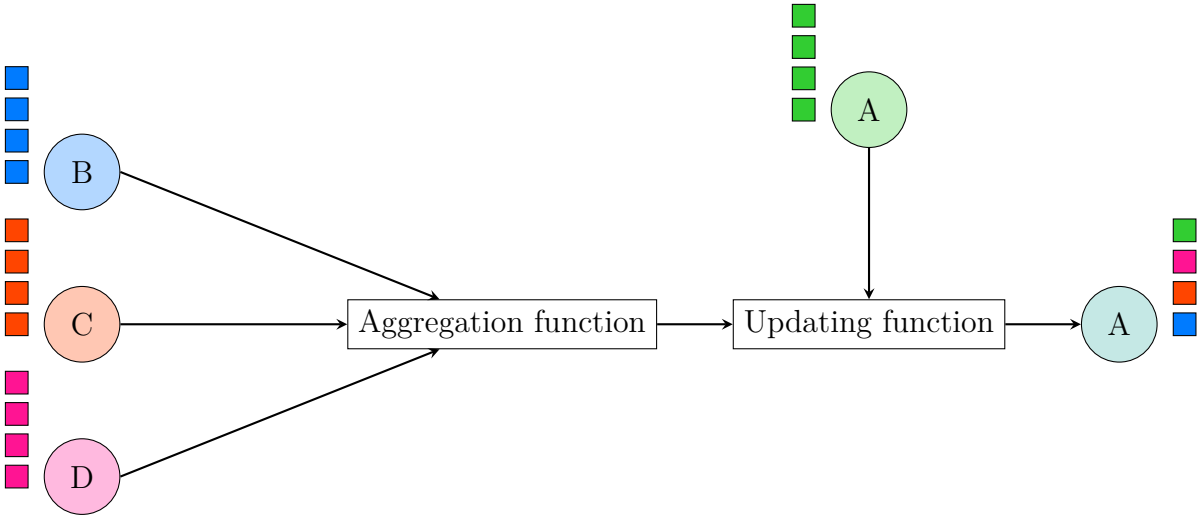


Figure 2.2: A visualization of how the neighborhood information is aggregated and used to update the embedding of node A. In this diagram, circles represent nodes and squares represent their feature vectors. Node A (initially green) aggregates information from its neighboring nodes B (blue), C (orange), and D (pink), as well as from itself, to produce a new node representation for A (shown as a combination of the colors of its neighbors).

tion. GCNs have become a promising model for various applications, including those in social network analysis, recommendation systems, and bioinformatics, where the intrinsic link between data points critically informs their analysis [43]–[45].

In the GCN used in this thesis, the node representations are updated using the ReLU activation function. The combined aggregation and update rule is given by:

$$\mathbf{h}_v^{(l+1)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{c_{vu}} \mathbf{h}_u^{(l)} \mathbf{W}^{(l)}\right), \tag{2.3}$$

where ReLU is the activation function, $\mathcal{N}(v)$ represents the set of neighbors of node v, c_{vu} is a normalization constant (typically based on the degrees of nodes v and u), $\mathbf{h}_u^{(l)}$ denotes the feature vector of node u at layer l, and $\mathbf{W}^{(l)}$ is the weight matrix for layer l. This equation captures both the aggregation of features from the neighboring nodes and the node itself, as well as the subsequent update of the node's feature representation.

2.6.2 GraphSAGE

GraphSAGE is a framework designed for learning on large graphs, developed by William L. Hamilton, Rex Ying, and Jure Leskovec [46]. GraphSAGE employs a unique approach involving sampling and aggregation. It samples a fixed-size neighborhood and aggregates their features to update a node's representation. This allows GraphSAGE to efficiently handle dynamically growing graphs by incorporating new nodes and continuing to generate high-quality embeddings without needing access to the full graph.

The GraphSAGE model used in this thesis employs specific aggregation and updating functions. The mathematical representation of these functions is as follows:

$$\mathbf{h}_v^{(k)} = \mathrm{ReLU}\Big(\mathbf{W}^{(k)} \cdot \mathrm{MEAN}\Big(\{\mathbf{h}_v^{(k-1)}\} \cup \{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v)\}\Big)\Big) \tag{2.4}$$

where the MEAN() function calculates the feature-wise mean of the node embedding vectors. ReLU is used as the activation function, and $\mathbf{W}^{(k)}$ represents a weight matrix specific to each layer k.

2.6.3 Graph attention network

The graph attention network (GAT) incorporates attention mechanisms into the graph neural network framework, allowing for more nuanced aggregation of neighbor features [47]. GAT computes the hidden representations of each node by attending over its neighbors, thus assigning different weights to different nodes in a neighborhood without any predefined pooling strategy. This attention-based approach enables the model to focus on more relevant information and diminish the less useful data dynamically, enhancing the model's adaptability to complex and noisy data environments.

The GAT model utilized in this thesis is represented as follows

$$\mathbf{h}_v^{(k)} = \mathrm{ReLU}\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{u\in\mathcal{N}(v)} \alpha_{vu}^k \mathbf{W}^k \mathbf{h}_u\right). \tag{2.5}$$

The rectified linear unit (ReLU) activation function is employed in this thesis, K

represents the number of attention heads, and α_{vu}^{k} are the normalized attention coefficients that indicate the significance of the information from node u to node v.

2.7 Weakly supervised learning

Weakly supervised learning is a machine learning paradigm in which the model is trained with imprecise, noisy, or incomplete labels [17]. Unlike fully supervised learning, where each training instance is associated with a precise output label, weakly supervised learning tackles scenarios where acquiring high-quality labels is either expensive or impractical. Let's denote the input space by $\mathcal{X}$ and the output space by $\mathcal{Y}$. A typical fully supervised learning model is trained using a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. This thesis investigates incomplete supervision on its own, as well as the combination of incomplete and inaccurate supervision, which one expects to find in real-world data.

Incomplete supervision occurs when only a portion of the training data is labeled. In this scenario, the dataset can be represented as

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n} \cup \{x_i\}_{i=n+1}^{n+m}. \tag{2.6}$$

Here, only the first n instances have labels, and the remaining m instances are unlabeled. Each $x_i \in \mathcal{X}$ represents an input instance, and $y_i \in \mathcal{Y}$ represents the corresponding label.

Inaccurate supervision, on the other hand, involves training data that contains incorrect labels. This can be represented as

$$\mathcal{D} = \{(x_i, \tilde{y}_i)\}_{i=1}^{n} \tag{2.7}$$

where $\tilde{y}_i$ may be an incorrect label. Here, $x_i \in \mathcal{X}$ represents an input instance, and $\tilde{y}_i \in \mathcal{Y}$ represents the potentially incorrect label.

2.8 Weakly supervised learning in AML

In real-world bank transaction data, obtaining reliable labels is extremely challenging. Research conducted by Copenhagen Business School [21] indicates that only

about 5% of all money-laundering attempts are detected and intercepted. Because of this statistic, it is reasonable to assume that, when curating a graph dataset for anti-money laundering detection, a significant portion of node labels will be unknown. To the best of our knowledge, no research has been conducted on this assumption in the context of anti-money laundering. Labels reported back from the financial police are subject to proof burden and human errors and may, hence, contain errors. Therefore, labeling only nodes with an absolute certainty could ensure that new money-laundering strategies do not fly under the radar.

Compared to statistical machine learning models (see Section 2.4), GNNs have different capabilities of handling incomplete and inaccurate labels in graph data, where GNNs have an advantage.

Firstly, GNNs are able to capture the relational dependencies between nodes in a graph [48]. In the case of AML, this means understanding the intricate transaction topologies for suspicious activities. Statistical models on the other hand, treat nodes as independent and identically distributed, which limits their ability to exploit the graph structured data.

Secondly, GNNs can incorporate both node and edge features [48]. This enables GNNs to consider not only the properties of the accounts, but also the nature of their interaction with other accounts. Statistical models are only able to extract node features from the edge information, and because of this, fails to capture the relation between accounts.

Moreover, GNN's neighbour aggregation mechanism allows nodes with missing labels to still contribute with valuable information [48]. By aggregating the features and labels of neighboring nodes, GNNs can propagate information across the network to fill in the information gaps left by accounts with an unknown status to money laundering. Statistical models have no way of mimicking this aggregating mechanism, putting them at a disadvantage in the field of AML where missing labels are an intrinsic property.

2.9 Different perspectives in AML

In AML, regulatory bodies and financial institutions approach the problem from different angles, each with unique priorities and measures of success. The Financial Action Task Force (FATF), along with other regulatory bodies, focus on maximizing the detection of money laundering activities with high accuracy. Their success is measured by the thoroughness and effectiveness of their enforcement actions, emphasizing comprehensive suspicious activity reporting and rigorous customer due diligence [49].

Financial institutions prioritize regulatory compliance to avoid penalties and ensure smooth operations. Their objective is to detect money laundering efficiently while managing costs and minimizing disruptions. Success for banks is often measured by their ability to adhere to regulatory standards and maintain operational integrity, balancing accurate detection with the demands of daily operations [50].

This difference in focus could lead to varied approaches in AML practices. Regulatory bodies may push for more exhaustive investigative techniques to uncover as many illicit activities as possible, even if it imposes a higher compliance burden on financial institutions. Banks, on the other hand, might streamline their processes to ensure compliance without overwhelming their resources.

3

Methods

3.0.1 Data generation and preparation

Within this thesis, three different datasets were generated: an easy dataset with approximately 25% money laundering transactions, a medium dataset with approximately 5% money laundering transactions, and a difficult dataset with approximately 1% money laundering transactions. Old reports have estimated the rate of money laundering transactions to be very low, around 0.05% to 0.1% [51]. Since exact numbers are hard to determine, we used a range of values to simulate different scenarios. We chose these numbers to ensure that the class imbalance does not obscure our results too much, providing a clearer evaluation of our methods. These datasets were created using a simulation tool based on AMLSim developed by IBM [26], and further enhanced by AI Sweden (see Section 2.3), to mimic the Swedish Swish transaction network [52]. This tool allows for the configuration of various spatial and temporal parameters for both the normal and money laundering accounts.

As there is no public information available on how to choose the parameters for the synthetic data, one must qualitatively guess reasonable numbers. To this end, a reverse engineering pipeline was devised, visually described in figure 3.1, with the objective to find parameters that yield a pre-determined false positive rate when the surveillance model is based on a decision tree. The data-generation procedure is outlined next.

Firstly, the spatial properties of the graph needed to be chosen. These parameters dictate how many accounts are present in the transaction network, how many ac-

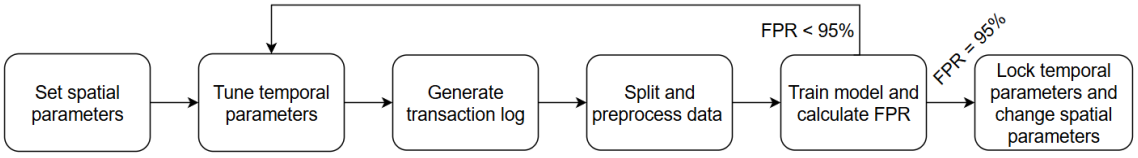


Figure 3.1: Visualisation of the data generation pipeline

counts that are part of normal patterns, and how many accounts that are partaking in money laundering topologies, se Figure 3.4. Also, there is a degree distribution where counts of in- and out-degrees for the accounts are chosen.

With these spatial parameters we are able to choose the class balance of the datasets. Increasing the number of money-laundering topologies will increase the number of money-laundering accounts, and therefore change the class imbalance. For all three of the datasets, a total number of 100 000 accounts was chosen. This number of accounts was deemed enough to be relevant while still being manageable. The size was manageable to handle whilst being sufficient to demonstrate a broad and complex transaction network. For the medium dataset a class imbalance of 5:95 was chosen. The graph was built as a scale-free network, following the power-law degree distribution, with a gamma of 2.0 [53]. The reasoning for modeling this as a scale-free network lies in its ability to more accurately represent real-world transaction networks, where a small number of nodes (accounts) tend to have a large number of connections (transactions), while the majority have relatively few.

Next, the temporal parameters for each of the classes in AMLSim need to be chosen. These parameters control the distributions of transaction properties from accounts to accounts, income to accounts, and outcome from accounts. These temporal parameters are found in table 3.1. Internal transactions in AMLSim, are mimicing Swish transactions from an account in the network to another account in the network. Income in AMLSim, is treated as transactions from a source node to an account in the transaction network. This allows for money to flow into the network. Accounts have a probability to receive income from the source node each simulated time step. Outcome in AMLSim, is treated as accounts sending a transaction to the sink node on the transaction network. The sink node serves the purpose of money leaving the network, such as accounts spending money at businesses outside of the

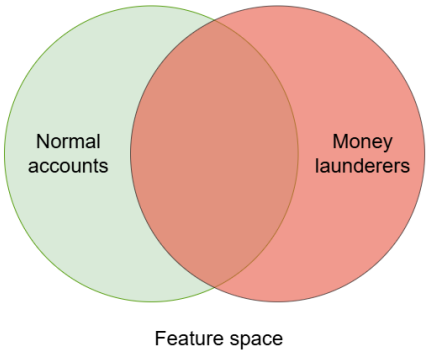


Figure 3.2: Visualisation of the two classes overlap in feature space

network. The probability of an outcome transaction to the sink node is controlled by the spending behavior. Spending behavior refers to the pattern where accounts are more inclined to make transactions to the sink node as their available balance increases. This tendency can be mathematically represented by a sigmoid function.

As aforementioned, real-life behavioral parameters are highly confidential and only known by banks and financial authorities. Because of this, these parameters are not readily available. To circumvent this issue, a reverse engineered iterative process was devised to find temporal parameters that reflects the performance of practical AML systems. The paper by Copenhagen Business School [21] states that current if-based systems exhibit a false positive rate (FPR) of about 95%-98% on real data. Inspired by this, the idea for the temporal parameters was to be tuned towards a model that performs at a 95% FPR, demonstrated in the figure 3.1.

Tuning the temporal parameters for the classes to be more similar results in data with a larger overlap in feature space, thus making them less distinguishable. Conversely, tuning the temporal parameters to be less similar creates data where the classes are easier to distinguish, as conceptually illustrated in Figure 3.2.

With the spatial parameters set, and temporal parameters tuned, the transaction log (TxLog) can be generated. The TxLog is generated from a discrete-time multi-agent simulation where, in each time step, each agent (account) gets the opportunity to perform transactions with other accounts or with the source/sink. All of the transactions are written into the log file including initial balances, transaction size, account ids, etc. Hence, the TxLog will describe a spatio-temporal directed graph

Label	Parameter	Description
(A)	`min_amount`	Global minimum transaction amount (shared)
(B)	`max_amount`	Global maximum transaction amount (shared)
Class-specific Parameters		
A	`mean_amount`	Mean transaction amount
B	`std_amount`	Standard deviation of transaction amount
C	`prob_income`	Probability of income transactions
D	`mean_income`	Mean income amount
E	`std_income`	Standard deviation of income amount
F	`mean_outcome`	Mean outcome amount
G	`std_outcome`	Standard deviation of outcome amount
H	`mean_phone_change_freq`	Mean phone number change frequency
I	`std_phone_change_freq`	Standard deviation of phone number change frequency

Table 3.1: List of the temporal parameters used in AMLSim with their descriptions. Parameters (A) and (B) are shared between both classes. The remaining parameters are defined separately for each class (Normal and SAR).

with nodes being accounts and edges being transactions. The time-step interval is one day, and the total time for the simulation is set to one year.

From the TxLog, we create the training and test sets through a preprocessing step. This preprocessing involves splitting the data into training and test sets with respect to time, and then by engineering features for each of the splits. For both the training and test sets, a node file and an edge file are created, structuring the data for further analysis and model training.

To create the training and test splits, we divide the TxLog data based a 50/50 split in time. This means, that all transaction activities occurring in the first half of the simulated year are included in the training set, while the activity from the second half of the year is included in the test set. No overlap is used between the training and test sets to minimize data leakage. This approach is further discussed in Section 5.0.2.

After splitting the data based on time to create the training and test sets, feature engineering is performed separately on each split. The label indicating whether an account is laundering money or not is determined based on whether the account is part of a money laundering topology during the specific time split, ensuring the label accurately reflects the account's status within each period.

For both the training and test sets, a node file and an edge file are created. The node files contain the created features for each account, such as the total amount spent by an account, the number of incoming transactions, and other account-specific features. All the node features are shown and further explained in Table 3.2. The edge file represents transactions between accounts, including the source. The edges are undirectional, meaning the direction of the transaction is not considered in the graph representation.

From the training and test datasets we are now able to train a model. In this thesis we are treating the classification on a node level, i.e., we are trying distinguish between accounts partaking in money-laundering activities from normal accounts. Since the rule-based system in the paper by Copenhagen Business School [21] that performs at a 95-98% FPR is unknown, we opted to use a tree-based classifier as a replacement. This tree-based classifier offers a more sophisticated method for classifying money laundering compared to the rule-based system. Consequently, this approach results in a dataset that is likely more challenging than real-data. The tree-based classifier is only able to use tabular data which, in this case, corresponds to the node features previously been described.

With the training and test data in place, a classifier is trained and its FPR is compared to the target. If the target is not achieved, the iterative process restarts, as illustrated in Figure 3.1. In particular, if the model achieves lower than 95% FPR, the temporal parameters are updated to attempt to get closer to the target in the next iteration.

If the tree-based classifier achieves 95% FPR or higher, we can assure that the the two classes in the data are extremely difficult to distinguish between, based on the node features, and the benchmarking dataset is now completed. Ultimately, this procedure results in a dataset that mimics that of real-data in a more systematic

way than guessing the parameters.

From this point, the temporal parameters, see in Table 3.3, are locked. This benchmarking dataset will be called the medium level dataset (MID). From the MID dataset, two other datasets are generated: an easy (EASY) and one more difficult (HARD), distinguished by different class imbalances. The datasets were generated by locking the temporal parameters, and tuning the spatial parameters to increase the number of money-laundering accounts for the EASY dataset to approximately 25%, and decreasing the number of money-laundering accounts for the HARD dataset to approximately 1%. These transaction graph networks, EASY and HARD, with their respective spatial parameters go through the pipeline of generating the transaction log, splitting and preprocessing the data. Thus, finalizing the three datasets, EASY, MID and HARD, with respective node and edge files for both the training a test sets.

3.0.2 Missing labels

In this thesis, we are assuming that a substantial portion of the data labels are unknown. This assumption is fair since there is limited feedback on the cases that are sent to the financial authorities, and not a certain ground truth for the non-suspicious accounts. To strenghten this argument, we also note that only about 5% of all financial crimes are found by today's systems [21]. Therefore it is fair to assume, that if all accounts in a transaction network were labeled, a large amount of these could be mislabeled. From these facts and estimations, we opted for a heavily semisupervised setting where 90% of the accounts do not have a label in the training data. To realize this, in each of the three datasets (EASY, MID, and HARD), 90% of all normal nodes, and 90% of all money-laundering nodes have their labels removed. Nodes whose labels are removed are selected at random. This procedure results in an additional three datasets, EASY with missing label, MID with missing labels, and HARD with missing labels.

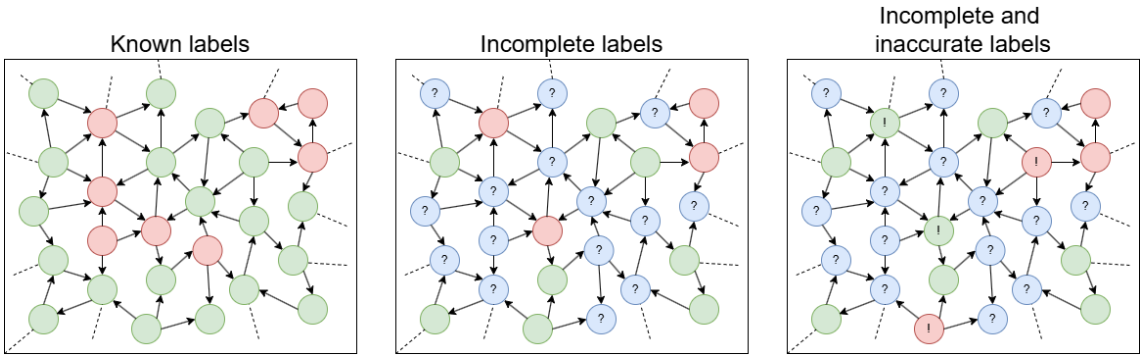


Figure 3.3: Example of the differences between known labels, incomplete labels and incomplete and inaccurate labels. Blue nodes with a "?" are incompletely labeled, and nodes with a "!" are inaccurately labeled.

3.0.3 Inaccurate labeling

To investigate the importance of label accuracy, i.e., the correctness of the labels available, we introduce label noise to the nodes that have a label attached, demonstrated in Figure 3.3. To address realistic scenarios, we identified three different categories of methods to select labels: Class, topology, and neighbour.

We consider two different ratios, 10% and 25%, which translates into the fraction of the selected *labeled* nodes, that have their labels flipped.

The selection categories contain different noises that will be applied to the train-set of the three datasets, EASY, MID, and HARD, at both of the ratios of 10%, and 25%, which implies, that for every noise, six more datasets will be generated.

3.0.3.1 Class

In real data, inaccuracies can occur due to accounts being falsely flagged by a detection system or by illicit accounts that have not been found. The class selection category introduces label noise by separately handling normal accounts and money-laundering accounts. The **False positive** noise selects all the normal accounts, and flips 10% and 25% of their labels, and the **False negative** noise selects all the money-laundering accounts and flips 10% and 25% of their labels. From this type of noise, one may study the importance of labeling and whether one class is more important than the other.

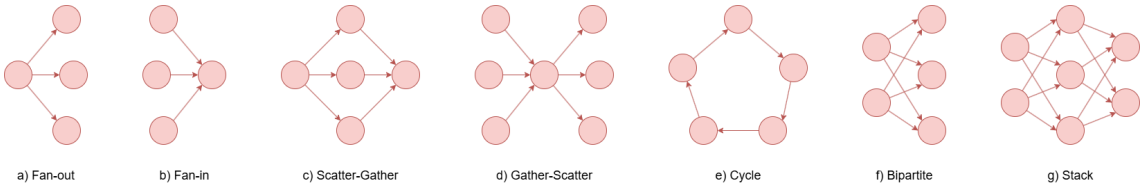


Figure 3.4: The different money laundering topologies

3.0.3.2 Topology

Money launderers funnel illicit funds in many different ways, so called topologies, as can be seen in Figure 3.4. The topology selection category introduces noise in seven different ways, all following the same selection principle. In particular, we select all the nodes of every money-laundering topology at a given time and flip 10% and 25% of their labels, thus creating noises called **Fan-out**, **Fan-in**, **Scatter-Gather**, **Gather-Scatter**, **Cycle**, **Bipartite**, and **Stack**.
This is done to analyze if the models are more sensitive to inaccurately labeled nodes in certain money-laundering topologies.

3.0.3.3 Neighbour

In a real world scenario, accounts that have engaged in transactions with money laundering accounts may have a higher possibility to also be flagged as a money launderer. We denote this noise as **Neighbour**. It is created by selecting all normal neighbouring nodes to a money laundering typology whereafter 10% and 25% of their labels are flipped. This is done to analyze if inaccurately labeling neighbouring normal accounts as money-launderers has a significant effect on the performance of the models or not.

3.0.4 Statistical machine learning models

In this thesis, five widely used statically machine learning models were considered, i.e., SVM, KNN, LOG, Random Forest, and XGBoost, as outlined in section 2.4. Notably, these models are unable to utilize the graph structure of the data. Instead, these models are only able to use the node features as independent data-points.

Therefore, it is highly interesting to compare the results of the statistical machine learning models with the GNNs that are able to use the graph structure to try and understand how the incomplete and inaccurate datasets are effecting the different models.

3.0.4.1 Model implementation

All five statistical models were implemented using the Scikit-learn library in python [54]. This is a widely used library used for classification, regression, clustering, etc. It enables easy implementations of various statically machine learning models in an easy to use environment.

3.0.4.2 Data loading

Each of the models were trained on the node files for every dataset. Prior to training, all features were normalized using the standard scaler, which removes the mean and scales to unit variance, to ensure proper function of the models.

3.0.4.3 Optimization

All five statistical models were optimized with the grid search technique. Grid search works by defining a search space as a grid of hyper-parameter values. It evaluates every combination of the hyper-parameters in the grid to find the best combination [55]. The performance metric optimized was Average precision (AP), this will be explained in the next section. The hyper-parameters were tuned for the EASY, MID and HARD datasets.

3.0.4.4 Evaluation

In evaluating the performance of the selected statistical machine learning models, the thesis focused on two key metrics: AP for the positive label and the Area Under the Curve (AUC) score. These metrics were chosen to provide a comprehensive view of model effectiveness. AP offers a focused measure of the models' ability to identify accounts that are laundering money, which are critical to the project's objectives, while the AUC score provides a broader assessment of overall model performance

across all classes. To calculate the AUC score, we first determine the True Positive Rate

$$\text{TPR} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}, \tag{3.1}$$

and False Positive Rate

$$\text{FPR} = \frac{\text{False positives}}{\text{False positives} + \text{True negatives}}, \tag{3.2}$$

at various thresholds.

These values are used to plot the Receiver Operating Characteristic (ROC) curve. The area under this curve quantifies the model's AUC score **fawcett2006ROCAUC**. Since there is no randomness in fitting the statistical machine learning models to the training data, no cross-validation was performed. Since the data is split in time there is no intuitive way of generating cross-validation datasets. Therefore, in the results for the statistical machine learning models, there are only scores without standard deviation.

3.0.4.5 Handling incomplete labels

Since all nodes in the datasets are interpreted as independent by the statistical machine learning models, the only intuitive way of handling the incomplete labeled nodes is to not use them. Not being able to use the node features of nodes with incomplete labels is a significant disadvantage compared to the GNNs.

3.0.5 GNNs

For this thesis, three well-established GNN architectures were selected: GCN, GraphSAGE, and GAT. These models were chosen for their demonstrated success and versatility in handling graph based learning tasks across various domains [42], [46], [47]. Each model offers a distinct approach to processing graph-structured data, which is crucial for addressing the challenges presented by datasets with missing labels and label noise. The selection of GCN, GraphSAGE, and GAT allows for a detailed comparison of their performance in managing these specific data irregularities, providing valuable insights into their effectiveness and robustness under varying conditions.

3.0.5.1 Model implementation

All three GNN models were implemented using the PyTorch Geometric library [56], [57]. This library is a specialized extension of PyTorch, tailored specifically for the creation and operation of graph neural networks. PyTorch Geometric provides efficient data structures and APIs for managing complex graph data, facilitating the development and application of advanced GNN architectures.

3.0.5.2 Data loading

In the thesis, data preparation entails a series of steps to ready the graph data for analysis using the selected GNNs. A specialized dataset handling module is used to manage both node and edge data, ensuring that the models receive structured and relevant information. Data is loaded from CSV files containing node and edge information. This includes incorporating node features when specified and applying labels where available, whether to nodes or edges.

It should be noted that in the current setup, the GNN models do not utilize the edge features nor operate on directed edges. This approach simplifies the model's architecture and focuses the learning on node features and their relationships, which are represented as undirected edges.

The data for nodes includes features and, possibly, labels, loaded and normalized to improve model training efficiency. Normalization involves subtracting the mean and then dividing by the standard deviation of the training data's features, a common practice to scale the input features to a similar range, reducing the possibility of instability during training due to vastly different feature scales.

3.0.5.3 Optimization

To enhance the performance of the GNN models for detecting positive labels in the data, the hyperparameters were optimized using the Tree-structured Parzen Estimator (TPE) method [58]. This method predicts which hyperparameter configurations are likely to yield better results by analyzing outcomes from past trials.

As for the statistical machine learning models, AP was used as the objective func-

tion for the positive label across different datasets. TPE was deployed to optimize each model individually for each dataset, adjusting key parameters such as learning rate, dropout rates, and the number of attention heads for attention-based models. For each dataset, 30 trials were conducted, with each trial configuration undergoing a training process spanning 1000 epochs to thoroughly evaluate the model's performance.

3.0.5.4 Evaluation

The evaluation of the selected GNN models will follow the same principle as for the statistical models, described in section 3.0.4.4, to be able to compare the different methods.

To ensure the robustness and reliability of the evaluation results, each GNN model underwent multiple runs under different conditions. Specifically, each model was trained and tested five times with random initializations. This approach mitigates any potential biases or anomalies that could arise from a single training session and allows for a more generalized assessment of the models' performance.

For each of the five runs, both the AP for the positive label and the overall AUC score were calculated. The results from these runs were then aggregated to compute an average AP and average AUC, along with the standard deviation (std) for these metrics. This evaluation approach not only demonstrates each model's effectiveness in identifying money laundering accounts but also reliably measures their overall performance.

3.0.5.5 Handling incomplete labels

The GNN models can effectively handle partially labeled datasets. Specifically, due to the neighbor aggregation mechanism illustrated in Figure 2.2, these models can leverage the features of nodes with missing labels, making them advantageous for datasets with incomplete labels.

A possible way of incorporating the nodes without a label in the loss function is by means of label propagation, i.e., creating pseudo labels via majority voting between neighbours [59]. However, experiments yielded poor results since most of the neigh-

bouring nodes are normal account, hence, voting also money laundering accounts to be labeled as normal. Therefore, nodes without a label was not used in the loss function.

Feature	Description
sum_spending	Total amount spent by an account.
mean_spending	Average amount spent by an account.
median_spending	Median amount spent by an account.
std_spending	Standard deviation of amounts spent by an account.
max_spending	Maximum amount spent in a single transaction.
min_spending	Minimum amount spent in a single transaction.
count_spending	Total number of outgoing transactions.
sum	Total transaction amount for an account.
mean	Average transaction amount for an account.
median	Median transaction amount for an account.
std	Standard deviation of transaction amounts.
max	Maximum transaction amount for an account.
min	Minimum transaction amount for an account.
count_in	Number of incoming transactions.
count_out	Number of outgoing transactions.
count_unique_in	Number of unique counterparties sending transactions.
count_unique_out	Number of unique counterparties receiving transactions.
count_days_in_bank	Number of days an account has been active.
count_phone_changes	Number of phone number changes for an account.
Label	**Description**
money_launderer	Indicator if an account is laundering money.

Table 3.2: List of node features and label their descriptions.

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
Normal	637	500	0.1	400	250	200	500	1460	365
SAR	850	700	0.1	500	400	300	650	1000	200

Table 3.3: Parameter values for normal and money-laundering classes. *A* = mean_amount, *B* = std_amount, *C* = prob_income, *D* = mean_income, *E* = std_income, *F* = mean_outcome, *G* = std_outcome, *H* = mean_phone_change_freq, *I* = std_phone_change_freq.

4

Results

4.1 The finished datasets

Tables 4.1,4.2, and 4.3 display all key properties of all datasets. Class imbalance is shown as percentage of normal : percentage of money launderers.

Table 4.1: Data analysis from the EASY data

Dataset	Total Accounts	Labeled Accounts	Normal Accounts	ML Accounts	Class Imbalance	Inaccurate Labels
Ratio: No noise						
Known labels	99 938	99 938	85 120	14 818	82.6:17.4	0
Missing labels	99 938	9 994	8 512	1 482	82.6:17.4	0
Ratio: 10%						
Neighbour	99 938	9 994	8 197	1 797	78.1:21.9	315
Cycle	99 938	9 994	8 540	1 454	83.0:17.0	28
Fan-out	99 938	9 994	8 537	1 457	82.9:17.1	25
Fan-in	99 938	9 994	8 538	1 456	82.9:17.1	26
FP	99 938	9 994	7 661	2 333	69.6:30.4	851
Bipartite	99 938	9 994	8 533	1 461	82.9:17.1	21
Gather-scatter	99 938	9 994	8 524	1 470	82.8:17.2	12
Scatter-gather	99 938	9 994	8 528	1 466	82.8:17.2	16
Stack	99 938	9 994	8 532	1 462	82.9:17.1	20
FN	99 938	9 994	8 660	1 334	84.6:15.4	148
Ratio: 25%						
Neighbour	99 938	9 994	7 724	2 270	70.6:29.3	788
Cycle	99 938	9 994	8 581	1 413	83.5:16.5	69
Fan-out	99 938	9 994	8 575	1 419	83.5:16.5	63
Fan-in	99 938	9 994	8 577	1 417	83.5:16.5	65
FP	99 938	9 994	6 384	3 610	43.5:56.5	2 128
Bipartite	99 938	9 994	8 564	1 430	83.3:16.7	52
Gather-scatter	99 938	9 994	8 542	1 452	83.0:17.0	30
Scatter-gather	99 938	9 994	8 552	1 442	83.1:16.9	40
Stack	99 938	9 994	8 562	1 432	83.3:16.7	50
FN	99 938	9 994	8 882	1 112	87.5:12.5	370

Table 4.2: Data analysis from the MID data

Dataset	Total Accounts	Labeled Accounts	Normal Accounts	ML Accounts	Class Imbalance	Inaccurate Labels
Ratio: No noise						
Known labels	99 932	99 932	97 032	2 900	97.0:3.0	0
Missing labels	99 932	9 993	9 703	290	97.0:3.0	0
Ratio: 10%						
Neighbour	99 932	9 993	9 625	368	96.2:3.8	78
Cycle	99 932	9 993	9 709	284	97.1:2.9	6
Fan-out	99 932	9 993	9 708	285	97.1:2.9	5
Fan-in	99 932	9 993	9 708	285	97.1:2.9	5
FP	99 932	9 993	8 733	1 260	85.6:14.4	970
Bipartite	99 932	9 993	9 708	285	97.1:2.9	5
Gather-scatter	99 932	9 993	9 705	288	97.0:3.0	2
Scatter-gather	99 932	9 993	9 706	287	97.0:3.0	3
Stack	99 932	9 993	9 707	286	97.1:2.9	4
FN	99 932	9 993	9 732	261	97.3:2.7	29
Ratio: 25%						
Neighbour	99 932	9 993	9 508	485	94.9:5.1	195
Cycle	99 932	9 993	9 719	274	97.2:2.8	16
Fan-out	99 932	9 993	9 715	278	97.1:2.9	12
Fan-in	99 932	9 993	9 715	278	97.1:2.9	12
FP	99 932	9 993	7 277	2 716	62.7:37.3	2 426
Bipartite	99 932	9 993	9 715	278	97.1:2.9	12
Gather-scatter	99 932	9 993	9 708	285	97.1:2.9	5
Scatter-gather	99 932	9 993	9 710	283	97.1:2.9	7
Stack	99 932	9 993	9 712	281	97.1:2.9	9
FN	99 932	9 993	9 775	218	97.8:2.2	72

Table 4.3: Data analysis from the HARD data

Dataset	Total Accounts	Labeled Accounts	Normal Accounts	ML Accounts	Class Imbalance	Inaccurate Labels
Ratio: No noise						
Known labels	99 923	99 923	99 344	579	99.4:0.6	0
Missing labels	99 923	9 992	9 934	58	99.4:0.6	0
Ratio: 10%						
Neighbour	99 923	9 992	9 918	74	99.3:0.7	16
Cycle	99 923	9 992	9 935	57	99.4:0.6	1
Fan-out	99 923	9 992	9 935	57	99.4:0.6	1
Fan-in	99 923	9 992	9 935	57	99.4:0.6	1
FP	99 923	9 992	8 941	1 051	88.2:11.8	993
Bipartite	99 923	9 992	9 935	57	99.4:0.6	1
Gather-scatter	99 923	9 992	9 934	58	99.4:0.6	0
Scatter-gather	99 923	9 992	9 934	58	99.4:0.6	0
Stack	99 923	9 992	9 935	57	99.4:0.6	1
FN	99 923	9 992	9 940	52	99.5:0.5	6
Ratio: 25%						
Neighbour	99 923	9 992	9 894	98	99.0:1.0	40
Cycle	99 923	9 992	9 936	56	99.4:0.6	2
Fan-out	99 923	9 992	9 936	56	99.4:0.6	2
Fan-in	99 923	9 992	9 937	55	99.4:0.6	3
FP	99 923	9 992	7 450	2 542	65.9:34.1	2 484
Bipartite	99 923	9 992	9 936	56	99.4:0.6	2
Gather-scatter	99 923	9 992	9 935	57	99.4:0.6	1
Scatter-gather	99 923	9 992	9 935	57	99.4:0.6	1
Stack	99 923	9 992	9 937	55	99.4:0.6	3
FN	99 923	9 992	9 948	44	99.6:0.4	14

4.2 Comparison of known and missing labels

Table 4.4: AP and AUC Scores for Different Datasets and Label Conditions

Model	AP Scores					
	EASY		MID		HARD	
	Known labels	Missing labels	Known labels	Missing labels	Known labels	Missing labels
KNN	42.16	37.11	13.03	8.98	1.77	0.84
LOG	37.32	34.52	10.60	10.80	1.93	1.35
RF	52.69	48.82	23.17	18.75	8.30	3.19
SVM	34.91	27.84	3.09	5.69	0.73	0.94
XGB	53.53	51.00	24.61	18.06	7.83	2.57
GSAGE	55.16 ± 0.42	40.84 ± 2.01	25.56 ± 1.25	5.57 ± 0.95	6.54 ± 1.40	1.84 ± 0.52
GCN	40.57 ± 3.69	44.98 ± 0.86	17.06 ± 0.74	16.13 ± 0.56	5.31 ± 0.62	4.26 ± 0.17
GAT	55.30 ± 0.34	54.31 ± 0.56	28.08 ± 1.52	23.40 ± 2.09	8.86 ± 0.73	7.90 ± 0.73
	AUC Scores					
	Known labels	Missing labels	Known labels	Missing labels	Known labels	Missing labels
KNN	79.93	76.06	74.53	70.08	62.96	56.17
LOG	78.23	75.82	74.75	73.24	73.63	68.39
RF	86.12	85.14	86.36	83.36	85.27	77.18
SVM	74.78	66.22	47.54	64.30	49.77	62.14
XGB	86.32	85.48	86.51	84.53	85.70	78.41
GSAGE	86.27 ± 0.15	77.09 ± 1.22	85.86 ± 0.31	62.04 ± 5.17	78.88 ± 2.46	60.67 ± 2.72
GCN	82.97 ± 0.47	83.31 ± 0.17	84.86 ± 0.06	82.78 ± 0.51	80.61 ± 0.45	79.58 ± 0.30
GAT	86.25 ± 0.05	85.57 ± 0.16	87.10 ± 0.13	86.09 ± 0.44	85.27 ± 0.29	83.89 ± 1.32

In table 4.4 the results for all models regarding both known labels and missing labels on the EASY, MID, and HARD datasets are shown. For the EASY dataset, GAT generally shows superior performance compared to other models, achieving the highest AP scores for both known labels (55.30) and missing labels (54.31). GraphSAGE shows competitive performance with AP scores of 55.16 for known labels and 40.84 for missing labels, indicating a significant drop but still maintaining a robust performance relative to other models. GCN also shows strong results with an AP score of 40.57 for known labels and 44.98 for missing labels, showcasing better robustness to missing labels compared to GraphSAGE. XBG follows closely with an AP score of 53.53 for known labels and 51.00 for missing labels, showing slightly

lower robustness compared to GAT. RF performs well among statistical models with AP scores of 52.69 for known labels and 48.82 for missing labels. SVM and LOG show relatively lower performance, particularly SVM, with a notable drop from 34.91 (known labels) to 27.84 (missing labels). The overall performance decreases slightly in the presence of missing labels across all models, but GAT maintains the highest robustness. In terms of AUC scores, XBG shows the highest AUC for known labels (86.32), but GAT performs almost equally well (86.25). For missing labels, GAT (85.57) and XBG (85.48) remain very close, with GAT showing slightly better robustness.

For the MID dataset, GAT again leads with an AP score of 28.08 for known labels and 23.40 for missing labels. GraphSAGE shows competitive performance with AP scores of 25.56 for known labels and 5.57 for missing labels, indicating a significant drop. GCN also shows strong results with an AP score of 17.06 for known labels and 16.13 for missing labels, showcasing better robustness to missing labels compared to GraphSAGE. XBG scores 24.61 for known labels and 18.06 for missing labels, demonstrating a significant drop when labels are missing. Random Forest shows close results to XBG with an AP score of 23.17 for known labels and 18.75 for missing labels. SVM has a lower result with known labels than for missing labels with an AP performance from 3.09 for known labels to 5.69 for missing labels. LOG scores 10.60 for known labels and 10.80 for missing labels, indicating a slight improvement with missing labels. In terms of AUC scores, GAT performs best with AUC scores of 87.10 for known labels and 86.09 for missing labels. XBG also performs closely in terms of AUC score with 86.51 for known labels and 84.53 for missing labels.

For the HARD dataset, GAT continues to outperform other models with AP scores of 8.86 for known labels and 7.90 for missing labels. GraphSAGE shows competitive performance with AP scores of 6.54 for known labels and 1.84 for missing labels, indicating a significant drop. GCN also shows strong results with an AP score of 5.31 for known labels and 4.26 for missing labels, showcasing better robustness to missing labels compared to GraphSAGE. Random Forest shows a significant drop in AP performance from 8.30 for known labels to 3.19 for missing labels. XBG scores 7.83 for known labels and 2.57 for missing labels, demonstrating a notable drop.

LOG and KNN perform the worst in terms of AP scores, with KNN scoring 1.77 for known labels and 0.84 for missing labels. SVM performs poorly, but still better than KNN in some cases, with scores of 0.73 for known labels and 0.94 for missing labels. In terms of AUC scores, GAT maintains the highest AUC for missing labels with 85.27 for known labels and 83.89 for missing labels, showing strong robustness. XBG also shows the highest performance for known labels with 85.70 but drops to 78.41 for missing labels. Overall, we can see that the AP scores for all models are low regarding the HARD dataset.

There is a clear decline in both AP and AUC scores as the dataset becomes more challenging (from EASY to HARD), with all models showing reduced performance with missing labels. GAT consistently shows the best performance across all difficulty levels and conditions, particularly excelling in robustness to missing labels. Among statistical models, Random Forest and XBG generally perform best, but they still lag behind GAT in robustness and overall performance. SVM shows the lowest performance for the MID and HARD datasets, while KNN performs the worst in the HARD missing labels scenario.

In summary, GAT demonstrates the highest performance and robustness across all scenarios, particularly excelling with incomplete labels. Among statistical models, Random Forest and XBG show the highest performance but still fall behind GAT in robustness. Overall, the presence of missing labels affects the performance of all models, with a notable decline in both AP and AUC scores as dataset difficulty increases.

4.3 Model performance for incorrect labels

Table 4.5: Results of EASY datasets with 10% of selected nodes label flipped

	AP Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	37.11	35.54	32.81	34.75	36.91	37.50
LOG	34.52	33.98	35.76	34.01	34.31	34.74
RF	48.82	48.41	48.12	48.95	48.36	49.03
SVM	27.84	20.32	21.23	25.71	26.08	26.56
XGB	51.00	49.86	45.23	50.24	50.65	51.54
GSAGE	40.84 ± 2.01	43.15 ± 4.07	43.10 ± 3.31	46.25 ± 1.21	39.40 ± 3.82	36.46 ± 2.32
GCN	44.98 ± 0.86	42.55 ± 1.26	43.87 ± 3.15	46.44 ± 0.33	41.03 ± 3.88	43.57 ± 2.44
GAT	54.31 ± 0.56	53.93 ± 0.21	53.24 ± 0.29	54.37 ± 0.15	51.35 ± 4.75	54.09 ± 0.52

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	37.11	37.17	37.10	36.92	37.13	37.02
LOG	34.52	34.47	34.08	34.66	34.56	34.28
RF	48.82	48.92	48.85	48.95	48.95	48.44
SVM	27.84	28.68	29.24	27.31	27.22	28.24
XGB	51.00	51.20	50.53	50.74	50.72	50.46
GSAGE	40.84 ± 2.01	38.22 ± 4.15	38.90 ± 2.89	36.96 ± 3.19	40.60 ± 5.95	39.66 ± 5.47
GCN	44.98 ± 0.86	43.35 ± 2.53	44.49 ± 0.56	43.10 ± 2.95	40.97 ± 3.01	44.01 ± 1.79
GAT	54.31 ± 0.56	51.49 ± 4.37	53.61 ± 0.23	54.06 ± 0.44	53.43 ± 0.60	53.66 ± 0.31

	ROC_AUC Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	76.06	75.05	70.70	74.51	75.90	76.23
LOG	75.82	75.45	78.84	76.30	75.64	75.79
RF	85.14	84.91	84.26	84.94	84.95	85.15
SVM	66.22	59.52	58.05	65.98	64.62	65.85
XGB	85.48	85.20	83.98	85.04	85.45	85.55
GSAGE	77.09 ± 1.22	79.32 ± 2.79	77.83 ± 2.95	81.21 ± 1.23	76.18 ± 2.70	74.00 ± 1.89
GCN	83.31 ± 0.17	82.80 ± 0.22	83.03 ± 0.28	83.13 ± 0.09	82.64 ± 0.64	83.06 ± 0.29
GAT	85.57 ± 0.16	85.56 ± 0.17	84.43 ± 0.30	85.66 ± 0.17	85.30 ± 0.72	85.72 ± 0.18

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	76.06	76.04	76.06	75.96	75.92	76.00
LOG	75.82	75.71	75.41	76.06	75.84	75.64
RF	85.14	85.07	85.07	85.08	85.12	84.99
SVM	66.22	66.57	67.35	65.63	65.84	66.60
XGB	85.48	85.50	85.35	85.45	85.46	85.32
GSAGE	77.09 ± 1.22	75.73 ± 2.82	75.60 ± 2.51	74.37 ± 2.87	76.54 ± 4.44	75.90 ± 4.34
GCN	83.31 ± 0.17	83.11 ± 0.27	83.16 ± 0.14	82.99 ± 0.35	82.73 ± 0.42	83.13 ± 0.23
GAT	85.57 ± 0.16	85.39 ± 0.51	85.57 ± 0.07	85.79 ± 0.25	85.68 ± 0.25	85.80 ± 0.12

4.3.1 EASY with 10% flipped labels

For the EASY dataset with 10% of selected nodes labels flipped shown in table 4.5, GAT demonstrates the highest performance across most scenarios. GAT consistently achieves the highest AP scores for both missing labels and various types of inaccurate labels. It also maintains robust AUC scores, indicating its strong overall performance and robustness to label inaccuracies.

XGB follows closely with high AP and AUC scores but shows less robustness to label flipping compared to GAT. While it performs well, its sensitivity to inaccurate labels results in a more noticeable performance decline.

RF also performs well among the statistical models. It achieves competitive AP and AUC scores, although it too is affected by label inaccuracies, showing some performance decline with increased label flipping.

GraphSAGE shows competitive performance with moderate drops in AP and AUC scores when labels are flipped. It remains robust, but the impact of label inaccuracies is more pronounced compared to GAT.

GCN also shows strong results with respectable AP and AUC scores. Like GraphSAGE, GCN is affected by label inaccuracies but maintains competitive performance levels.

SVM and LOG show relatively lower performance. Both models experience significant declines in AP and AUC scores as label inaccuracies increase, with SVM being particularly sensitive.

KNN exhibits moderate performance, showing noticeable declines in both AP and AUC scores with label inaccuracies. Its performance is less robust compared to the other models evaluated.

Table 4.6: Results of EASY datasets with 25% of selected nodes label flipped

	AP Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	37.11	33.58	26.45	32.25	36.50	36.80
LOG	34.52	32.77	32.93	32.74	33.97	34.60
RF	48.82	48.80	45.59	47.80	48.07	49.37
SVM	27.84	21.63	16.99	25.45	25.79	22.74
XGB	51.00	48.94	44.33	50.40	50.26	50.86
GSAGE	40.84 ± 2.01	46.91 ± 4.78	38.66 ± 2.70	40.22 ± 1.01	33.86 ± 3.64	37.34 ± 3.33
GCN	44.98 ± 0.86	43.62 ± 0.75	46.69 ± 0.56	41.58 ± 4.11	42.98 ± 1.83	41.65 ± 3.50
GAT	54.31 ± 0.56	52.15 ± 0.85	51.96 ± 0.63	54.20 ± 0.69	53.70 ± 0.40	52.35 ± 3.08

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	37.11	36.99	37.02	37.36	37.36	36.85
LOG	34.52	34.05	34.04	34.42	34.42	34.62
RF	48.82	49.18	48.95	48.40	48.97	48.00
SVM	27.84	24.26	26.61	26.59	26.59	22.27
XGB	51.00	51.20	50.72	50.22	50.95	49.31
GSAGE	40.84 ± 2.01	40.51 ± 5.96	38.59 ± 4.76	40.91 ± 3.19	40.73 ± 5.42	35.65 ± 5.11
GCN	44.98 ± 0.86	41.82 ± 2.67	41.92 ± 4.31	41.03 ± 2.60	42.45 ± 1.88	43.46 ± 2.22
GAT	54.31 ± 0.56	53.95 ± 0.27	54.02 ± 0.46	53.97 ± 0.44	54.16 ± 0.25	52.49 ± 1.43

	ROC_AUC Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	76.06	73.58	66.02	73.08	75.62	75.78
LOG	75.82	75.16	77.06	75.89	75.25	75.78
RF	85.14	84.82	83.49	84.39	84.91	85.22
SVM	66.22	62.41	54.26	66.13	64.25	62.62
XGB	85.48	84.87	82.97	84.43	85.29	85.44
GSAGE	77.09 ± 1.22	81.84 ± 3.15	75.79 ± 2.05	76.11 ± 1.69	71.93 ± 3.99	74.64 ± 2.60
GCN	83.31 ± 0.17	83.11 ± 0.15	83.39 ± 0.18	82.59 ± 0.57	82.87 ± 0.29	82.81 ± 0.49
GAT	85.57 ± 0.16	85.43 ± 0.13	83.88 ± 0.64	85.59 ± 0.25	85.54 ± 0.08	85.48 ± 0.26

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	76.06	75.96	75.93	76.07	75.96	75.83
LOG	75.82	75.54	75.48	75.81	75.81	76.04
RF	85.14	85.08	85.12	85.07	85.07	84.95
SVM	66.22	63.17	66.57	65.20	65.20	63.13
XGB	85.48	85.53	85.53	85.32	85.43	85.16
GSAGE	77.09 ± 1.22	76.54 ± 4.54	75.72 ± 4.14	77.37 ± 2.38	77.15 ± 4.61	73.41 ± 4.72
GCN	83.31 ± 0.17	82.89 ± 0.37	82.80 ± 0.61	82.73 ± 0.33	82.99 ± 0.30	83.05 ± 0.16
GAT	85.57 ± 0.16	85.76 ± 0.20	85.62 ± 0.25	85.79 ± 0.25	85.66 ± 0.13	85.70 ± 0.21

4.3.2 EASY with 25% flipped labels

The results for 25% flipped labels are shown in table 4.6. For this dataset GAT still achieves the highest performance across both missing and inaccurate labels scenarios. Despite the increased label flipping, GAT maintains high AP and AUC scores, showcasing its robustness.

XGB continues to perform well but shows greater sensitivity to the increase in label inaccuracies. Its AP and AUC scores decline more noticeably compared to the 10% flip scenario, indicating its lower robustness relative to GAT.

RF also shows a notable decline in performance with increased label flipping. While it still performs well among the statistical models, the impact of inaccurate labels is more evident.

GraphSAGE and GCN continue to show competitive performance, but with more significant drops compared to the 10% flip scenario. Both models experience larger declines in AP and AUC scores, highlighting their sensitivity to higher levels of label inaccuracies.

SVM and LOG continue to show lower performance. Both models exhibit significant performance drops in AP and AUC scores with increased label flipping, reinforcing their sensitivity to label inaccuracies.

KNN shows a notable decline in performance with increased label flipping. Its AP and AUC scores decrease significantly, indicating its lower robustness compared to other models.

4.3.3 Summary for EASY

Across both the 10% and 25% label flip scenarios, GAT consistently achieves the highest performance, maintaining robust AP and AUC scores even as label inaccuracies increase. XGB and RF also perform well, but they show a more significant decline in performance with increasing label flips compared to GAT. GraphSAGE and GCN maintain competitive performance but are more affected by label inaccuracies, particularly in the 25% flip scenario. SVM and LOG continue to show lower performance across all scenarios, with significant drops in both AP and AUC scores

as label inaccuracies increase. KNN exhibits a moderate performance that declines notably with increased label flips.

4.3.4 MID with 10% flipped labels

The results for 10% flipped labels are shown in table 4.7. For this dataset GAT is the best performing model overall. With a 5 times lower rate of money laundering compared to the EASY dataset, GAT drops about half of it's previous performance with respect to the AP metric.
XGB and RF are the best performing statistical models, with respect to both performance metrics.
From the greater class imbalance SVM, KNN, and LOG show a significant drop in performance on the AP metric.
GraphSAGE also shows a large decrease in performance with respect to both performance metrics. Showing a big effect of the increased class imbalance.
GCN shows a drop in performance just below of the performance of XGB and RF.
The noise that has the greatest impact on the models performances is the FP noise. Comparing with the other noises that all seem to have a similar impact on the model performance. GCN is the model that handles the FP noise the best showing just a slight drop in performance for the AP metric and a slight increase in AUC score.

Table 4.7: Results of MID datasets with 10% of selected nodes label flipped

	AP Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	8.98	8.29	4.72	8.65	8.86	9.17
LOG	10.80	10.53	6.89	9.08	10.87	12.14
RF	18.75	17.87	13.20	18.19	18.72	19.49
SVM	5.69	5.42	3.65	4.35	5.00	5.61
XGB	18.06	17.89	11.19	17.50	17.54	17.86
GSAGE	5.57 ± 0.95	6.37 ± 1.42	3.46 ± 0.48	4.11 ± 0.44	5.39 ± 0.74	4.85 ± 0.78
GCN	16.13 ± 0.56	16.71 ± 1.05	15.70 ± 0.94	17.56 ± 1.31	17.29 ± 1.19	16.79 ± 1.12
GAT	23.40 ± 2.09	22.26 ± 1.22	17.44 ± 2.07	22.50 ± 1.28	23.46 ± 1.59	23.35 ± 1.93

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	8.98	8.97	8.89	8.96	8.97	8.80
LOG	10.80	10.83	10.62	10.69	10.64	10.74
RF	18.75	18.43	18.95	19.46	19.34	18.73
SVM	5.69	5.40	6.24	5.65	5.58	5.35
XGB	18.06	17.90	17.94	18.09	17.20	17.74
GSAGE	5.57 ± 0.95	4.93 ± 0.55	5.52 ± 0.47	6.05 ± 0.92	5.31 ± 0.76	4.41 ± 0.19
GCN	16.13 ± 0.56	16.04 ± 0.31	17.06 ± 1.14	17.40 ± 1.20	16.20 ± 0.33	16.72 ± 1.33
GAT	23.40 ± 2.09	22.45 ± 1.45	23.89 ± 1.82	23.84 ± 1.35	24.26 ± 1.72	22.96 ± 2.32

	ROC_AUC Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	70.08	67.74	61.94	70.14	69.75	70.10
LOG	73.24	73.12	67.23	71.48	73.29	73.74
RF	83.36	83.03	75.70	82.50	83.34	83.58
SVM	64.30	63.07	58.56	57.83	62.04	63.87
XGB	84.53	84.43	71.15	82.76	84.55	84.57
GSAGE	62.04 ± 5.17	66.32 ± 6.29	55.61 ± 4.70	56.68 ± 0.84	61.82 ± 4.06	58.67 ± 3.91
GCN	82.78 ± 0.51	83.01 ± 0.19	83.38 ± 0.16	83.60 ± 0.10	82.94 ± 0.17	83.23 ± 0.29
GAT	86.09 ± 0.44	85.94 ± 0.25	77.57 ± 4.04	85.18 ± 0.55	85.90 ± 0.22	85.85 ± 0.22

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	70.08	70.40	69.37	70.07	70.08	69.63
LOG	73.24	73.29	73.22	73.13	73.17	73.22
RF	83.36	83.54	83.66	83.85	83.78	83.30
SVM	64.30	63.87	65.69	64.57	64.02	62.17
XGB	84.53	84.70	84.46	84.53	84.43	84.47
GSAGE	62.04 ± 5.17	60.25 ± 3.70	64.13 ± 1.37	64.77 ± 3.03	61.76 ± 4.58	56.35 ± 1.62
GCN	82.78 ± 0.51	82.12 ± 0.34	83.09 ± 0.43	82.65 ± 0.45	82.69 ± 0.31	82.50 ± 0.43
GAT	86.09 ± 0.44	86.07 ± 0.16	85.79 ± 0.24	86.02 ± 0.39	86.15 ± 0.20	86.09 ± 0.33

4.3.5 MID with 25% flipped labels

The results for 25% flipped labels are shown in table 4.8. Once again the strongest performance is from GAT with respect to both performance metrics. Comparing with the previous 10% flipped labels the general performance of all models decrease in performance across all different noises.

The statistical models SVM, KNN, LOG, and GraphSAGE have the largest performance drops on both metrics. With GraphSAGE showing the greatest decrease for the neighbour noise in particular.

XGB and RF still perform at high levels especially with respect to the AUC score, showing a high overall classification performance for taking both classes in consideration.

Comparing the different noise effects on the models, neighbour is a noise that with 25% flipped labels have a large impact on all models. FP is still the noise with the largest impact on the performance for all models.

4.3.6 Summary for MID

GAT consistently performs well across both the 10% and 25% flipped labels of various noises. XGB and RF also show little to no performance drop over the various noises with respect to both performance metrics for the two different ratios. SVM, KNN, and LOG are all consistently showing poor performance with accurate labeling, showing that the increases gap in class imbalance has a great negative effect, and also showing a slight decrease in performance with added inaccurate labeling. The FP and neighbour noises are the most detrimental to the performance of all models, with the largest effects on 25%.

Table 4.8: Results of MID datasets with 25% of selected nodes label flipped

	AP Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	8.98	8.38	3.99	8.21	8.84	8.97
LOG	10.80	10.75	5.01	7.91	10.36	10.52
RF	18.75	17.90	6.10	14.46	18.79	18.90
SVM	5.69	3.76	3.01	3.92	5.41	5.44
XGB	18.06	17.92	7.46	16.66	16.67	18.17
GSAGE	5.57 ± 0.95	4.61 ± 0.92	5.71 ± 1.54	3.37 ± 0.29	6.64 ± 1.00	5.72 ± 0.22
GCN	16.13 ± 0.56	16.00 ± 0.82	14.74 ± 0.71	16.86 ± 1.04	16.62 ± 0.26	16.86 ± 1.17
GAT	23.40 ± 2.09	21.94 ± 0.87	15.67 ± 4.10	18.26 ± 3.10	22.43 ± 1.97	24.22 ± 2.26

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	8.98	8.49	8.95	8.91	8.61	8.78
LOG	10.80	11.67	10.32	10.77	10.75	10.49
RF	18.75	18.63	19.12	19.10	18.20	18.78
SVM	5.69	4.62	5.36	5.37	4.43	5.98
XGB	18.06	17.38	18.42	17.67	17.83	16.85
GSAGE	5.57 ± 0.95	5.09 ± 0.45	4.73 ± 0.51	5.48 ± 0.93	4.96 ± 0.45	5.54 ± 1.11
GCN	16.13 ± 0.56	16.61 ± 0.99	16.27 ± 0.46	16.58 ± 1.18	15.83 ± 0.17	15.78 ± 0.27
GAT	23.40 ± 2.09	21.80 ± 0.68	24.10 ± 1.70	24.85 ± 1.45	21.86 ± 1.23	23.89 ± 1.37

	ROC_AUC Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	70.08	67.88	58.50	70.76	69.56	69.93
LOG	73.24	72.97	61.83	70.04	73.02	73.23
RF	83.36	82.23	67.24	80.98	83.72	83.00
SVM	64.30	57.23	51.90	56.96	65.22	63.97
XGB	84.53	83.69	68.13	81.56	84.38	84.64
GSAGE	62.04 ± 5.17	61.40 ± 5.43	65.66 ± 5.63	54.25 ± 1.37	66.32 ± 3.73	63.90 ± 1.51
GCN	82.78 ± 0.51	81.98 ± 0.30	83.51 ± 0.10	83.11 ± 0.16	82.95 ± 0.23	82.68 ± 0.24
GAT	86.09 ± 0.44	85.43 ± 0.20	77.72 ± 5.12	82.30 ± 3.21	86.17 ± 0.24	85.41 ± 0.84

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	70.08	69.07	70.39	69.98	69.56	69.89
LOG	73.24	73.56	73.16	73.22	73.33	73.08
RF	83.36	83.31	83.54	83.51	83.52	83.30
SVM	64.30	60.16	63.43	63.49	59.00	66.95
XGB	84.53	84.56	84.68	84.46	84.59	84.14
GSAGE	62.04 ± 5.17	60.70 ± 3.34	59.49 ± 2.58	63.14 ± 4.16	60.67 ± 3.99	62.22 ± 5.92
GCN	82.78 ± 0.51	82.55 ± 0.34	82.67 ± 0.36	82.67 ± 0.52	82.58 ± 0.19	81.97 ± 0.07
GAT	86.09 ± 0.44	85.95 ± 0.26	86.11 ± 0.23	85.82 ± 0.41	86.09 ± 0.28	86.29 ± 0.36

4.3.7 HARD with 10% flipped labels

The results for 10% flipped labels are shown in table 4.9. Here the extreme class imbalance is showing its effect across all models on the AP metric. GAT is still the best performing model showing some capabilities of handeling an extreme class imbalance.

SVM, KNN, LOG, and GraphSAGE all perform at low scores for both AP and AUC, showing extremely poor overall performance.

XGB, RF, and GCN do not show as big of a performance drop on the AUC score, showing that these models are still able to find patterns mainly for the normal class. The FP noise once again shows to have a large effect across all models. Consequently, the neighbour noise has the second to largest effect. These noises both introduce false positive to the dataset, thus overwhelming the positive class with inaccuracies.

Table 4.9: Results of HARD datasets with 10% of selected nodes label flipped

	AP Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	0.84	0.84	0.58	0.78	0.84	0.84
LOG	1.35	1.34	0.70	1.34	1.28	1.29
RF	3.19	2.79	0.95	2.67	3.30	2.99
SVM	0.94	0.99	0.55	0.68	1.11	0.96
XGB	2.57	2.06	0.59	2.07	2.26	2.83
GSAGE	1.84 ± 0.52	1.85 ± 0.44	0.57 ± 0.03	1.36 ± 0.17	1.64 ± 0.20	2.01 ± 0.61
GCN	4.26 ± 0.17	3.11 ± 0.13	1.05 ± 0.15	3.43 ± 0.48	3.28 ± 0.51	2.67 ± 0.63
GAT	7.90 ± 0.73	7.94 ± 1.02	1.99 ± 0.30	7.65 ± 0.77	7.39 ± 1.10	6.88 ± 0.85

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	0.84	0.84	0.84	0.84	0.84	0.84
LOG	1.35	1.32	1.32	1.35	1.41	1.43
RF	3.19	3.12	2.98	2.76	2.67	3.52
SVM	0.94	0.97	0.93	0.94	0.98	0.92
XGB	2.57	2.46	2.80	2.70	2.87	2.86
GSAGE	1.84 ± 0.52	1.73 ± 0.32	2.05 ± 0.43	1.72 ± 0.35	1.67 ± 0.39	1.51 ± 0.46
GCN	4.26 ± 0.17	3.24 ± 0.67	4.31 ± 0.51	3.59 ± 0.68	4.06 ± 0.52	3.51 ± 0.72
GAT	7.90 ± 0.73	6.81 ± 1.13	7.16 ± 1.66	8.01 ± 0.36	8.19 ± 0.66	7.73 ± 1.20

	ROC_AUC Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	56.17	56.01	49.64	56.07	56.17	56.17
LOG	68.39	68.25	57.89	69.02	67.46	67.74
RF	77.18	76.25	64.41	74.40	77.05	76.77
SVM	62.14	63.85	49.86	57.51	65.08	62.25
XGB	78.41	75.73	51.59	76.68	78.17	79.06
GSAGE	60.67 ± 2.72	54.89 ± 0.95	52.43 ± 1.30	57.13 ± 0.70	58.82 ± 1.24	60.79 ± 2.84
GCN	79.58 ± 0.30	80.23 ± 0.13	59.73 ± 2.35	79.34 ± 0.56	79.11 ± 0.64	76.35 ± 4.50
GAT	83.89 ± 1.32	82.89 ± 1.37	73.11 ± 4.36	82.51 ± 1.65	83.52 ± 0.98	83.19 ± 1.25

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	56.17	56.17	56.17	56.17	56.17	56.17
LOG	68.39	68.02	68.30	68.39	69.07	69.37
RF	77.18	76.70	76.02	76.79	76.30	77.77
SVM	62.14	62.71	61.71	62.14	62.95	62.09
XGB	78.41	76.07	80.51	79.58	78.78	79.32
GSAGE	60.67 ± 2.72	60.26 ± 2.77	59.23 ± 0.72	58.77 ± 1.08	59.11 ± 1.99	58.62 ± 1.86
GCN	79.58 ± 0.30	78.45 ± 1.68	79.49 ± 0.59	78.77 ± 1.25	79.53 ± 0.41	78.98 ± 1.51
GAT	83.89 ± 1.32	83.96 ± 0.77	82.17 ± 3.37	83.90 ± 0.48	84.14 ± 0.67	83.32 ± 0.70

4.3.8 HARD with 25% flipped labels

The results for 25% flipped labels are shown in table 4.10. Here once again the extreme class imbalance is showing it's negative effects across all models.
SVM, KNN, LOG, and GraphSAGE all continue showing extremely poor performance on the AP metric, showing no capabilities in finding the money laundering class.
XGB, RF and GCN are performing at similar levels on both metrics, with GCN having an edge on XGB and RF.
GAT once again is the model showing the best performance across all results.
Comparing the noise effects with for the previous 10%, FN and neighbour have the largest negative impact. With FP having detrimental effects on all models pushing the performance down to the lowest possible levels, even for GAT.
All noise from the topology category seem to have little to no effect on the performance of all models.

4.3.9 Summary for HARD

Overall for all HARD datasets the extreme class imbalance shows across all models for the AP metric. All models except for GAT are completely incapable of detecting the money laundering class, with GAT showing slight capabilities. RF, XBG and GCN show some performance on the AUC score. GAT is once again the best performing model across both metric scores. The most prevalent noises are FP, and neighbour, both are adding false positives as inaccuracies. Analysing the noises in the topology category, there are no differences in performance.

Table 4.10: Results of HARD datasets with 25% of selected nodes label flipped

	AP Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	0.84	0.80	0.57	0.71	0.73	0.84
LOG	1.35	0.88	0.67	1.43	1.29	0.98
RF	2.43	1.70	0.66	2.16	2.11	2.47
SVM	0.94	0.80	0.53	0.78	0.80	0.92
XGB	2.57	1.51	0.63	3.12	2.07	2.30
GSAGE	1.84 ± 0.52	1.86 ± 0.49	0.59 ± 0.05	1.38 ± 0.10	1.54 ± 0.17	1.98 ± 0.40
GCN	4.26 ± 0.17	2.77 ± 0.38	1.08 ± 0.09	3.47 ± 0.56	3.77 ± 0.53	4.00 ± 0.20
GAT	7.90 ± 0.73	4.78 ± 0.64	0.61 ± 0.15	4.21 ± 1.16	6.37 ± 1.56	6.33 ± 1.49

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	0.84	0.84	0.84	0.84	0.83	0.78
LOG	1.35	1.43	1.15	1.35	1.33	1.43
RF	2.43	3.06	3.26	2.86	3.19	2.78
SVM	0.94	1.00	0.92	0.94	1.21	0.96
XGB	2.57	2.39	2.35	2.63	2.38	2.61
GSAGE	1.84 ± 0.52	1.96 ± 0.22	1.46 ± 0.32	1.81 ± 0.28	1.75 ± 0.14	1.59 ± 0.26
GCN	4.26 ± 0.17	3.42 ± 0.69	3.47 ± 0.51	3.57 ± 0.47	3.47 ± 0.59	4.34 ± 1.03
GAT	7.90 ± 0.73	8.11 ± 1.19	8.26 ± 0.28	7.31 ± 0.80	7.58 ± 1.39	7.49 ± 0.53

	ROC_AUC Scores					
Model	Missing Label	FN	FP	Neighbour	Bipartite	Cycle
KNN	56.17	55.22	48.73	54.86	54.18	56.17
LOG	68.39	62.83	56.78	69.82	67.57	63.76
RF	74.26	72.17	54.24	71.47	76.39	77.38
SVM	62.14	60.13	48.42	59.97	59.44	62.90
XGB	78.41	72.08	52.18	74.38	77.37	75.69
GSAGE	60.67 ± 2.72	54.56 ± 0.81	50.62 ± 1.66	58.42 ± 1.56	57.57 ± 1.42	59.61 ± 2.03
GCN	79.58 ± 0.30	78.38 ± 1.69	60.60 ± 4.26	71.82 ± 1.44	79.31 ± 0.87	79.71 ± 0.15
GAT	83.89 ± 1.32	80.35 ± 2.23	51.51 ± 5.79	76.38 ± 6.35	82.52 ± 1.75	82.82 ± 1.81

Model	Missing Label	Fan In	Fan Out	Ga-Sc	Sc-Ga	Stack
KNN	56.17	56.17	56.17	56.17	55.91	55.48
LOG	68.39	69.42	66.07	68.39	68.25	69.31
RF	74.26	76.76	76.82	76.90	75.19	74.89
SVM	62.14	62.83	62.56	62.14	65.01	62.57
XGB	78.41	77.38	77.70	78.06	78.58	77.69
GSAGE	60.67 ± 2.72	63.86 ± 1.57	59.14 ± 1.79	60.62 ± 1.70	59.80 ± 2.19	57.33 ± 1.07
GCN	79.58 ± 0.30	79.05 ± 0.81	79.79 ± 0.15	79.09 ± 0.41	78.74 ± 1.56	77.86 ± 1.45
GAT	83.89 ± 1.32	84.37 ± 0.57	84.18 ± 0.27	83.49 ± 1.08	83.37 ± 1.42	83.75 ± 0.55

5

Discussion

5.0.1 Implications of findings

In scenarios where a significant portion of the labels is missing, our results in table 4.4 have demonstrated that GNNs that are capable of incorporating and utilizing the graph structure to outperform other types of models such as XGBoost.

Our comparative analysis revealed that GNNs not only leverage node features but also the underlying graph structure. This is particularly beneficial in weakly supervised settings typical in AML applications, where obtaining complete and accurate labels can be complicated. The superior performance of GNNs in these conditions underscores their robustness and flexibility, making them a preferable choice for AML tasks with incomplete data.

It should be noted that while XGBoost has higher AUC scores in some instances in table 4.4, this does not present the full picture. When looking at AP, it becomes clear that XGBoost does not perform as well as GAT in identifying the positive labels.

Moreover, in scenarios where labels are both incorrect and missing in tables 4.5 to 4.10, GNNs, particularly the GAT model, demonstrate an even greater advantage. This is a scenario that financial institutions are likely to encounter. GAT's ability to handle both types of label imperfections more effectively than other models further supports its use in real-world AML efforts. The attention mechanisms within GAT allow it to prioritize relevant parts of the graph, enhancing its capability to make accurate predictions despite the presence of noisy and incomplete labels. This suggests that implementing a model like GAT in a practical AML context could

significantly improve the detection of suspicious activities, offering a more reliable and effective solution compared to traditional models.

Comparing the results of inaccurate labels between specific money-laundering topologies, shows that there are minimal to no difference. Having inaccurate labels in specific topologies, does not differ between the models to a large extent. The largest difference in performance is found in table 4.8 where the fan-in and scatter-gather topologies for the GAT model received an average AP score of 21.80 and 21.86, where the other topologies received an average AP score of around 23-24. Showing that in a real world scenario, correctly labeling these topologies could deem more important than for the other topologies.

Our results have shown that in the scenario where significant imbalance is in the dataset it is a crucial factor that must be addressed. All models struggled to handle the severe imbalance present for the HARD dataset in tables 4.9 and 4.10 resulting in very low AP and AUC scores. To tackle this problem, one potential approach is to use graph-based sampling methods like GraphSMOTE [60]. These methods generate synthetic samples within the graph structure, improving class balance and enhancing the representation of minority classes.

For financial institutions working with AML, this finding has important implications. High class imbalance could be present in real-world AML datasets, where the number of illicit transactions is typically much smaller than the number of legitimate ones. This imbalance can severely affect the performance of AML models, leading to a high number of false negatives.

The low evaluation scores observed in our models under these conditions highlight the need for strategies to mitigate the effects of class imbalance. Financial institutions may need to explore various methods to balance their datasets and enhance model performance in these scenarios. Addressing dataset imbalance is critical to improving the effectiveness of AML systems, ensuring that they can reliably detect money laundering activity. This, in turn, can lead to more efficient and accurate AML efforts.

5.0.2 Train and test split

In this thesis, we chose to use a time-based split for the training and test sets, with a 50/50 split to prevent data leakage and ensure accurate model evaluation. This approach helps avoid skewed results and provides a clear assessment of model performance. However, in real-world applications, allowing some overlap between the training and test sets might yield better results.

Having some overlap can be beneficial to ensure sufficient graph structure is available to identify patterns. This is particularly relevant for financial institutions. If the classification were to be performed on a smaller time window than the half-year period used in this thesis, it might be advantageous to include some overlap to maintain enough graph structure and accurately detect money laundering patterns. This approach is suggested in the paper "Realistic Synthetic Financial Transactions for Anti-Money Laundering Models" by Altman et al [23]. The authors propose a data split where the validation graph includes both training and validation transactions, but only the validation indices are used for evaluation. The test graph encompasses all transactions, but only the test indices are utilized for evaluation. The train graph contains only the training transactions and the corresponding nodes. However, with this approach, data leakage from surrounding nodes outside of the chosen indices could still occur. To determine the best strategy for a real-world scenario, one must investigate different methods and decide what works best for their specific data and goals. To determine the best data-splitting approach for a given situation, one must examine various strategies, considering the specific data and goals.

5.0.3 Node classification vs edge classification

In this thesis, we have framed the task as node classification, meaning we aim to classify accounts rather than transactions as involved in money laundering. Alternatively, one could approach it as an edge classification task, focusing on classifying transactions as part of money laundering activities.

Opting for node classification could offer several advantages in real-life applications.

Firstly, it could be more efficient in terms of time complexity. Real datasets typically contain significantly more transactions than accounts. Even if each transaction between two accounts is aggregated into a single edge, edge classification could still face greater scalability challenges compared to node classification.

Node classification leverages local neighborhoods. Aggregating features from neighboring nodes is straightforward, usually requiring fewer layers of computation and reducing the risk of over-smoothing [61]. However, this approach also has some limitations. It may focus primarily on the attributes of individual nodes and their immediate neighborhoods, potentially overlooking more complex relationships across the entire graph.

5.0.4 Ethical considerations

This study demonstrates the strength of GNNs in node classification within a weakly supervised setting and highlights the shortcomings of statistical models in the same context. It is important to note that no specific information about the systems used by particular banks or the exact data they work with has been disclosed, as this research was conducted using synthetic data with estimated features. The intent of this work is to emphasize the need for more robust models to improve the classification capabilities of AML systems and to show the potential of GNNs for AML purposes when a portion of the data is inaccurate or missing.

5.0.5 Future work

Future research could focus on novel approaches to enhance the classification capabilities of GNNs, especially when dealing with missing or inaccurate labels. This could involve building out their architecture or developing complementary methods to improve their performance in AML applications. A promising candidate for further exploration is GAT, as this study has shown the model outperforms all other evaluated models in settings with inaccurate and incomplete labels.

Beyond addressing missing and inaccurate labels, another important area for future research is handling incomplete graph structures. Investigating methods to main-

tain classification performance despite incomplete graph data could yield significant improvements in potential scenarios where a bank does not have all transaction data available.

Additionally, this study has not addressed class imbalance. The synthetic dataset used in this work, and what we anticipate in real-world AML scenarios, are likely to have unbalanced data sets where correctly classified money laundering instances are a minority. Class imbalance is known to affect the accuracy of GNNs, and there has been some but limited research on addressing this issue for GNNs [60]. To the best of our knowledge, no work in this area has been specifically tailored for AML settings. Addressing class imbalance in future studies could potentially significantly enhance the practical applicability of GNNs in AML contexts.

In this report, we have chosen to train the models on all money laundering topologies simultaneously. In a real-world scenario, it might be more common to have dedicated models for specific identified money laundering patterns. Therefore, it would be interesting in future work to explore whether having models dedicated to individual topologies could improve the results. It is possible that a model trained exclusively on one pattern could handle missing and inaccurate labels even better. However, this approach could make the problem of dataset imbalance more noticeable, which would need to be addressed in such a scenario.

6

Conclusion

This thesis investigated how the presence of incomplete and inaccurate labels affects the performance of statistical models and GNNs in detecting simulated money laundering activities. By generating synthetic transaction data, we evaluated the performance of GNNs and traditional statistical models to understand how these models handle data imperfections.

Our findings demonstrate that GNNs, particularly GAT, are highly effective in handling scenarios with missing and inaccurate labels. GAT consistently outperformed traditional statistical models, achieving the highest AP and AUC scores across various datasets and label inaccuracy scenarios. This robustness indicates that GAT can effectively leverage graph structures and node features, making it a promising model for detecting suspicious activities in complex and imperfect data environments.

The study also revealed that models like XGB and RF while competitive, showed greater sensitivity to label inaccuracies, resulting in more significant performance declines. GraphSAGE and GCN, although competitive, experienced more pronounced drops in performance compared to GAT with higher levels of label inaccuracies.

In summary, the research highlights the superior performance and robustness of GNNs, particularly GAT, in scenarios with missing and inaccurate labels. These insights can guide the development of more resilient anti-money laundering systems.

7

Contributions

This thesis has been conducted by two students, Jesper Bergquist from Chalmers technical university and David Hovstadius from Uppsala university.
Both students have contributed to the literature review of the introduction and background. Jesper's work has focused on implementation of the three graph neural networks. David's work has focused on the generation and analysis of the data, and implementation of the five statistical machine learning models.
For the sections, Jesper has completed the majority of the introduction, background, discussion and conclusion. David has completed the majority of the methods and results.

Bibliography

[1] United Nations Office on Drugs and Crime, *Unodc estimates that criminals may have laundered usd 1.6 trillion in 2009*, Accessed: 2024-05-28, 2011. [Online]. Available: https://www.unodc.org/unodc/en/press/releases/2011/October/unodc-estimates-that-criminals-may-have-laundered-usdollar-1.6-trillion-in-2009.html.

[2] European Commission, Directorate-General for Migration and Home Affairs, *Money laundering*, https://home-affairs.ec.europa.eu/policies/internal-security/organised-crime-and-human-trafficking/money-laundering_en, Accessed: 2023-05-09, 2023.

[3] United Nations Office on Drugs and Crime, *Overview of money-laundering*, https://www.unodc.org/unodc/en/money-laundering/overview.html, Accessed: 2023-05-09, 2023.

[4] Europol, "Eu serious and organised crime threat assessment 2021", 2021, Accessed: 2024-05-28. [Online]. Available: https://www.europol.europa.eu/cms/sites/default/files/documents/socta2021_1.pdf.

[5] Europol, *Serious and organised crime in the eu: A corrupting influence*, Accessed: 2024-05-26, 2024. [Online]. Available: https://www.europol.europa.eu/media-press/newsroom/news/serious-and-organised-crime-in-eu-corrupting-influence.

[6] European Commission, *Money laundering*, Accessed: 2024-05-26, 2024. [Online]. Available: https://home-affairs.ec.europa.eu/policies/internal-security/organised-crime-and-human-trafficking/money-laundering_en.

[7] W. Maxwell, A. Bertrand, and X. Vamparys, "Are AI-based Anti-Money Laundering (AML) Systems Compatible with European Fundamental Rights?", in *ICML 2020 Law and Machine Learning Workshop*, Conférence en ligne, Vienne, Austria, Jul. 2020. [Online]. Available: `https://hal.science/hal-02884824`.

[8] Z. Dobrowolski and Ł. Sułkowski, "Implementing a sustainable model for anti-money laundering in the united nations development goals", *Sustainability*, vol. 12, no. 1, p. 244, 2019.

[9] P. Gerbrands, B. Unger, M. Getzner, and J. Ferwerda, "The effect of anti-money laundering policies: An empirical network analysis", *EPJ Data Science*, vol. 11, no. 1, p. 15, 2022.

[10] J. Simser, "Money laundering: Emerging threats and trends", *Journal of Money Laundering Control*, vol. 16, no. 1, pp. 41–54, 2012.

[11] E. U. A. for Criminal Justice Cooperation (Eurojust), "Eurojust report on money laundering", Oct. 2022, Accessed: 2024-05-28. [Online]. Available: `https://www.eurojust.europa.eu/publication/eurojust-report-money-laundering`.

[12] Z. Chen, L. D. Van Khoa, E. N. Teoh, A. Nazir, E. K. Karuppiah, and K. S. Lam, "Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: A review", *Knowledge and Information Systems*, vol. 57, pp. 245–285, 2018.

[13] G. S. Mann and A. McCallum, "Generalized expectation criteria for semi-supervised learning with weakly labeled data.", *Journal of machine learning research*, vol. 11, no. 2, 2010.

[14] F. Johannessen and M. Jullum, "Finding money launderers using heterogeneous graph neural networks", *arXiv preprint arXiv:2307.13499*, 2023.

[15] R. Karim, F. Hermsen, S. A. Chala, P. De Perthuis, and A. Mandal, "Scalable semi-supervised graph learning techniques for anti money laundering", *IEEE Access*, 2024.

[16] S. Marasi and S. Ferretti, "Anti-money laundering in cryptocurrencies through graph neural networks: A comparative study", in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, IEEE, 2024, pp. 272–277.

[17] Z.-H. Zhou, "A brief introduction to weakly supervised learning", *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.

[18] F. Schneider and U. Windischbauer, "Money laundering: Some facts", *European Journal of Law and Economics*, vol. 26, pp. 387–404, 2008.

[19] S. D. Jayasekara, "How effective are the current global standards in combating money laundering and terrorist financing?", *Journal of Money Laundering Control*, vol. 24, no. 2, pp. 257–267, 2021.

[20] J. Lorenz, M. I. Silva, D. Aparício, J. T. Ascensão, and P. Bizarro, *Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity*, 2021. arXiv: `2005.14635 [cs.LG]`.

[21] J. Gerlings and I. Constantiou, "Machine learning in transaction monitoring: The prospect of xai", *arXiv preprint arXiv:2210.07648*, 2022.

[22] D. V. Kute, B. Pradhan, N. Shukla, and A. Alamri, "Deep learning and explainable artificial intelligence techniques applied for detecting money laundering–a critical review", *IEEE access*, vol. 9, pp. 82 300–82 317, 2021.

[23] E. Altman, J. Blanuša, L. von Niederhäusern, B. Egressy, A. Anghel, and K. Atasu, *Realistic synthetic financial transactions for anti-money laundering models*, 2024. arXiv: `2306.16424 [cs.AI]`.

[24] M. Weber, G. Domeniconi, J. Chen, *et al.*, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics", *arXiv preprint arXiv:1908.02591*, 2019.

[25] Elliptic, *Elliptic data set*, `https://www.kaggle.com/datasets/ellipticco/elliptic-data-set`, Accessed: 2024-05-29, 2019.

[26] T. Suzumura and H. Kanezashi, *Anti-Money Laundering Datasets: InPlusLab anti-money laundering datadatasets*, http://github.com/IBM/AMLSim/, 2021.

[27] AI Sweden, *Federated learning in banking*, https://www.ai.se/en/project/federated-learning-banking, Accessed: 2024-05-07, 2023.

[28] J. He, J. Tian, Y. Wu, *et al.*, "An efficient solution to detect common topologies in money launderings based on coupling and connection", *IEEE Intelligent Systems*, vol. 36, no. 1, pp. 64–74, 2021. DOI: 10.1109/MIS.2021.3057590.

[29] C. Cortes and V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[30] N. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression", in *The American Statistician*, Taylor & Francis, vol. 46, 1992, pp. 175–185.

[31] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. John Wiley & Sons, 2013, vol. 398.

[32] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[33] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", pp. 785–794, 2016.

[34] S. Chelgani, H. Nasiri, and M. Alidokht, "Interpretable modeling of metallurgical responses for an industrial coal column flotation circuit by xgboost and shap-a "conscious-lab" development", *International Journal of Mining Science and Technology*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095268621001154.

[35] M. Amjad, I. Ahmad, M. Ahmad, and P. Wróblewski, "Prediction of pile bearing capacity using xgboost algorithm: Modeling and performance evaluation", *Applied Sciences*, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/4/2126/pdf.

[36] J. Henriques, F. Caldeira, T. Cruz, and P. Simões, "Combining k-means and xgboost models for anomaly detection using log datasets", *Electronics*, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/7/1164/pdf.

[37] E. Altman, J. Blanuša, L. Von Niederhäusern, B. Egressy, A. Anghel, and K. Atasu, "Realistic synthetic financial transactions for anti-money laundering models", *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[38] P. Zhang and G. Chartrand, "Introduction to graph theory", *Tata McGraw-Hill*, vol. 2, pp. 2–1, 2006.

[39] B. Bollobás, *Modern Graph Theory* (Graduate Texts in Mathematics 184), 1st ed. Springer-Verlag New York, 1998, ISBN: 978-0-387-98488-9,978-1-4612-0619-4.

[40] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications", *AI open*, vol. 1, pp. 57–81, 2020.

[41] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?", *arXiv preprint arXiv:1810.00826*, 2018.

[42] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2017. arXiv: 1609.02907 [cs.LG].

[43] V. Authors, "Deep learning with graph convolutional networks", *International Journal of Intelligent Systems*, 2023. [Online]. Available: https://downloads.hindawi.com/journals/ijis/2023/8342104.pdf.

[44] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks and their current applications in bioinformatics", *Frontiers in Genetics*, vol. 12, p. 690 049, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fgene.2021.690049/pdf?isPublishedV2=False.

[45] F. Valeria and J. Smith, "Graph neural networks for affective social media", in *Proceedings of the 2023 CEUR Workshop*, 2023. [Online]. Available: https://ceur-ws.org/Vol-3318/paper2.pdf.

[46] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs", *Advances in neural information processing systems*, vol. 30, 2017.

[47] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2018. arXiv: 1710.10903 [stat.ML].

[48] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks", *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[49] Financial Action Task Force, *Risk-based approach to aml: Effective supervision and enforcement*, 2021. [Online]. Available: https://www.fatf-gafi.org/content/dam/fatf-gafi/guidance/RBA-Effective-supervision-and-enforcement.pdf.

[50] McKinsey & Company, "Aml and financial crimes investigation", 2023. [Online]. Available: https://www.mckinsey.com/business-functions/risk-and-resilience/our-insights/aml-and-financial-crimes-investigation.

[51] J. Smith and J. Doe, "Detecting money laundering transactions with machine learning", *Emerald Insight*, 2022, Accessed: 2024-07-03. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/JMLC-07-2021-0075/full/html.

[52] Swish, *Swish yearly report 2023*, Accessed: 2024-06-02, 2023. [Online]. Available: https://assets.ctfassets.net/zrqoyh8r449h/2a3UFzEBCt1wRmHXRakaPZ/8ae3f6dc0638351f986de346dd820a4f/Swish_yearly_report_2023.pdf.

[53] A. T. Stephen and O. Toubia, "Explaining the power-law degree distribution in a social commerce network", *Social Networks*, vol. 31, no. 4, pp. 262–270, 2009.

[54] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, *Scikit-learn: Machine learning in Python*, https://scikit-learn.org/stable/, Accessed: 2024-05-30, 2011. [Online]. Available: https://scikit-learn.org/stable/.

[55] J. Brownlee, *Hyperparameter optimization with random search and grid search*, https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/, Accessed: 2024-05-30, 2020. [Online]. Available: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/.

[56] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric*, 2019. arXiv: 1903.02428 [cs.LG].

[57] *Pytorch*, https://pytorch.org/, Accessed: 2024-05-24.

[58] O. Contributors, *Optuna.samplers.tpesampler*, Accessed: 2024-05-26, 2024. [Online]. Available: https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html.

[59] Y. M. Lim, Y. H. Tay, H. T. Chua, and K. H. Goh, "Detecting community structure by using a constrained label propagation algorithm", *PLOS ONE*, vol. 11, no. 5, e0155320, 2016. DOI: 10.1371/journal.pone.0155320. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155320.

[60] T. Zhao, X. Zhang, and S. 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*, ser. WSDM '21, ACM, Mar. 2021. DOI: 10.1145/3437963.3441720. [Online]. Available: http://dx.doi.org/10.1145/3437963.3441720.

[61] X. Liu, J. Chen, and Q. Wen, "A survey on graph classification and link prediction based on gnn", *arXiv preprint arXiv:2307.00865*, 2023.

DEPARTMENT OF SPACE, EARTH AND ENVIRONMENT

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden

CHALMERS
UNIVERSITY OF TECHNOLOGY