A GRAPH-BASED DEEP LEARNING MODEL FOR ANTI-MONEY LAUNDERING

NAZANIN BAKHSHINEJAD

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL, 2023

Abstract

Anti-money laundering (AML) refers to a set of laws, regulations, and procedures that financial institutions and other regulated entities are required to implement to identify and prevent the use of their services for illicit financial activities. Current AML solutions rely on rule-based algorithms, which are not scalable and ineffective against new, evolving or complex money laundering patterns. On the other hand, the rapid advancement of technology and new sophisticated financial instruments have increased the complexity of money laundering methods. Machine learning has the capability to learn and identify new or complex money laundering patterns. Within this context, the thesis offers two major contributions. First, we conducted a survey that provides a comprehensive review of existing machine learning-based AML solutions from a data-oriented perspective. We studied existing machine learning models proposed for AML in terms of datasets used, input and output data, approaches to the class imbalance problem, and classification metrics. To the best of our knowledge, this survey is the first that focuses on different aspects of data, classification metrics and related issues (e.g., the class imbalance problem). Second, we propose an AML detection system and a graph-based machine learning model to identify suspicious transactions. The detection system first transforms a dataset of accounts and transactions into a graph structure and applies the node2vec (N2V) algorithm to convert the graphs into feature vectors. The feature vectors are then input into a graph convolution network (GCN), which will then classify the transactions as normal or suspicious. (Each suspicious transaction, which is known as an alarm, will be investigated manually by a financial analyst to confirm if it is a normal transaction

or a money laundering transaction.) To overcome the inherent class imbalance of AML data (i.e., the number of money laundering transactions in a dataset is much smaller than the number of normal transactions), we use a combination of techniques, including over-sampling and classifier threshold moving. Our experimental results show that the proposed N2V-GCN system can achieve very low false negative rates (money laundering transactions misclassified as normal transactions), reaching zero in one experiment. At the same time, the proposed system lowers the false alarm rates (normal transactions classified as suspicious transactions) to under 50%, much lower than the current industry standard of 90% or more.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. U.T Nguyen, whose unwavering support and encouragement helped me to navigate the complexities of this research project. Her insightful guidance and constructive criticism helped me to develop my ideas, refine my arguments, and hone my writing skills. I am fortunate to have had such a knowledgeable and supportive supervisor, and I will always be grateful for the lessons I learned from her during the course of this thesis.

I also would like to thank Prof. Aijun An, and Prof. Regina Lee for dedicating their time to review and provide valuable feedback on my thesis, which helped to enhance the quality of my work.

I would also like to extend a special thank you to my dear friend and team mate, Reza Soltani, for his constant support during my thesis. Reza was a source of inspiration and motivation, and his friendship helped to make the long hours of research and writing more enjoyable. I would like to thank all my team mates, who always supported me mentally and made the teamwork more pleasurable.

I am most grateful to my dear family for supporting me in all stages of my life and making me progress always even from a long distance.

Table of Contents

List of Tables

List of Figures

Abbreviations and Acronyms

AML Anti Money Laundering.

BFS Breadth First Search.

DFS Depth First Search.

FAR False Alarm Rate.

FATF Financial Action Task Force.

FinCen Financial Crimes Enforcement Network.

FINTRAC Financial Transactions and Reports Analysis Centre of Canada.

FNR False Negative Rate.

FPR False Positive Rate.

GCN Graph Convolution Network.

GNN Graph Neural Network.

KNN K Nearest Neighbour.

ML Money Laundering.

SAR Suspicious Activity Report.

SMOTE Synthetic Minority Over-sampling technique.

SVM Support Vector Machine.

TPR True Positive Rate.

Chapter 1

Introduction

This chapter includes an introduction to our research on anti-money laundering (AML), where we provide background information about AML and the reasons behind our research investigations. We explain the problem and its associated challenges and highlight what we contribute to the subject. Furthermore, in this chapter, we aim to provide an overview of our research objectives and demonstrate the importance of addressing money laundering.

1.1 Background Information

Money laundering involves the conversion of illicit funds, known as "dirty" money, into seemingly legitimate assets. This process typically involves the proceeds of criminal activities like tax evasion, human trafficking, illegal gambling, terrorism, and theft. It is worth noting that money laundering is the world's third-largest industry, accounting for roughly 2% to 5% of global Gross domestic product (GDP), which equates to a staggering $800 billion to $2 trillion in current US dollars, according to sources such as the united nations office on drugs and crime (UNODC) [1, 2].

Money laundering is a complex process that involves three distinct stages [3, 4] as illustrated in Figure 1.1. These stages are:

1. **Placement:** This is the initial phase where the dirty fund is first introduced into the legitimate financial system. The purpose of this stage is twofold: to

relieve the criminal of having to hold onto a large amount of cash and to make it appear as though the money has come from a legal source. However, there is a risk of arousing suspicion from officials during this phase due to the large amounts of money being moved. To avoid this, money launderers use various methods of cash placement, including gambling, loan repayments, currency exchanges, and blending funds.

2. **Layering:** This is the most intricate stage of money laundering. The goal of layering is to make it difficult to trace the origin of the dirty money by creating multiple layers of transactions that obscure the audit trail. Money launderers may use tactics such as electronic transfers between different countries, investing in various financial institutions or foreign markets, and exploiting legal loopholes to avoid detection.

3. **Integration:** This is the final stage, where the laundered money is reintroduced into the economy through legitimate sources without attracting any suspicions. The aim is to return the money to the criminal in a way that makes it appear as if it came from a legal source, such as through a purchase of high-value items like jewelry or high-end vehicles.

To combat money laundering, financial institutions such as banks are required by law and industry regulations to implement various measures for identifying and preventing money laundering activities [5]. For example, financial institutes must closely monitor their customers' accounts and promptly report any suspicious activity to the appropriate authorities using a suspicious activity report (SAR) form[6].

The term anti-money laundering refers to the set of laws, regulations, and procedures designed to prevent criminals from disguising illegally obtained funds as lawful income [7, 8]. These AML regulations begin at the initial point of interaction between the financial institution and its customers.

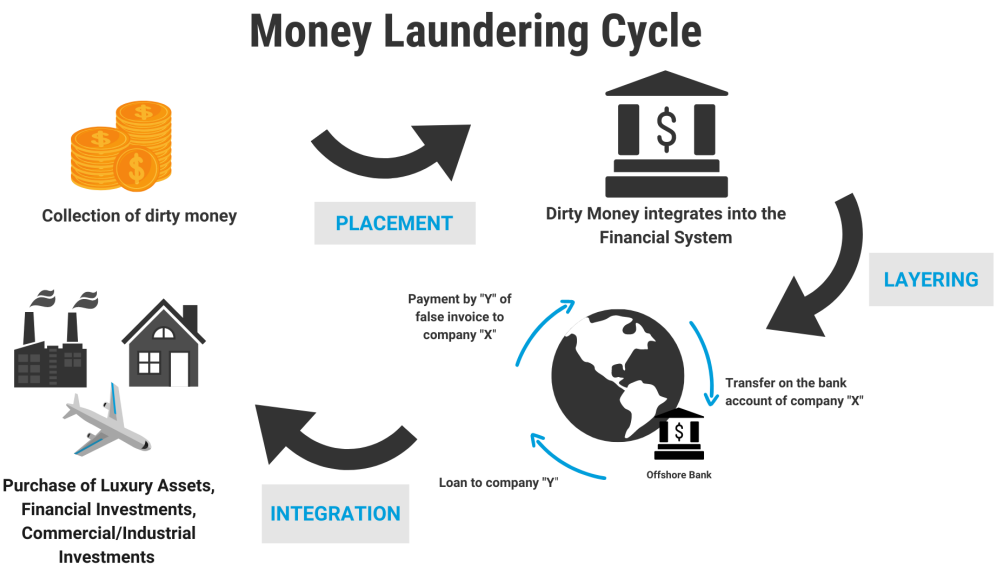

Figure 1.1: Different stages of money laundering

For instance, AML regulations require financial institutions to establish comprehensive customer due diligence (CDD) plans to evaluate the potential risks associated with a customer [9]. The CDD involves a thorough review of a customer's information, including their identity, business interactions, and financial transactions. This information is used to determine the customer's risk score and to assess whether they pose a potential risk to the financial institution. The goal of CDD is to ensure that financial institutions are not facilitating illegal activities by doing business with harmful clients [10].

Government organizations are responsible for establishing AML compliance regulations. For instance, the Financial Transactions and Reports Analysis Centre of Canada (FINTRAC) is a Canadian government agency that sets the rules and policies aimed at detecting, preventing, and deterring money laundering and financing of terrorism activities. These regulations must be followed by all Canadian financial institutions that deal with money and securities, such as accountants, real estate agencies, and casinos. In the United States, Financial Crimes Enforcement Network (FinCEN) provides guidance and AML regulations to financial institutions. Lastly,

Financial Action Task Force (FATF) is one of the many international organizations that establish standards for the effective implementation of AML measures to prevent money laundering [8].

Currently, most banks rely on conventional rule-based AML systems [11]. These systems rely on a simple set of conditions and predefined thresholds established by domain experts [12, 13]. The thresholds are based on a set monetary value for each transaction and are used to detect specific money laundering patterns and user scenarios that exceed those values. Despite their ease of implementation, these models have four significant drawbacks [14]. First, they lack the ability to learn and generalize. Secondly, they are unable to improve the rules to support the changes in the technology landscape. Third, they cannot identify complex money laundering topologies and patterns. Finally, they fail to adapt to the growing number of transactions.

Recently, machine learning algorithms have been introduced as a solution to the problems associated with traditional, rule-based models in the AML research [15, 16, 17, 18, 19, 20, 21, 22, 23, 24]. These algorithms have proven to be a valuable solution to AML experts, enabling them to learn and adapt to new ML patterns. This is achieved through their ability to learn from the previous execution of the AML system and the ability to use new data, capturing complex patterns through information obtained in the training step. This makes machine learning algorithms a solid alternative to traditional methods of detecting money laundering activities, particularly in managing and analyzing large amounts of unstructured information, recognizing complex patterns, and self-updating mechanisms using anonymous data [25].

1.2 Motivations and Problem Definition

Despite the fact that machine learning algorithms can aid in identifying potentially fraudulent transactions, there are two main significant challenges while applying machine learning in the AML field [26].

The first challenge is the scarcity of publicly available AML datasets. Machine learning algorithms need a significant amount of data for training purposes, but the confidential nature of financial transactions means limited access to realistic financial and money laundering datasets. Banks are hesitant to share their customer data with external entities to avoid breaching customer confidentiality and privacy policies. As a result, most research works in AML space use either genuine financial datasets that have been anonymized or augment genuine datasets with artificially generated money laundering records.

The second major challenge in AML datasets is the problem of class imbalance. AML data is inherently imbalanced, meaning the number of money laundering transactions (i.e. positive samples) is significantly lower than the number of clean transactions (negative samples). The issue of class imbalance in AML datasets can result in several problems, mostly impacting the training phase of an AML model [27, 28].

These problems include training an AML model that is biased towards the negative class, leading to the model being unable to identify positive samples and misleading high accuracy of detection, where the model can detect almost all regular transactions while money laundering cases go undetected. Additionally, there can be a high rate of false negatives, where money laundering transactions are incorrectly classified as regular transactions, putting financial institutions at risk. There can also be a high rate of false positives where clean transactions are flagged as ML records, which can prolong the time experts spend investigating the flagged transactions.

Therefore, it is essential to address the inherent class imbalance of AML data during the training phase. However, most existing AML solutions only focus on the learning algorithm aspect of machine learning and do not consider the class imbalance issue. In this thesis, we conducted a survey of existing machine learning-based AML solutions, focusing on the data-related aspects of their model. Also, we introduced a new data-oriented taxonomy for AML works. Although there are many AML surveys in the literature, these surveys tend to neglect data-related aspects of machine learning,

which is just as important as the learning algorithm itself [26, 29, 30, 22]. This thesis aims to fill this gap by conducting a survey on machine learning-based AML models from the data perspective and provide a new taxonomy for existing AML solutions. The main objective of our survey is to answer the following questions:

- The types of data used as inputs to the AML model and the data types returned by the models. The input data for money laundering detection models include transactions, account information, or a combination of both, as well as other types of data, such as relations among accounts. The purpose of an AML model is to identify suspicious transactions or customers.

- The methods by which existing works address the problem of imbalanced data: There are various techniques available to handle class imbalance. Still, their effectiveness depends on the specific use case at hand, and there are no guidelines for determining the suitability of a given technique.

- The evaluation metrics used to assess the performance of AML models: It is crucial to use appropriate evaluation metrics when dealing with imbalanced datasets because common metrics such as accuracy may not accurately reflect the performance of the model.

In addition to conducting a survey of existing AML models, we have developed a new AML model that addresses the issue of imbalanced data. Our model is named N2V-GCN. Our model uses node2vec, which is a graph embedding technique and a graph convolution network to classify transactions as either clean or money laundering. To handle the issues of imbalanced data, we employed two approaches: sampling at the data level to adjust the distribution of classes and modifying the threshold of the classifier to fine-tune the classification algorithm. The steps involved in our N2V-GCN model are as follows:

- Adjusting the ratio of the original dataset by applying a sampling technique called synthetic minority over-sampling (SMOTE) to create a new dataset with a balanced ratio of money laundering transactions to clean transactions.

- Converting the dataset into a graph structure, where customer accounts are represented as nodes and the transactions between them as edges.

- Applying node2vec to generate embedding vectors for the transactions, taking into account the relationships between customer accounts. These embedding vectors serve as feature vectors.

- Determining the optimal threshold for our graph convolution network classifier and adjusting the default threshold to the optimal value to classify transactions accordingly.

1.3 Contributions

The main contributions of the thesis are as follows:

- We conducted a survey which provides a comprehensive overview of existing AML solutions from a new data-oriented perspective. While previous surveys focused on the algorithms used and their performance, our survey explicitly considers the crucial role of data in AML. Our main goal is to examine existing AML machine learning models in terms of their datasets, input and output data, approaches to dealing with the class imbalance problem, and evaluation metrics. To the best of our knowledge, this is the first work that investigates these aspects together.

- We designed a graph-based model named N2V-GCN that utilizes two techniques, namely node2vec and graph convolution networks. The node2vec algorithm was initially applied to generate highly informative embedding vectors that represent transactions. These embedding vectors are considered as feature

vectors that capture the essence of each transaction. Next, we employed a graph convolution network to classify transactions using the embedding vectors generated from node2vec. The graph convolution network is a type of neural network that operates on graphs and is capable of capturing the relational information of graph-structured data. By combining the strengths of these two approaches, we achieved highly reduced false alarm rates and minimized false negative rates to zero during transaction classification.

- We proposed a hybrid technique for handling the inherent class imbalance in AML data comprising two stages: data-level and algorithm-level. Firstly, at the data-level, we utilized Synthetic Minority Over-sampling Technique (SMOTE) to address the issue of the minority class (i.e., money laundering transactions) being underrepresented. This involved generating various datasets by adjusting the distribution of the minority and majority classes to evaluate the impact of class distribution on AML model performance. Secondly, at the algorithm-level, we fine-tuned the classifier's threshold to increase its ability to detect positive cases, thus reducing the possibility of missing any money laundering transactions. In other words, adjusting the threshold made the model more sensitive toward the minority class.

1.4 Thesis Organization

The remaining sections of the thesis are structured as follows. Chapter 2 provides an overview of current AML solutions, covering both rule-based and machine learning approaches, as well as their challenges. Chapter 3 comprises a detailed survey of existing AML solutions from data requirement and specification perspectives. In Chapter 4, we introduce our model for identifying money laundering transactions. Chapter 5 presents the results obtained from a series of experiments conducted to evaluate the efficacy of our proposed model. Finally, in Chapter 6, we finalize the

thesis by providing a summary of our findings and outlining potential areas for future research.

Chapter 2

Literature Review

Detecting and preventing money laundering activities is of utmost importance for the economy, businesses, banks, and financial institutions. Criminals often target banks due to a large amount of data and transactions, making it difficult to trace the source of their illicit funds. Unfortunately, the number of money laundering cases is rising, emphasizing the urgency of developing effective solutions to combat this issue.

This chapter aims to provide an overview of existing money laundering detection techniques, categorizing them into rule-based (or traditional), machine learning, and deep learning models. Additionally, we will introduce various machine learning and deep learning methods that do not fit into one specific approach but combine several methodologies. Furthermore, this chapter will explore the theoretical background of the deep learning algorithm used in our proposed AML framework.

2.1 Traditional Rule-Based Models

Money laundering detection has existed since the 1970s when financial institutions began reporting large transactions to their respective governments. In the late 1990s, statistical techniques such as Bayesian models and temporal sequence matching were employed to identify money laundering activities [31].

Over the past few decades, money laundering detection models have primarily been based on embedded rules - a set of if-then conditions that are verified based

on the features of a transaction or account. These rules are typically created by consultants and domain experts using historical data or business intuition [13]. One of the earliest anti-money laundering (AML) systems [32] was created in 1995 and was rule-based. The Wolfsberg AML principles were introduced in 2000 as a new initiative to combat money laundering [33]. These principles provide a set of rules and guidelines that regulate establishing and maintaining relationships between private bankers and clients.

Figure 2.1 depicts the process of traditional rule-based models. The overall procedure of these approaches involves three primary steps:

1. A transaction monitoring system consistently monitors money transfers between customer accounts and checks all transactions against the rules to determine whether anything unusual occurs.

2. If the system identifies a suspicious transaction that violates one or more rules of the model, it generates an alert for that specific transaction.

3. After creating a list of transactions with alerts, the AML specialist group reviews the alerts and investigates the details of related transactions to decide which alerts are generated correctly and whether the associated transaction is fraudulent.

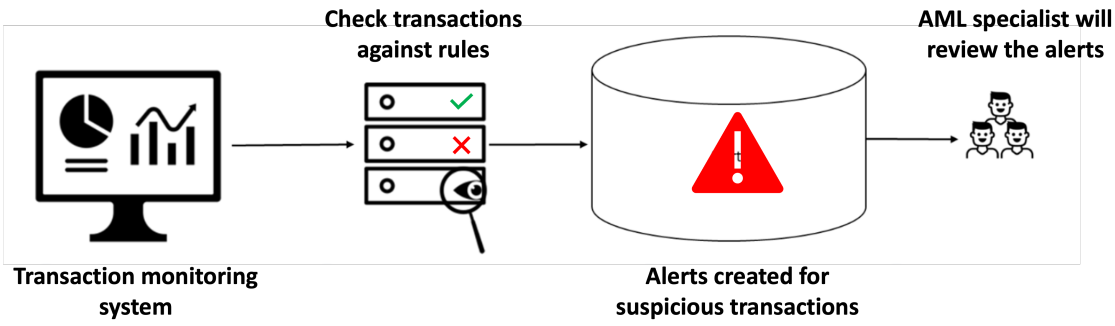

Figure 2.1: Traditional rule-based AML systems

Compared to other options, rule-based models for AML have two significant advantages. Firstly, rules provide an easy way for analysts to encode their domain knowledge into the solution by creating various rule scenarios that capture various recognized money laundering activities. Secondly, rules offer an interpretable solution, which is crucial to demonstrate compliance with the bank's jurisdictional regulations. However, despite these benefits, there are four major deficiencies of rule-based models:

1. **Lack of learning or generalization capabilities.** They can only identify patterns they already know, and if they encounter new, unseen money laundering schemes, they will fail to flag suspicious transactions. This is due to the static nature of the rules, which are limited to known patterns and overlook a significant portion of suspicious transactions due to the changing nature of money laundering.

2. **Inability to keep up with changes in technology.** They have difficulty adapting rules to the changing technological environment. As banks use more sophisticated technologies to detect money laundering, criminals develop more complex methods, requiring financial institutions to adapt continuously to different money laundering schemes and keep pace with technological advancements.

3. **Inadequacy in dealing with complex transaction patterns.** They often only consider simple patterns and single transactions. However, in today's technological environment, there are various banking services, each with its transactions, making it difficult to trace money transfers within these services.

4. **Limitations in handling the growing volume of transactions.** They do not have the capability to check every transaction comprehensively and consistently, and a few checks may result in undetected money laundering activities. Therefore, defining an appropriate set of rules for detecting illegal activities

among massive transactions is challenging.

In conclusion, embedding static rule-based systems into dynamic and digital banking environments does not provide adequate safeguards to combat money laundering.

2.2 Machine Learning Models

A few years after the introduction of rule-based AML models, in 2005, machine learning algorithms have been proposed as an alternative to mitigate the problems of traditional models [34]. Machine learning is a branch of artificial intelligence that enables computers to automatically learn from experience without being explicitly programmed. The fundamental objective of machine learning is to give computers the ability to self-learn, change, and develop from unseen data without human intervention [35].

Machine learning algorithms can enhance the effectiveness of existing AML models and overcome some of the limitations of traditional rule-based methods [26]. Unlike rule-based models, which rely on a pre-defined set of rules to identify suspicious transactions, machine learning algorithms can learn from historical data and identify patterns and anomalies that may not be evident in a static set of rules. This makes machine learning algorithms more adaptable and able to detect new and evolving money laundering schemes.

Furthermore, machine learning algorithms can analyze large volumes of data quickly and accurately, which is important given the increasing number of transactions and the complexity of modern financial systems. By automating the process of identifying suspicious transactions, machine learning algorithms can reduce the workload of human analysts and free up their time to focus on more complex tasks.

The development and use of machine learning algorithms increased rapidly since they decreased the labour-intensive work of suspicious alert investigation by AML analysts. Figure 2.2 indicates that the use of machine learning algorithms in the

AML field has increased in recent years.

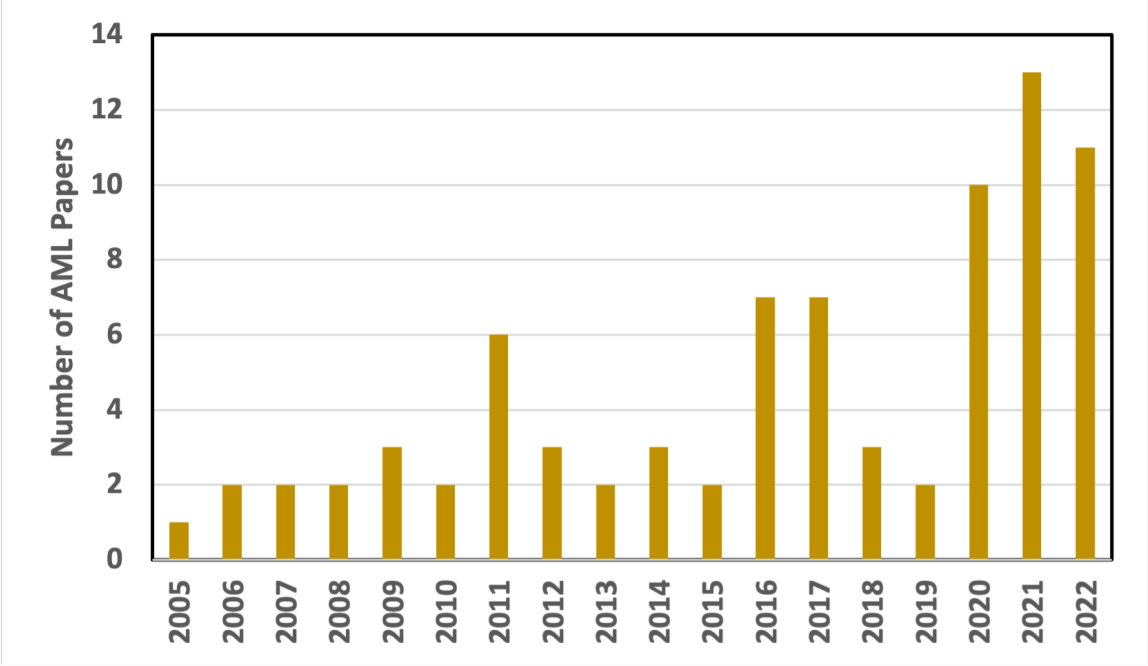

Figure 2.2: Distribution of machine learning-based AML papers

These algorithms can be integrated into various conventional money laundering detection stages. For example, the transaction monitoring system can be combined with a machine learning algorithm to classify transactions as usual or suspicious before being checked against the rules. In another scenario, machine learning can serve as a framework to determine whether a generated alert leads to an actual suspicious transaction [15]. Figure 2.3 illustrates where and how machine learning can modify traditional methods.

Machine learning models can be categorized into two main groups based on their approach: supervised and unsupervised algorithms [26]. However, with the emergence and popularity of graph-based machine learning models, they are now recognized as a distinct group. In the following, we will explore previous research on all three groups: supervised, unsupervised, and graph-based approaches.

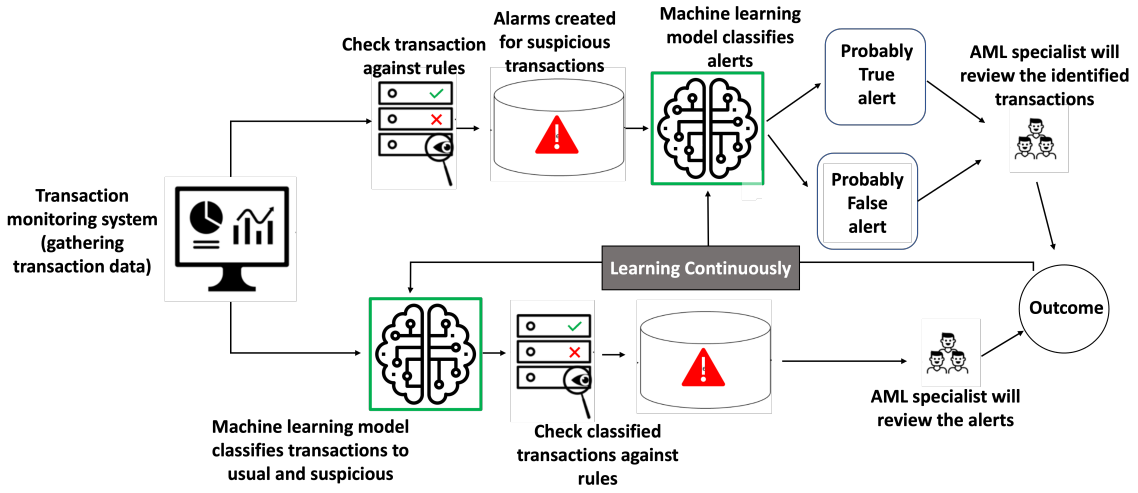

Figure 2.3: Integration of machine learning into traditional AML systems

2.2.1 Supervised Methods

Supervised machine learning models are designed to learn from a labeled training dataset containing both clean and suspicious transactions and the necessary patterns or features required to create a detection model. As the model trains, it gradually acquires knowledge from the data, enabling it to predict new unseen samples with a high level of accuracy [36]. However, the dataset must be appropriately structured to apply a machine learning algorithm in supervised mode. This technique is particularly useful for banks with experience in developing machine learning-based detection models. The following paragraphs provide a brief explanation of supervised machine learning techniques mostly used in the literature review.

Bayesian Classifiers

Naïve Bayes refers to a set of classification methods that rely on Bayes' conditional probability principles for fast analysis. It is a group of algorithms that follow the same fundamental principles rather than a single algorithm [37]. By assuming the independence of features, Naïve Bayes assigns a class to new observations that maximize the probability of the class label. The algorithm treats each class label as a random variable. Because of this, Naïve Bayes models can be created easily with-

out the need for complicated iterative parameter estimation. This crucial advantage makes the algorithm practical for analyzing large datasets.

Bayesian networks have shown promise in detecting money laundering activities as a standalone model and in combination with other machine learning algorithms. A variety of research studies [38, 39, 40, 41, 42, 43, 21] have explored the application of Bayesian networks to this important task. For example, Raza et al. [39] proposed a dynamic Bayesian network that analyzes transaction sequences to detect anomalies. Their model identifies patterns in monthly customer transactions and calculates an anomaly index using rank and entropy (AIRE). This index allows for the assignment of a label to the transaction based on the degree of anomaly detected compared to a pre-defined threshold. Overall, Bayesian networks are a powerful tool in the fight against money laundering.

Decision Trees

The decision tree is a robust and widely used supervised machine learning algorithm for classification and regression tasks. Its primary objective is to predict the target variable by creating decision rules based on the features of the dataset. The three fundamental components of a decision tree are the node, branch, and leaf [44]. Nodes correspond to tests performed on specific features, branches provide the test results, and leaves contain the final class label.

What sets the decision tree apart from other machine learning models is its ability to provide an explanation for its predictions. The rationale behind each prediction is embedded within the structure of the decision tree. However, the decision tree does have some limitations. It may not scale well and can be prone to over-fitting when applied to complex problems [45].

In the field of AML, decision trees and their variations, such as AdaBoost or bitmap-based models [46], are frequently employed among rule-based approaches that combine rules with machine learning techniques [47, 48, 49]. These models have been

used in various studies [50, 51, 52, 53, 15] to detect suspicious financial transactions.

Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are a versatile set of supervised machine learning techniques widely used in regression and classification tasks [54]. SVM calculates the maximal margin, which is the distance between two classes, using a separator super-vector of data points. One of the most significant advantages of SVM is its ability to transform a nonlinear issue space into a linear problem in a higher-dimensional space, allowing it to generalize well on high-dimensional space problems.

In the AML literature, SVM has been extensively applied [55, 56, 34, 49, 57]. For instance, Raiter et al.[19], Jun et al.[58], Villalobos et al.[59], and Tundis et al.[43] are among the studies that have used SVM as the primary or part of their detection models. To improve the performance of standard SVM, Lv et al.[60] and Lopes-Rojas et al.[61] proposed replacing the kernel function of SVM with a radial basis function (RBF).

Random Forest

Random forest is an ensemble method composed of numerous decision trees, and it is suitable for classification and regression problems. In classification, the random forest output is the class that most trees choose. In regression, the mean or average of the values obtained from individual trees is the result.

Random forests can decrease the risk of over-fitting and increase overall accuracy compared to a single decision tree. However, this success depends on uncorrelated decision trees because if a random forest uses identical or similar trees, the result will be very similar to a single decision tree. Random forests utilize bootstrapping and feature randomness [62] to achieve uncorrelated trees. Moreover, decision trees in a random forest model run in parallel, ensuring that performance cannot be affected by a single tree.

In the literature on money laundering detection, most works use the random forest in combination with other machine learning algorithms [23, 56, 49, 63, 43, 64, 57, 65, 19]. However, some works, such as Ketenci et al. [66], recommend using the random forest as a single detection algorithm. They utilized time-based features with random forests to detect suspicious transactions.

K-Nearest Neighbors (KNN)

The k-nearest neighbour (KNN) algorithm is a non-parametric supervised learning classifier. It works by identifying the k number of training samples closest in the distance to a new data point and then assigning a label or value to that data point based on the majority class of the k neighbours. The distance metric used can vary, with the most common being Euclidean distance. The choice of k is a hyperparameter that can significantly affect the algorithm's performance and can be determined using cross-validation [67]. The KNN algorithm is simple, easy to understand, and can be applied to a variety of applications such as image recognition, recommendation systems, and anomaly detection. However, its main drawback is its high computational complexity, making it less suitable for large datasets. Recent research [68] has shown that KNN effectively reduces false alarms and improves fraud detection rates.

Borrajo et al. [52] proposed a method for generating diverse and realistic new behaviours to detect and mitigate potential adversarial actions proactively. They explained that various factors drive customers' behaviours, including hidden goals and encountered states. Goals, conditions, and actions can be converted into a high-level representation, aligning with the requirements for suspicious activity reports (SARs) filed by banks. While most artificial intelligence models for AML contain this information, the representation size is typically limited. The researchers aimed to map the AML problem to a relational classification task using the KNN algorithm. The input was a trace of human behaviour and associated observable state components, and the output was whether the trace was part of a money laundering activity.

Artificial Neural Networks

An artificial neural network is a powerful machine learning technique that mimics the neural structure of the human brain. Its primary goal is to uncover hidden features that may not be readily apparent to humans. The network is composed of interconnected nodes and has input, hidden, and output layers, as illustrated in Figure 2.4 [69].

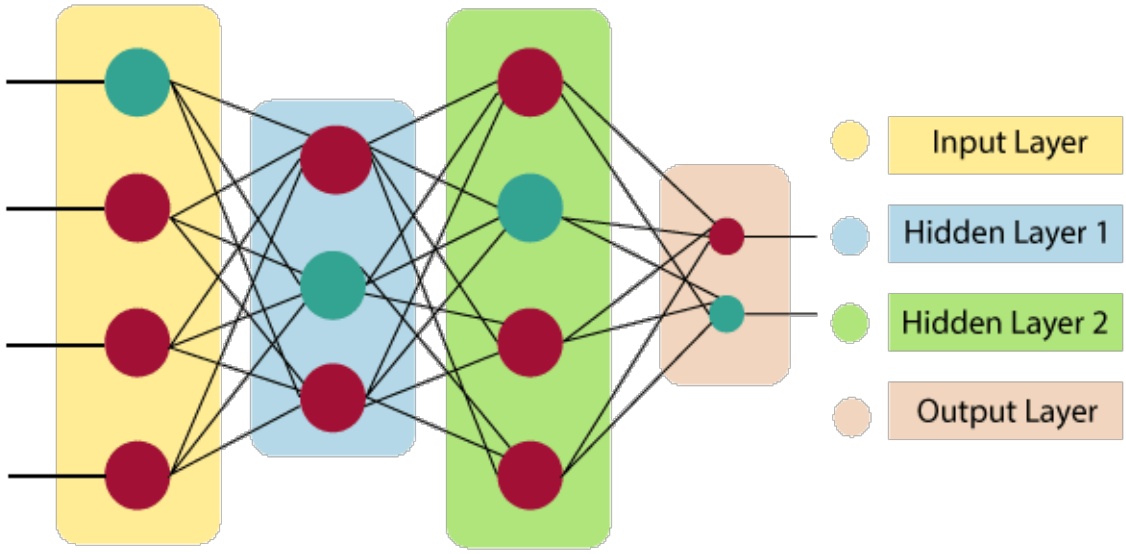

Figure 2.4: Different layers in an artificial neural network

The input layer receives and passes the data to the hidden layer, where neurons process the inputs using weighted connections and an activation function. The resulting output from the middle layer is then passed to the output layer. An artificial neural network typically has one or more hidden layers but only one input and output layer [69]. The number of neurons in the output layer equals the number of classes in a given machine learning problem. For example, in AML, a neural network may have two neurons in the output layer to identify regular and money laundering transactions.

Various neural network techniques have been used in several published works related to AML [60, 59, 70, 71]. For example, in one of the earlier works, Yang et

al. [72] utilized neural network algorithms to develop an AML system for a union bank in China to detect money laundering on online payments. Additionally, the neural network can be combined with other techniques. For instance, Zand et al. [53] designed an AML scheme that enables banks to share information while ensuring data confidentiality and integrity. Their framework utilized a neural network as the machine learning model for the detection phase.

Although the supervised models in the literature review frequently use the algorithms covered in our studies, there are alternative methods available for transaction classification, such as logistic regression [73, 21, 57, 74, 64], XGBoost [15, 75], and scan statistics [76].

2.2.2 Unsupervised Methods

Unsupervised machine learning methods attempt to group unlabeled data into clusters based on shared characteristics. Unlike supervised models, unsupervised algorithms are trained without labeled data. Instead, labels are applied to the clusters generated by the algorithm after training to assess their accuracy [77]. There are relatively fewer works focusing on unsupervised AML in comparison to supervised models. In the following, we provide a brief overview of unsupervised algorithms commonly utilized in the AML context.

K-Means Clustering

K-means is one of the most popular clustering algorithms for creating different groups of similar data. The goal of k-means is to identify groups of similar data, put them in separate clusters, and uncover hidden patterns in each cluster [78]. To this end, K-means attempts to learn from features of data and search for a pre-defined number (K) of clusters in an unlabeled dataset. Then, it defines the value K to indicate the number of centroids. It then produces K number of clusters in the dataset by assigning each data sample to a cluster based on the smallest delta between the

cluster's centroid and the dataset. From there, it designates and averages data close to a given centroid to discover the proper cluster for each data.

Among different clustering techniques, the k-means algorithm has been primarily used in the literature as the main [79, 80, 81, 82, 83] or part [84, 85, 86, 87] of the AML framework due to its simplicity.

Outlier Detection

Outlier detection techniques are used to identify abnormal behaviour, unusual patterns, rare items, and events that deviate significantly from the majority of observations in a dataset [88]. Unlike clustering techniques that group data points into different clusters, anomaly detection models focus solely on identifying data points that are different from the normal behaviour of the dataset.

Given their ability to distinguish unusual samples, outlier detection techniques can be an excellent tool in the context of AML [14, 89, 39, 90]. However, these techniques may have a high rate of false positives [88]. Currently, one-class SVM (OCSVM) and isolation forest are two of the most widely used outlier detection approaches for identifying suspicious money laundering cases [91].

One-Class SVM (OCSVM)

A traditional SVM aims to differentiate between various classes in test data based on training data. However, when we have samples from only one class (known as the normal class) and our objective is to determine which samples differ from normal ones, we utilize OCSVM. OCSVM creates a boundary around regular samples to separate them from abnormal ones. It is helpful in identifying anomalies, but if the training dataset includes too many abnormal samples, the accuracy will suffer [92].

Tang et al. [34] used OCSVM to develop one of the earlier machine learning-based works in the AML field to identify customers' criminal activities. Additionally, we noted that OCSVM had been employed for customer account risk assessment.

Shokry et al. [93] and Camino et al. [94] proposed OCSVM, which generates a list of suspicious accounts with their associated probabilities as output, by utilizing transaction and account data as input for their AML frameworks.

In addition to the algorithms described previously, several other unsupervised algorithms have been proposed in the AML domain, including expectation maximization [79, 95], CLOPE [96], minimum spanning tree [97], structural similarity-based clustering [98], distance-based clustering [99], empirical mode decomposition [100], and AutoEncoders [101].

2.2.3 Graph-Based Methods

Graph analysis has emerged as a valuable technique for investigating money laundering, as it provides a way to represent the complex network of interactions and relationships between individuals involved in these activities [102, 103, 104, 50, 105, 106]. By converting a dataset of transactions and customer accounts into a graph, where nodes represent accounts and edges represent transactions, it becomes possible to visualize and analyze money flow and identify unusual behaviour patterns.

One of the primary objectives of this approach is to identify money laundering groups, and Soltani et al. [98] proposed a graph-based model for this purpose. This model scans large volumes of financial data to identify pairs of transactions with shared attributes and behaviours that may indicate money laundering. These transactions are then combined into a graph, and clustering methods are applied to identify potential money laundering groups.

2.3 Deep Learning Models

Deep learning is a branch of machine learning that employs artificial neural networks consisting of three or more layers. The primary objective of these neural networks is to mimic the human brain's functioning and acquire knowledge from vast data sets. While a single-layered neural network can make estimations, the inclusion of multiple

hidden layers can significantly enhance the model's efficiency.

Deep learning distinguishes itself from classical machine learning in terms of data type and learning strategy [107], as shown in Figure 2.5. While machine learning algorithms make predictions using structured, labeled data organized into tables, they can also employ unstructured data, which typically goes through a pre-processing phase known as feature extraction. This process involves creating hand-crafted features for the machine learning model, which can be tedious and labour-intensive. In contrast, deep learning can handle unstructured data, eliminating the need for pre-processing and manual feature extraction by human experts.

As an example, if we want to categorize a set of transactions into legal or illegal groups, deep learning can automatically determine which features are most effective in distinguishing each transaction from another, such as the transaction type. This is unlike machine learning, where a human expert would need to establish the hierarchy of features manually. By automating the feature extraction process through gradient descent and back-propagation, deep learning eliminates the need for human expertise, making it a more efficient and effective method for handling unstructured data [108].

Graph neural networks (GNNs) have proven to be a powerful tool for analyzing graphs and detecting suspicious transactions in the Bitcoin network [109, 110, 111]. These deep learning models are a type of neural network that incorporates both convolutional neural networks (CNNs) and graph embedding, allowing them to capture the dependencies between nodes in a graph. By passing the input graph through a series of neural networks and converting it into graph embedding, GNNs can retain information on nodes, edges, and global context and capture the properties of neighbour nodes. The overall structure of GNNs is similar to other neural network models, consisting of several hidden layers connected by an activation function. Figure 2.6 provides a simple illustration of the structure of a GNN.

Weber et al. [112] developed a modified version of GNN called graph convolutional neural network (GCNN) for forensic analysis of financial data. The GCNN predicts

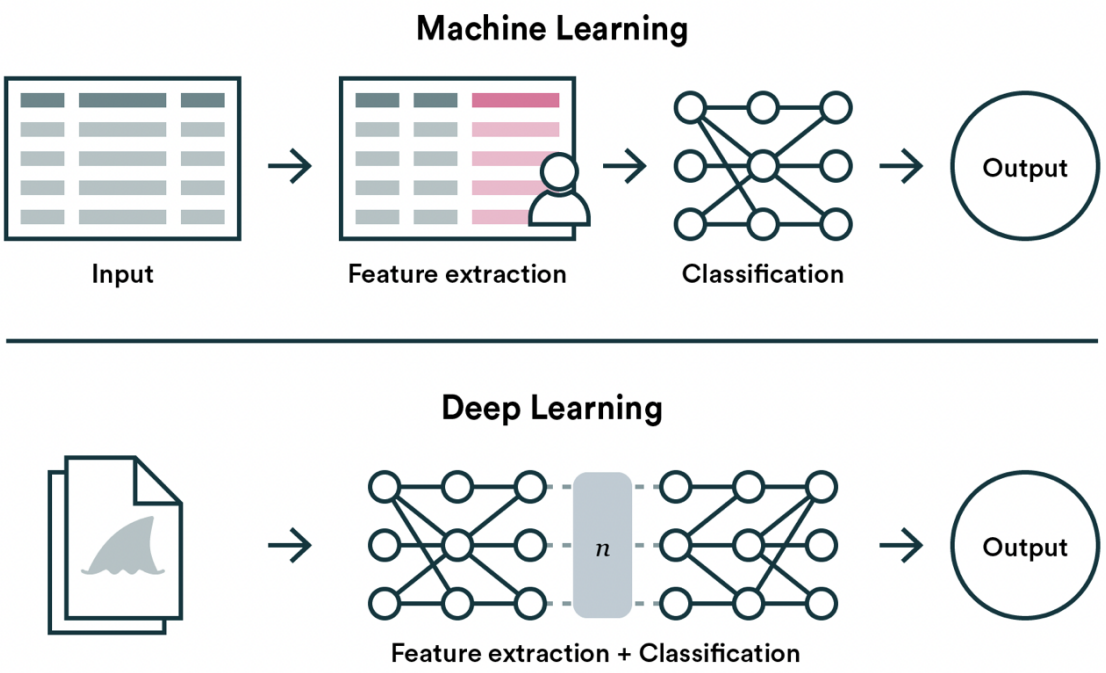

Figure 2.5: Machine learning vs. deep learning

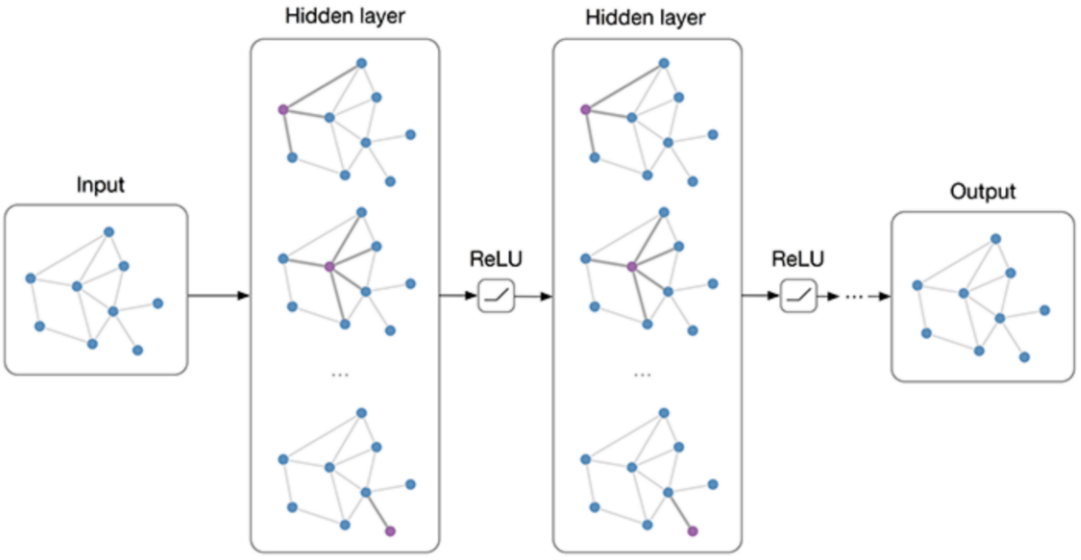

Figure 2.6: An overview of graph neural network structure

whether a given node belongs to a risky customer and identifies potential bad actors in the transaction graph through their direct or indirect connections to nodes that are known to be suspicious. In many cases, following nodes connected to a criminal node has resulted in discovering a group of criminals.

GCNNs, like traditional CNNs, learn features by examining neighbouring nodes. They aggregate node vectors, pass the result to the dense layer, and apply non-linearity through the activation function. In short, GCNNs are composed of graph convolution, linear layer, and nonlinear activation function. The GCN layer's formal expression can be defined as follows:

$$H = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\mathbf{\Theta}\right) \tag{2.1}$$

Where H is the matrix of node representations, X is the matrix, σ (.) is an activation function such as ReLU, A is the graph adjacency matrix with the addition of self-loops, D is the graph degree matrix with the addition of self-loops, and Θ is a matrix of trainable parameters.

Based on the findings from literature reviews, it is obvious that the number of supervised approaches is significantly much more than the unsupervised methods. Also, the number of graph-based techniques has increased in recent years. Figure 2.7 illustrates that the majority of recent publications focus on supervised models in both machine learning and deep learning.

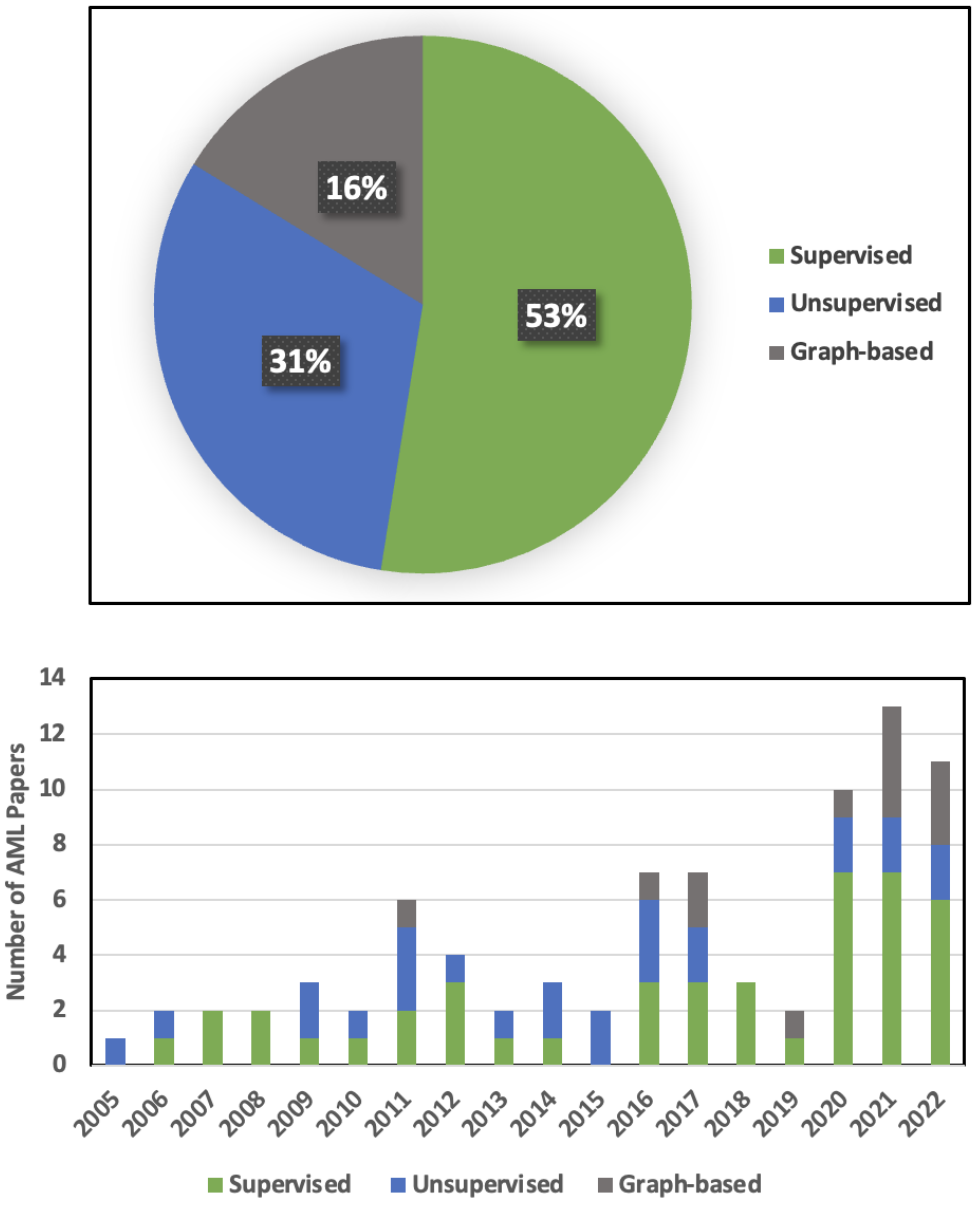

Figure 2.7: Category-wise distribution of AML models

Chapter 3

Data-Oriented Survey of Machine Learning Based AML Solutions

In order to build a machine learning model for a task, there are three crucial factors to consider: the learning algorithm, hyperparameters, and data for training and testing. Existing surveys on AML have focused on classifying the learning algorithms used in AML research papers. Some of these surveys, such as those by Alsuwailem et al. [30] and Labib et al. [29], categorize the papers based on broad machine learning models, such as unsupervised, semi-supervised, supervised, and deep learning. Meanwhile, other surveys, such as those by Chen et al. [26] and Mohammed et al. [22], categorize the papers based on particular algorithms like support vector machines, random forests, and artificial neural networks. However, these surveys tend to neglect data-related aspects of machine learning, which is just as important as the learning algorithm itself. This paper aims to fill this gap by conducting a survey on machine learning-based AML models from a data-oriented perspective and providing a new taxonomy for existing AML solutions.

The main contributions of this survey are:

- We designed a new data-oriented taxonomy for existing machine learning-based AML solutions.
- We conducted a thorough investigation of data-related aspects, including input/output, dataset, and data processing for each existing AML model.

- We compared different AML models in terms of their approach to handling data imbalance issues.
- We provided recommendations for applying appropriate approaches to deal with imbalanced data and selecting performance metrics.

3.1 Data-Related Challenges in AML Research

Data plays a crucial role in machine learning because it is the foundation on which learning algorithms build their models [12]. However, many papers in the field of AML tend to emphasize learning algorithms and overlook the importance of data. There are several important challenges related to data that need to be taken into account when training and evaluating a machine learning model for AML purposes. These challenges include the following:

- Data availability. The first and most important challenge in AML research is the lack of real-world data. The scarcity of large, publicly accessible datasets makes it challenging to train and validate efficient AML models. The confidential nature of financial transaction data, privacy, and security concerns also pose a barrier to obtaining representative datasets for research and development purposes. Financial transaction data contains customer-related information such as names, addresses, and account numbers, which are highly confidential. As a result, banks are often reluctant to share their customer information to maintain confidentiality and avoid data breaches. Hence, using synthetically generated data is an alternative to AML research.
- Limitations of synthetic data. Synthetic data may not accurately reflect real-world scenarios. In other words, using synthetic data to train and validate AML models may not reflect the complexities and nuances of real-world financial transactions, leading to models that are not effective in practice.

- Data size. The size of the data sets used to train and validate AML models is often too small, either real or synthetic. This can lead to models that have poor performance and over-fitting or under-fitting to the data. It is important to have a sufficient number of samples to train and validate machine learning models effectively.

- Lack of positive samples in AML data. Even when real transactions or accounts are available, there is often a scarcity of positive cases (i.e., money laundering transactions or accounts).

3.1.1 Imbalanced Data Problem

In a binary classification problem with two classes, the data imbalance problem is a common phenomenon where samples from one class, called the minority group, are significantly fewer than the other class, called the majority group [113]. However, in many problems, the minority group is the class of interest. For example, in the medical diagnosis task [114, 115] of disease detection, the majority of patients are healthy and are referred to as the negative class, but detecting disease is of greater interest. Learning from these imbalanced data can be difficult, tricky, and problem-dependent [116, 117] due to several reasons.

One major challenge is that standard machine learning algorithms tend to be biased toward the majority class, leading to poor performance in the minority class. This bias occurs because the algorithms are designed to minimize the overall error rate, which can result in a focus on the majority class at the expense of the minority class. Additionally, imbalanced datasets often have very few samples of the minority class, making it difficult for the algorithm to learn patterns and characteristics specific to that class. This can lead to overfitting, where the model learns the noise or other irrelevant features of the data. The severity of the imbalance can also affect the difficulty of learning, as it can be challenging to achieve a balance between different evaluation metrics of the model. Consequently, given the prevalence of data imbalance

issues in real-world applications, it is essential to have a thorough understanding of the class imbalance problem and available techniques to overcome it.

Banking and other financial services are among the most common real-world areas where highly imbalanced data is encountered, particularly in the detection of money laundering and other financial crimes [118]. In this context, the ratio of fraudulent transactions to legitimate ones can be as extreme as one in tens or hundreds of thousands. The extreme disparity between the majority (legitimate transactions) and minority classes (fraudulent transactions) makes money laundering detection challenging since conventional machine learning algorithms can be biased towards the majority class and may fail to recognize the minority class accurately. Furthermore, the exponential growth of daily transactions has exacerbated this situation by increasing the ratio of money laundering cases over regular ones. In AML, the negative and positive classes refer to legitimate and money laundering, respectively. Therefore, the minority class in AML (money laundering transactions/accounts) is also referred to as the positive class; the majority class is the negative class.

Data imbalance can have significant impacts on the training and evaluation of machine learning models. When a model is trained on imbalanced data, it may result in a high misclassification rate for the minority class, as the model will tend to over-classify the majority class due to its increased prior probability [28]. Misclassifying (missing) money laundering transactions/accounts lead to severe consequences for financial institutions, such as fines in the range of millions of dollars.

For example, in 2020, Westpac, one of Australia's largest banks, was fined AUD 1.3 billion (USD 920 million) for breaching AML and counter-terrorism financing laws. The bank was accused of failing to correctly report over 19.5 million international fund transfer instructions, including high-risk transactions that were not flagged. The misclassification of these transactions allowed for potential criminal activity, including child exploitation, to go undetected and lead to significant reputational damage for the bank.

Similarly, in 2021, TD Bank, a subsidiary of Canadian bank TD Bank Group, was fined $122 million by US regulators for allegedly violating AML regulations. The bank was accused of failing to file reports on suspicious activity, including transactions that were red-flagged as potentially fraudulent. The bank also allegedly had deficiencies in its AML program, including a lack of resources and ineffective monitoring of high-risk accounts. The misclassification of these transactions not only resulted in financial penalties but also harmed the bank's reputation and eroded customer trust.

The problem of data imbalance also impacts the choice of performance metrics used in evaluating a machine learning model. Accuracy has typically been used as the primary performance metric in binary classification problems; nevertheless, accuracy can mislead the analyst with high scores for the majority class in the case of imbalanced data that falsely show good performance [119, 120]. For instance, consider a dataset of banking transactions where the minority class makes up only 1/1000 of the total. The machine learning algorithm may achieve an impressive accuracy rate of 99.99%, but it could miss all instances of money laundering. In other words, a high accuracy rate does not guarantee the effectiveness of a machine learning model in AML. It is possible that the system is missing many fraudulent transactions, which could potentially pass through undetected.

In AML, the metrics of the most interest are false negative rate and false alarm rate. A false negative occurs when a fraudulent transaction goes undetected and slips through the system, allowing illegal funds to be used for criminal activities. A high false negative rate can be a significant problem for financial institutions, potentially resulting in severe financial losses, reputational damage, and even legal consequences.

On the other hand, a false alarm happens when the system generates an alarm for a legitimate transaction. A high false alarm rate can be equally problematic, as it leads to a large number of wrong alarms, making the process of identifying suspicious transactions more time-consuming and costly. However, following the AML alarm investigation process can reduce the time needed to investigate system-generated

alarms. The AML investigation process comprises three main stages:

1. **L1 investigation:**The aim of this stage is to quickly assess the alarms and identify false positives. The generated alarms are swiftly investigated and disposed of either as false positives or suspicious alarms. The suspicious alarms are then forwarded to the L2 stage for further investigation.

2. **L2 investigation:** In this stage, L2 investigators review and validate the alarms forwarded by the L1 team. Based on the output, investigators create relevant cases to determine the nature of alarm activity, such as anomalous activity and high-risk transfer, and perform a detailed examination to verify if the transactions associated with the case are part of money laundering activity.

3. **L3 investigation:** In this stage, L3 investigators review the cases passed from the L2 stage and validate their findings. After ensuring accuracy and compliance, they report to the money laundering reporting officer (MLRO) and recommend the filing of a suspicious activity report (SAR).

The manual effort needed for rapid investigation of alarms at the L1 stage can be minimized or replaced by creating a machine learning algorithm to quickly identify and classify alarms into true alarms and false positives. Training the machine learning models with historical data of investigated alarms (L1 alarms) can help investigators save time and effort, thereby building efficiencies in the alarm-clearing process.

In summary, it is necessary to take into account all data-related challenges when developing a machine learning-based AML model. To achieve this, we need to analyze current AML solutions and understand their methods for addressing data-related problems, particularly the issue of imbalanced data. The objective of this survey is to provide a new taxonomy for analyzing existing machine learning-based AML solutions that focus on data factors and encompass a thorough examination of data-related considerations.

3.2 Our New Data-Oriented Taxonomy

We conducted a survey of over 50 papers that use machine learning to identify money laundering transactions/accounts or flag potentially risky customers. Our new taxonomy for analyzing existing machine learning-based AML solutions are based on data, specifically input and output data, dataset, data processing approaches (regarding techniques used to address the problem of imbalanced data), and classification performance metrics. In the following sections, we will describe each of these factors in detail.

3.2.1 Input/Output Data

The following types of data are commonly used as input to AML models:

- **Transactions:** A transaction typically includes information such as the date and time of the transaction, the amount involved, the source and destination accounts, the type of transaction(such as deposit, transfer, cash, credit), and the account balance before and after the transaction.

- **Accounts:** An account usually contains a customer's information, such as a person's name, address, date of birth, identification number, and nationality, as well as financial-related data such as account type, account balance, account activities, and risk score associated with the customer account. Also, an account is associated with a list of transactions conducted by the customer.

- **Other types of input:** Besides considering transactions and accounts, AML models can incorporate network information. Network information provides insight into the connections between transactions and customers, including associations between parties involved and shared funding sources. A common method for representing network information is through the use of graphs, which effectively demonstrate the relationships between transactions and customers. In

this regard, the nodes in a graph represent accounts while the edges illustrate the transactions made between different accounts.

Output, on the other hand, refers to the result of the AML model. For our analysis, we have divided the existing works into two broad categories based on the output data:

- **Binary classification**. The goal here is to classify transactions or accounts into two groups: normal and suspicious. In some cases, a probability score indicating the likelihood of a transaction or customer being suspicious is also provided.

- **Risk assessment**. In risk assessment, a risk score is assigned to a customer account, indicating the likelihood of that customer being involved in money laundering activities. This helps banks allocate resources more efficiently by monitoring high-risk customers more closely.

3.2.2 Datasets

The data used to train a machine learning model is its foundation, and this holds true for AML models as well. It is crucial to take into account three key factors when working with datasets in the AML context: the type of data, the size of the dataset, and the minority/majority ratio of the data. This last factor is particularly important because the data used in AML is often imbalanced, meaning that one class (negative or normal samples) significantly outnumbers the other (positive or money laundering samples).

In the context of AML, there are two types of data that can be employed: real and synthetic. Real data is derived from genuine transactions and customer records and is utilized for training AML models on how to recognize and deter money laundering activities. Since it reflects actual scenarios, this kind of data is often regarded as more reliable and precise than synthetic data. Nevertheless, obtaining real data can

be challenging because banking information is confidential, and its accessibility can be restricted.

Synthetic data, on the other hand, is generated using simulations or models to create artificial transactions and customer records. Synthetic data can be used to complement real data in AML by providing additional scenarios to test and train models. Additionally, synthetic data can be used to overcome the limitations of real data, such as privacy concerns, as the generated data is not tied to actual customers or transactions. However, synthetic data may not accurately represent real-life situations.

In addition to the type of data, the efficiency of a machine learning model in analyzing and handling large amounts of complex transaction data is directly linked to the size of the dataset. Having insufficient data to train a machine learning algorithm may result in overfitting, where the model memorizes the training data instead of generalizing it to new, unseen data. Small datasets can also lead to a lack of diversity, resulting in biased models that perform poorly on diverse new data. Furthermore, small datasets may not be representative of real-world scenarios, leading to models that do not generalize well and perform poorly on real-world data.

The final crucial factor in the dataset is the ratio of minority or positive samples over the majority or negative samples. It is important to assess the distribution of minority and majority samples in the dataset when working with AML models. AML data tends to be highly imbalanced, with the number of suspicious transactions/accounts being much smaller compared to the number of regular transactions/accounts (in the order of 1/1000 to 1/10,000 [121, 122, 123]). This can pose various challenges when training a machine learning model, such as:

- Biased models that have a skewed understanding of the majority class and under-represent the minority class.
- Low detection rates for the minority class.

- High number of false negatives, causing money laundering transactions to be wrongly flagged as normal.
- High number of false positives, leading to an increased number of false alarms and wasted resources.

Therefore, it is crucial to address the issue of class imbalance to ensure accurate and effective classification by the AML model.

3.2.3 Data Processing Approaches

To properly handle an imbalanced dataset in the AML context, the data must be prepared and processed correctly. Data processing usually involves splitting the dataset into training and test sets [124]. However, when dealing with an imbalanced dataset, randomly selecting data for the training and test sets can result in a training set that only contains samples from the majority group. To address this issue, one of the most straightforward solutions is to split the data based on the original minority-to-majority ratio. For instance, if the minority-to-majority ratio is 1/1000, 70% of the data can be used for training and 30% for testing while maintaining the 1/1000 ratio in both sets.

In order to tackle the issue of class imbalance, there are various techniques that can be implemented during the data processing stage, such as feature selection [125, 126, 127] and sampling [128, 36]. Alternatively, techniques can be applied to the machine learning algorithm itself [129] like cost-sensitive learning and threshold adjustment. Another option is to utilize a hybrid approach, which combines multiple techniques [130]. Further information regarding these methods is elaborated on in Section 3.4.

3.2.4 Classification Performance Metrics

Presenting relevant performance evaluation metrics is crucial for any machine learning model. Accuracy, recall, precision, F1 score, and false positive rate (FPR) are

commonly used in existing AML solutions. In the following discussion, we will explain each metric and its suitability for AML tasks, followed by a discussion on appropriate metrics in the AML field.

To properly understand these metrics, it is important to first define the confusion matrix, as all other metrics are calculated using it. In the context of binary classification in machine learning, the confusion matrix, also known as an error matrix, is a table that displays the performance of a supervised learning algorithm. Table 3.1 shows the standard layout of a confusion matrix.

Table 3.1: Confusion matrix

	Actually Positive (1)	**Actually Negative (0)**
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

This matrix has fours cell defined as follows:

- **TP:** true positive, the number of samples being correctly identified as positive out of the total actual positives.
- **FP:** false positive, the number of samples being wrongly identified as positive out of the total actual negatives.
- **FN:** false negative, the number of samples being wrongly identified as negative out of the total actual positives.
- **TN:** true negative, the number of samples being correctly identified as negative out of the total actual negatives.

Accuracy is a simple and widely used performance metric in machine learning for binary classification problems. It measures the proportion of correct predictions

made by the model out of all predictions using equation 3.1:

$$\frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

However, when it comes to evaluating AML systems, accuracy is not a suitable metric. This is because the data used in AML is imbalanced, meaning that the proportion of actual positive cases (TP) is much smaller compared to the proportion of negative cases (TN). In such scenarios, a model could predict all cases as negative and still achieve high accuracy. This high accuracy may not reflect the actual performance of the model in detecting positive cases, which is critical in the context of AML.

For example, consider a scenario where a model is evaluating 10,000 transactions, and only 10 of them are actual cases of money laundering. If the model predicts all transactions as negative (i.e., no money laundering taking place), it will achieve an accuracy of 99.9%. However, this high accuracy does not reflect the fact that the model is not detecting the actual cases of money laundering, which is the primary concern in AML.

False positive rate (FPR) refers to the proportion of actual legitimate transactions (negative samples) that are incorrectly detected as suspicious transactions (positive samples) by the AML model. FPR can be calculated as follows (equation 3.2):

$$\frac{FP}{TN + FP} \tag{3.2}$$

FPR is one of the key metrics for AML systems as it provides information about the accuracy of the model's positive predictions and its ability to prevent false positive alerts. When the FPR is high, it indicates that the AML model is frequently marking many legitimate transactions as possible cases of money laundering. This can lead to increased expenses for the financial institution in reviewing these transactions, as well as decreased customer satisfaction.

Out of the performance evaluation metrics employed in current AML solutions, recall and FPR holds the most significance. Nonetheless, relying solely on these metrics

may not provide a comprehensive evaluation of an AML model's performance. To obtain a complete understanding of a money laundering detection model's effectiveness, two other metrics are crucial: false negative rate (FNR) and false alarm rate (FAR).

False negative rate (FNR) is one of the most important metrics to evaluate the ability of the AML system to accurately detect and prevent financial crimes. FNR refers to the proportion of actual suspicious transactions (positive cases) that are incorrectly identified as legitimate (negative) by the AML system. The formula for FNR is (equation 3.3):

$$\frac{FN}{TP + FN} \tag{3.3}$$

A high FNR means that the AML system is missing many suspicious transactions, which can lead to financial losses for the financial institution, increased financial crime risk, decreased public trust in the institution's ability to detect money laundering cases, and reduced system effectiveness. Therefore, it is crucial to minimize the FNR as much as possible, ideally to zero, to ensure the success and reliability of the AML system.

False alarm rate (FAR), also known as false discovery rate, refers to the proportion of all legitimate transactions that are wrongly identified as suspicious. FAR can be calculated by equation 3.4:

$$\frac{FP}{TP + FP} \tag{3.4}$$

FPR and FAR seem identical, but the main difference is in their denominator. FPR is the ratio of incorrectly identified transactions as a positive class to all legitimate transactions or negative samples (FP + TN). However, FAR is the ratio of incorrectly identified transactions as a positive class to all positive samples identified by the AML system (FP + TP).

FAR is a significant metric for measuring the applicability of an AML model. A low

FAR means that the AML system is accurately recognizing legitimate transactions and avoiding false positive alerts. On the other hand, a high FAR suggests that the system is generating a large number of false positive alerts. This has several implications for financial institutions, such as increased operational costs, decreased customer satisfaction, reduced system effectiveness, missing suspicious transactions as a result of focusing on false alarms, and potential regulatory penalties for failing to detect and report suspicious transactions effectively. Therefore, it is crucial to keep the false alarm rate as low as possible to ensure the success and efficiency of the AML system.

Recall, also known as true positive rate (TPR), measures the proportion of relevant positive cases that the AML system has retrieved out of all positive cases. TPR can be calculated using equation 3.5:

$$\frac{TP}{TP + FN} \tag{3.5}$$

Recall (or TPR) plays an important role in the performance evaluation of the AML system because it indicates the effectiveness of the model in detecting suspicious transactions. A low TPR value suggests that the system is missing many suspicious transactions, which is terrible. On the other hand, a high recall value means that the system has a low number of false negatives, meaning it is not missing many suspicious transactions. A high recall value is important for AML systems because missing even a single suspicious transaction can have serious consequences, such as facilitating financial crimes like money laundering or terrorism financing. As such, it's essential to have an AML system that has a high recall value to ensure that all suspicious transactions are detected and reported.

Precision is a complement for recall which calculates the proportion of correctly identified positive cases (suspicious transactions) among all cases identified as positive by the AML system. The formula for precision is (equation 3.6):

$$\frac{TP}{TP + FP} \tag{3.6}$$

Precision and Recall, together, provide a balance between detecting all suspicious transactions while avoiding false positive alerts. A high precision indicates that the AML system accurately identifies suspicious transactions, whereas a high recall indicates that the system detects all suspicious transactions. While precision and recall are both important metrics, they are often in conflict with each other. For example, increasing recall might result in a decrease in precision and vice versa. Therefore, it is often necessary to balance precision and recall to obtain an optimal model that meets specific requirements or goals.

F1-score is one of the commonly used metrics for evaluating the performance of binary classifiers and provides a balanced measure of precision and recall. F1 score can be calculated as follows (equation 3.7):

$$2 \times \frac{precision \times recall}{precision + recall} \tag{3.7}$$

F1 score combines precision and recall to give an overall measure of a classification model's performance. Precision measures the proportion of true positives (correctly classified as fraudulent) out of all instances that were classified as fraudulent. Recall, on the other hand, measures the proportion of true positives out of all actual fraudulent instances.

On the other hand, false negatives and false alarms are two types of errors that can occur in classification models. As mentioned before, false negatives occur when a fraudulent transaction is incorrectly classified as non-fraudulent, leading to a low recall and a high precision. In contrast, false alarms occur when a non-fraudulent transaction is incorrectly classified as fraudulent, leading to a low recall and a high precision. However, in AML, the cost of these errors is not equal. False negatives have more severe consequences such as financial and reputational damages for the bank,

as they allow fraudulent transactions to slip through the system undetected. For example, TD bank in Canada was fined 122 us million dollars due to breaching AML regulations. However, false alarms can be rectified by further investigation which is a time-consuming process. Therefore, the F1 score is not a suitable metric to evaluate the performance of an AML model as illustrated by the following example.

Suppose we have a dataset consisting of 1 million legitimate transactions and 1000 fraudulent transactions. If we create two models to detect fraudulent transactions, the confusion matrices for Model 1 and Model 2 can be seen in Tables 3.2 and 3.3, respectively.

Table 3.2: Model 1 confusion matrix

	Actual	
	Fraudulent	Normal
Predicted Fraudulent	1000	1,000
Predicted Normal	0	999,000

Table 3.3: Model 2 confusion matrix

	Actual	
	Fraudulent	Normal
Predicted Fraudulent	800	50
Predicted Normal	200	999,950

By computing precision, recall, and F1 score for each of them, it becomes apparent that:

Model 1:

Precision = TP / (TP + FP) = 1000 / (1000 + 1000) = 0.5

Recall = TP / (TP + FN) = 1000 / (1000 + 0) = 1

F1 Score = 2 × (Precision × Recall) / (Precision + Recall) =

$2 \times (0.5 \times 1) / (1 + 0.5) =$ **0.66**

Model 2:

Precision = TP / (TP + FP) = 800 / (800 + 50) = 0.94

Recall = TP / (TP + FN) = 800 / (800 + 200) = 0.8

F1 Score = 2 × (Precision × Recall) / (Precision + Recall) =

$2 \times (0.94 \times 0.8) / (0.94 + 0.8) =$ **0.86**

Although Model 2 has a higher F1 score than Model 1 (0.86 vs. 0.66), it misclassified 200 fraudulent transactions, which lead to severe financial and legal consequences. In contrast, Model 1 identified all fraudulent transactions but also produced 1000 false positives, which can be investigated further to re-classify them as normal transactions.. Therefore, in AML, it is essential to prioritize minimizing the false negative rate, even if it means sacrificing some precision, to avoid missing any fraudulent transactions.

3.3 Review of Existing AML Solutions

In this section, we perform an in-depth analysis of AML solutions through a review of relevant research papers. As mentioned before, our analysis focuses on the data-related aspects, including input, output, dataset, data processing, and evaluation metrics for classification performance. The existing solutions are categorized into two main categories: binary classification and risk assessment. Tables 3.4 and 3.5 list the existing supervised and unsupervised binary classification solutions respectively. Also, Table 3.6 provides a list of risk assessment models covered in the survey.

Table 3.4: Comparison of the dataset, evaluation metrics, and data processing approaches of existing supervised binary classification AML solutions. (I/O = input/output data, m/M = minority/Majority samples, AC = accuracy, F1 = F1 score, P = precision, R = recall, AUC = area under curve, FPR = false positive rate, CM = confusion matrix, T = transaction, A = account, ML = money Laundering, CV = cross-validation)

1	2	3	4	5	6	7	8	9	10	11	12	13
Ref.	**I/O**	**Real data** **Country** **Size** **m/M ratio**	**Hybrid data** **Country** **Size(normal)** **Size(ML)**	**Synthetic data** **Name** **Size** **m/M ratio**	**Data Processing (Train/Test)**	**AC**	**F1**	**P**	**R**	**FPR**	**AUC**	**CM**
[58]	T,A/A		China 5,000 A, 12,000,000 T 80 A (ML)							✓		
[72]	T,A/T	PBC Bank, China Size ✗ m/M ratio ✗										
[47]	T,A/Sequences	China 704 A, 122,926 T 0.1%/99.9% A, 0.01%/99.99% T							✓	✓		
[60]	T,A/A		China 250 A[1] 110 A			✓			✓	✓		
[84]	T/T	BEP, Ireland 2,000,000 T m/M ratio ✗										
[41]	T,A/T	US 92 A[1] m/M ratio ✗										
[76]	T/T	China 122,783 T 1.5%/98.5%							✓			
[55]	T,A/T		Agriculture Bank, China 318,070 T[2] 1,930 T		K-fold CV	✓				✓		
[61]	T/T			Name ✗ 492,983 T 0.01%/99.99%					✓	✓		
[56]	T/A	AUSTRAC, Australia 758,271 T 0.07%/99.93%			10-fold CV						✓	
[112]	T,A/T			AMLSim 1,000,000 A, 9,000,000 T m/M ratio ✗								
[131]	T/T			Name ✗ 5,500 T 0.09%/99.91%		✓						
[132]	T/T			FCA Techsprint Size ✗ m/M ratio ✗		✓	✓					

Continued on next page

[1] They only provide number of accounts. The number of transactions was not reported.

[2] Number of accounts was not reported.

[3] Only the number of money laundering transactions was reported.

Table 3.4 – continued from previous page

1	2	3	4	5	6	7	8	9	10	11	12	13
Ref.	**I/O**	**Real data** **Country** **Size** **m/M ratio**	**Hybrid data** **Country** **Size(normal)** **Size(ML)**	**Synthetic data** **Name** **Size** **m/M ratio**	**Data Processing (Train/Test)**	**AC**	**F1**	**P**	**R**	**FPR**	**AUC**	**CM**
[17]	T/A	SinoPac Bank, Taiwan 747,582 T 0.01%/99.99%			5-fold CV	✓	✓	✓	✓			
[15]	T/T	Norway 33,134 T m/M ratio ✗			10-fold CV		✓		✓	✓	✓	
[38]	T/T			Name ✗ 10,000 T m/M ratio ✗	Random split (70%/30%)	✓						
[73]	T,A/T	UK 330,000 T[1] 43%/57%										
[53]	T/T			Name ✗ Size ✗ m/M ratio ✗		✓						
[109]	T/T	Elliptic 46,564 T 10%/90%			Random split (64%/36%)	✓						
[42]	T/T				Stratified K-fold	✓		✓	✓			✓
[43]	T/T			PaySim Size ✗ m/M ratio ✗	5-fold CV	✓	✓		✓	✓		
[66]	T/T	Akbank, Istanbul 10,916 T 25.48%/74.52%			Random split (60%/40%)		✓	✓	✓	✓	✓	✓
[19]	T/T			PaySim Size ✗ 1/1000		✓	✓	✓	✓			
[104]	T,A/T	China Size ✗ m/M ratio ✗					✓					
[21]	T/T			Name ✗ 2,340 T (ML)[3]	Random split (60%/40%)	✓	✓	✓	✓		✓	
[110]	T/T	Elliptic 234,355 T m/M ratio ✗			Random split (70%/30%)		✓	✓	✓		✓	
[133]	T,A/T	Elliptic 234,355 T 2%/98%		AMLSim Size ✗ m/M ratio ✗	Random split (80%/20%)	✓						

[1] They only provide number of accounts. The number of transactions was not reported.

[2] Number of accounts was not reported.

[3] Only the number of money laundering transactions was reported.

Table 3.5: Comparison of the dataset, evaluation metrics, and data processing approaches of existing unsupervised binary classification AML solutions. (I/O = input/output data, m/M = minority/Majority samples, AC = accuracy, F1 = F1 score, P = precision, R = recall, AUC = area under curve, FPR = false positive rate, CM = confusion matrix, T = transaction, A = account, ML = money Laundering, CV = cross-validation)

1	2	3	4	5	6	7	8	9	10	11	12	13
Ref.	**I/O**	**Real data** **Country** **Size** **m/M ratio**	**Hybrid data** **Country** **Size(normal)** **Size(ML)**	**Synthetic data** **Name** **Size** **m/M ratio**	**Data Processing** **(Train/Test)**	**AC**	**F1**	**P**	**R**	**FPR**	**AUC**	**CM**
[34]	T/A		Agriculture bank, China 318,070 T 1,930 T			✓				✓		
[97]	T/T		China 65,001 T 60 T									
[99]	T,A/A		China 696 A 40 A									
[85]	T,A/T	CE bank, Ireland Size ✗ m/M ratio ✗			Over-sampling Under-sampling							
[87]	T,A/T	Pakistan 8,200,000,000 T, 100,000 A m/M ratio ✗										
[39]	T,A/T	Pakistan 8,200,000 T m/M ratio ✗			Time-based split (10/2 months)	✓						
[96]	T,A/A		Vietnam 12,350 A 25 A									
[79]	T/T	Malaysia 5,006 T 2.6%/97.4%			10-fold CV							
[83]	T,A/A	Prtugal 1,600,000 A m/M ratio ✗			Random split (66%/34%)	✓					✓	✓
[98]	T,A/A			Name ✗ Size ✗ m/M ratio ✗		✓						
[93]	T,A/A			AMLSim 123,350 A 1/10	Random split (95.86%/4.14%)							
[101]	T/T	Malaysia 4,889 T (2.72%/97.28%)			Random split (80%/20%)	✓				✓		
[134]	T/A	Portugal 500,000 T m/M ratio ✗			Time-based split: 60%oldest(train) 10%validation 30%recent(test)			✓	✓	✓		
[91]	T/T			Name ✗ 29,704,090 T (0.27%/99.73%)	Time-based split (7/5 weeks)	✓	✓	✓	✓	✓	✓	

Table 3.6: Comparison of the dataset, evaluation metrics, and data processing approaches of existing risk assessment AML solutions. (I/O = input/output data, m/M = minority/Majority samples, AC = accuracy, F1 = F1 score, P = precision, R = recall, AUC = area under curve, FPR = false positive rate, CM = confusion matrix, T = transaction, A = account, ML = money Laundering, CV = cross-validation)

1	2	3	4	5	6	7	8	9	10	11	12	13
Ref.	**I/O**	**Real data** **Country** **Size** **m/M ratio**	**Hybrid data** **Country** **Size(normal)** **Size(ML)**	**Synthetic data** **Name** **Size** **m/M ratio**	**Data Processing (Train/Test)**	**AC**	**F1**	**P**	**R**	**FPR**	**AUC**	**CM**
SUPERVISED MODELS												
[48]	A/A	China 28 A m/M ratio ✗			Random split 75%/25%							
[46]	A/A			Statlog credit report Size ✗ m/M ratio ✗					✓	✓		
[102]	T,A/A	Italy 599 A, 33,670 T m/M ratio✗										
[49]	A/A	Taiwan A with risk labels: Low=1,500,000 Medium=64 High=22,000			Over-sampling 10-fold CV	✓		✓	✓	✓		
[75]	T,A/A	Scotia bank, Canada A with risk labels: Low = 1,311 Medium = 1,759 High = 1,399			Random split (70%/30%)	✓	✓		✓			
[135]	T,A/Alarms	Estonia 330,000 A, 240,000,000 T 0.4%/96.6%							✓	✓		✓
UNSUPERVISED MODELS												
[94]	T/A	(1)Luxembourg 1,000,000 T m/M ratio ✗ (2)Ripple 14,560,124 T m/M ratio ✗										

The information in the tables is organized as follows: column 2 pertains to the input/output (I/O) data; columns 3 to 5 summarize information about the datasets used in each AML paper; column 6 highlights the data processing approaches used in the AML solutions, and finally, columns 7 to 13 specify the performance evaluation metrics employed by each paper. The content of each column will be discussed in the following.

3.3.1 Input Data of AML Models

AML systems typically accept various data types as input to identify potential money laundering transactions or accounts. The most common types of inputs are customer information and data related to transactions. Customer information can include details such as the customer's name, address, date of birth, and business information, while transaction-related data refers to factors such as the frequency, size, source and destination of the transaction and its type.

While customer information and transaction-related data are the most commonly used inputs, AML systems can also benefit from incorporating less traditional forms of data. This can include analyzing social media data and customer account profiles to detect suspicious behaviour, as well as utilizing behavioural data, such as patterns of transactions, changes in spending habits, or other unusual behaviour.

Graph representation can play a particularly crucial role in AML systems as it effectively visualizes the relationship between customer accounts and transactions and provides valuable behavioural data. As a result, graphs can be integrated into AML models as inputs to enhance their ability to detect potential instances of money laundering.

In terms of input data (second column in Tables 3.4, 3.4, 3.6), our analysis show that:

- Transaction data is a fixed input in all papers. The main reason is that transactions are at the heart of money laundering activities; thus, analyzing them is

essential in detecting and preventing them. By analyzing transaction data, the AML model can identify patterns that may indicate money laundering, such as large, frequent, or unusual transactions. Also, transaction data is required by regulatory bodies to monitor compliance with AML laws and regulations [136, 137]. Therefore, it can be concluded that transaction data is a critical input for AML systems as it provides valuable information for identifying potential money laundering risks and ensuring compliance with AML laws and regulations.

- Some papers (38.5%) in both binary classification (supervised [58, 72, 47, 60, 41, 55, 112, 73, 104, 133] and unsupervised [99, 85, 87, 39, 96, 83, 98, 93]) and risk assessment [102, 75, 135] use both customer account information and transactions. In fact, customer account information and transaction data complement each other in AML systems. Customer accounts mainly provide demographic information used to recognize customers' identities. On the other hand, transactions provide details information on transactions that occurred by each customer, which can be applied to identify suspicious activities. Particularly, in risk assessment papers, customer information plays an important role because the AML model aims to calculate a risk score and assign it to each customer profile. Hence, both customer information and transaction data are significant inputs for designing a robust AML model to accurately detect suspicious money laundering cases.

- Several papers incorporate graphs as an additional input to the transaction or customer accounts [56, 98, 102, 94, 112, 131, 132, 109, 134, 110, 133]. These graphs illustrate the relationships between accounts, where each node represents an account, and edges linked to the node represent the associated transactions. Also, some papers use the actual graph of cryptocurrencies such as Bitcoin [109, 110, 133] and Ripple [94] as their input data.

3.3.2 Output of AML Models

As mentioned before, we categorized existing AML solutions into binary classification and risk assessment methods based on their output. The second column in all Tables (3.4, 3.5, 3.6) provides an overview of the output of each of the AML solutions included in our survey. In the following, we will explain the output produced by each of these approaches.

Binary Classification

The common output of a binary classification AML model is a binary label, which classifies a transaction or an individual as either normal (indicating no suspicion of illegal activity) or suspicious (indicating potential money laundering). In this scenario, a binary classification model takes transactions, customer information, or both as input and is trained to distinguish between two classes of normal and suspicious. Then, it assigns a label to each transaction or accounts in the output.

Our analysis of binary classification AML models revealed that nearly 75% of the papers aimed to detect suspicious transactions in their output, while only one-fourth were interested in detecting suspicious accounts as output [58, 60, 56, 17, 34, 99, 96, 83, 98, 93, 134].

However, Liu et al. [47] proposed a sequence-matching algorithm which identifies suspicious sequences of transactions in the output. Their algorithm compares the sequences of transactions within a peer group, using information such as transaction time, account number, direction, and amount. The high-risk sequences are identified using a probabilistic model and a Euclidean similarity distance score, and a pre-defined threshold is used to extract suspicious sequences. Their algorithm was tested on real financial data from a Chinese institution and showed improvement in performance when the threshold was decreased from 0.9 to 0.6, but this also led to an increase in false positives. The choice of the threshold value is a limitation of the method, as it may vary for each account, leading to trade-offs between the false

positive rate and the detection rate.

Risk assessment

In AML risk assessment papers, a common output is a risk score or risk assessment for each customer account or transaction under investigation [138]. This score is usually generated by a machine learning-based model trained on historical data and features related to the customer, transaction, or both. The risk score reflects the probability or likelihood of the customer or transaction being involved in money laundering activities. These scores are used by AML investigators to prioritize and focus their resources on high-risk customers. Additionally, in some cases, the risk assessment papers may also provide an explanation or justification for the risk score, allowing investigators to understand the reasons behind the score and make more informed decisions [135].

Table 3.6 lists AML papers that concentrate on risk assessment. The majority of these papers aim to determine a risk score for customers [48, 46, 102, 94, 49, 75], resulting in a customer account being the typical outcome. For example, Wang et al. [48] proposed an AML risk evaluation method using a decision tree algorithm. They categorized customer attributes such as industry type, business location, size, and bank product into low, medium, or high-risk levels. The method was used to evaluate the risks of current and future customers, with those at high risk put under strict monitoring.

To enhance the scalability and adaptability of decision tree models for evaluating the risk of customer accounts, Jayasree and Balan [46] proposed a bitmap index-based decision tree (BIDT). This approach uses bitmap indexing to store information and construct a decision tree in a binary fuzzy format. The method determines population frequencies using bitmaps with cardinality rows and columns, and the resulting decision tree is used to identify the customer's transactional region and the occurrence of a risk.

However, Tertychnyi et al. [135] proposed a different approach to risk assessment. They introduced a model that generates alerts for customers by training a machine learning classification model on a dataset of customer financial activities and history within the financial institution. This model produces alerts for accounts based on their classification score.

3.3.3 Dataset

After reviewing the input and output data in the AML papers, it is important to assess the characteristics of their datasets, including whether the data is real or synthetic (type), the number of samples (size), and the proportion of minority to majority samples (simply referred to as the m/M ratio). The dataset characteristics for each paper can be found in columns 3 to 5 of each table (3.4, 3.5, 3.6).

Type of Dataset

The following findings have been determined regarding the type of dataset used in the AML papers:

- Nearly 45% of papers in both binary classification (supervised [72, 47, 84, 41, 76, 56, 17, 15, 73, 66, 104] and unsupervised [85, 87, 39, 79, 83, 101, 134]) and risk assessment [48, 102, 94, 49, 75, 135] use real-world datasets to train and validate their AML model, with 17% of these papers coming from Asian banks in countries such as China, Taiwan, Vietnam, Malaysia, and Pakistan. Nearly 9% use data from European banks, while only 2% have data from North American banks. This information is reported in the third and fourth columns of all tables.

- Approximately 17% of real-world datasets in binary classification AML models include only regular transactions and do not have money laundering transactions, so money laundering transactions were created artificially and added to

the datasets [58, 60, 55, 34, 97, 99, 96]. We call these hybrid datasets, and their information is listed in column 4 of Tables 3.4 and 3.5.

- Some papers use real-world datasets of cryptocurrencies such as Elliptic [109, 110] and Ripple [94]. The Elliptic dataset [139] is a large database of Bitcoin transactions and related information collected and maintained by Elliptic, a leading provider of blockchain intelligence and analytics. The dataset contains information on millions of Bitcoin transactions, including transaction amounts, timestamps, and addresses of the sender and receiver. Similarly, the Ripple dataset provides a collection of data related to the Ripple cryptocurrency and its transactions.

- One-fourth of papers (25%) in both binary classification (supervised [23, 112, 131, 132, 38, 53, 43, 19, 21] and unsupervised [98, 93, 91]) and risk assessment [102] groups utilized synthetic data, with some of these datasets being publicly available for fraud detection purposes, such as PaySim [140], and AMLSim [141, 112, 142]. This information can be seen in column 5 of Tables 3.4, 3.5, 3.6.

- Some paper did not provide any information about the data they used [42], except that the data is real [104, 85] or synthetic [53, 98, 46].

Size of Dataset

As the volume of transactions continues to rise, having a large dataset is essential in accurately reflecting the real-world scenarios of money laundering detection. Nonetheless, processing large datasets also demands substantial computing power and can be quite time-consuming. Our analysis of existing AML solutions in terms of the size of the dataset showed that:

- A substantial number of AML solutions (nearly 22%) have been trained or validated on small datasets, with less than 10,000 transaction records [60, 15, 38, 63, 66, 41, 79, 101, 99, 93]. This can lead to quick performance for the detection

process, but it is not representative of real-life scenarios. These methods cannot be fully trusted for evaluation and comparison purposes, as their performance time may not be reliable. It is imperative to re-evaluate these methods using a sufficient amount of data to determine if they can handle larger datasets with a larger number of transactions.

- A number of papers lacked complete information regarding the size of their datasets. For instance, they only reported the count of artificially generated money laundering transactions [21].

Proportion of Minority to Majority Samples (m/M ratio)

Given the inherent class imbalance in AML data, it is crucial to determine the proportion of normal and money laundering transactions, also known as the minority/-Majority ratio (m/M). Our investigation regarding the m/M ratio revealed that:

- By analyzing columns 3 to 5 of the binary classification (Tables 3.4 and 3.5) and comparing them to the risk assessment (Table 3.6) solutions, it becomes evident that both types of models face the challenge of dealing with imbalanced data.

- Using either real or synthetic data, approximately 40% of papers did not report the m/M ratio, which is an important factor in evaluating the classification performance of an AML model.

- In all other papers [60, 47, 23, 131, 17, 109, 66, 19, 93, 101, 91, 135], the m/M ratio indicates that the input data is highly imbalanced, which is a common scenario in real-world financial systems.

Data Processing

Data processing is a critical component in the machine learning workflow. It involves transforming and cleaning the data, so it is in a format that can be utilized for training

and evaluation of machine learning models. The specific data processing steps will vary depending on the type of data, the machine learning algorithms being used, and the research question being addressed. Therefore, data processing is a crucial step in the machine learning process and must be performed carefully to ensure that the results are meaningful and accurate.

With regard to data processing, our analysis focused on two main aspects of existing AML papers:

1. How they divide the dataset into training and testing sets, given the imbalanced nature of AML data, which has a significant impact on this task.

2. How they address the imbalanced data problem while training a model.

After reviewing the existing binary classification and risk assessment AML papers, we found that less than half of them (43%) mention their method of dividing the dataset into training and testing sets. A majority of the papers did not provide any information on how they prepared the data for training and evaluating their AML models. The information about the data processing techniques used by each paper is provided in column 6 of all tables (3.4, 3.5, 3.6). These techniques include:

- Random splitting of the original dataset into training and testing data. Several papers [38, 109, 66, 19, 110, 83, 93, 101, 48, 75] performed traditional random splitting of the dataset, which is unlikely to preserve the m/M ratio, especially when the m/M ratio is very small (e.g., 1/1000). In this case, a random 80/20 or 70/30s split may result in one set not having any money laundering (minority) cases.

- K-fold cross validation [55, 56, 17, 15, 43, 79], where $k = 5$ or $k = 10$. Randomly splits the input data into k subsets of data (also known as folds). The machine learning model is trained on all but one subset (i.e., on $k1$ subsets) and evaluated on the subset that was not used for training [143]. This technique still suffers

from the problem of a random selection of data and not maintaining the natural distribution of the dataset.

- Stratified k-fold cross-validation [42]. Follows the same algorithm as k-fold cross-validation but maintains the natural distribution of the original data in each fold.

- Splitting based on transaction time [39, 134, 91, 135]. For example, Tertychnyi et al. [135] used a real-world dataset containing 81 continuous weeks of transactions that occurred in a bank. They selected the first 56 weeks of data for training and the following 25 weeks for testing.

It is worth mentioning that only a few of the papers in our survey [85, 49, 135] discuss how they addressed the class imbalance issue. For instance, Le et al. [85] used a genetic algorithm to increase the number of minority samples, a technique known as oversampling. Also, Chen et al. [49] utilized the synthetic minority over-sampling technique (SMOTE) to ensure an equal number of customers in each risk level. On the other hand, Tertychnyi et al. [135] employed a two-layer approach to AML. In the first layer, they performed under-sampling by removing customers who did not display behaviours similar to reported customers to tackle the problem of imbalanced data.

Metrics for Performance Evaluation

The selection of performance metrics is crucial in designing a machine learning model for AML. Our survey results, as shown in columns 7 to 13 of Tables 3.4, 3.4, and 3.6, indicate the following regarding the performance metrics used in existing AML solutions:

- A substantial number of papers (28.8%) failed to provide any performance metrics to report their results in the groups of supervised binary classification [72,

84, 41, 112, 73], unsupervised binary classification [97, 99, 85, 96, 79, 93], and risk assessment [48, 102, 94]. This lack of information makes it difficult to properly evaluate the effectiveness of their models.

- Some papers [131, 38, 53, 109, 39, 98] only used accuracy as a performance metric, which is an inappropriate choice for AML due to the imbalanced nature of AML data.

- Only 19% of the papers reported the false positive rate, which is an important metric in AML. This metric indicates the number of legitimate transactions wrongly classified as suspicious. While FPR can provide some information about the performance of the AML model with positive samples, it is not sufficient on its own.

- Some papers [47, 15, 43, 66, 134, 91] reported recall in addition to FPR, which provides a better understanding of the model's performance in accurately detecting positive samples.

- Some papers provide AUC (area under the ROC curve) as their performance metric [56, 15, 66, 21, 110, 83]. The ROC (receiver operating characteristic) curve plots the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds. AUC summarizes the overall performance of a classifier by considering both the true positive rate and the false positive rate. However, for imbalanced data, the precision-recall (PR) curve is more informative [144]. The PR curve plots precision against recall for different thresholds, allowing practitioners to see the trade-off between precision and recall. In AML, having high recall is important, as even a small number of missed suspicious transactions can result in significant losses or penalties. The PR curve helps practitioners determine the threshold that balances precision and recall for their specific AML problem.

It is clear from our study that the problem of imbalanced data is frequently ignored in the preparation of data and the reporting of performance metrics in Anti-Money Laundering (AML) models. Despite its significant impact on the efficiency of these models, very few solutions take this issue into account. In order to counter the negative effects of imbalanced data, the following section provides various methods and approaches to tackle this challenge. Implementing these solutions can help reduce the impact of imbalanced data on the performance of AML models and enhance their efficiency.

3.4 Solutions to the Imbalanced Data Problem

This section provides a brief overview of existing techniques to overcome the class imbalance problem. There are three main categories of techniques: preprocessing, algorithm-level, and hybrid techniques [28]. The classification of different strategies for handling imbalanced data is shown in Figure 3.1.

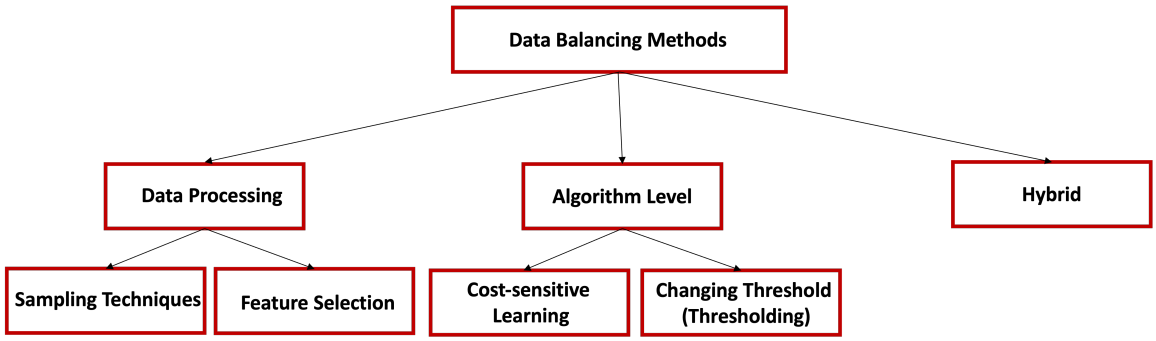

Figure 3.1: Taxonomy of various strategies to handle imbalanced data

3.4.1 Preprocessing Techniques

Preprocessing techniques are those that operate on the training data, also known as data-level techniques. The goal of these methods is to reduce the imbalance between classes by directly manipulating the data. Preprocessing approaches can be further classified into two categories: sampling methods and feature extraction.

Sampling Methods

Sampling is a straightforward and widely used approach to balance the class distributions of the training data. It involves changing the size of the training data set by adding or removing samples to mitigate the impact of imbalanced data. There are several sampling techniques:

- **Over-sampling:** The goal of over-sampling is to create a balanced dataset by increasing the size of the minority class. A simple over-sampling approach is to randomly duplicate the samples of the minority class, known as random over-sampling (ROS). However, this method can lead to over-fitting [128]. One of the most widely used over-sampling techniques is the synthetic minority over-sampling technique (SMOTE), proposed by Chawla et al. [145]. SMOTE creates new minority samples by interpolating between real minority samples and their nearest minority neighbours. There are several variants of SMOTE, such as borderline-SMOTE [146] and safe-level-SMOTE [147], that aims to improve the original SMOTE algorithm. Borderline-SMOTE focuses on over-sampling near the class boundaries, while safe-level-SMOTE establishes safe zones to prevent over-sampling in overlapping or noisy regions. An overview of over-sampling is shown in Figure 3.2.

- **Under-sampling:** The goal of under-sampling is to create a balanced dataset by reducing the size of the majority class. One of the simplest methods is called random under-sampling (RUS), which involves randomly selecting a set of samples from the majority class and ignoring the rest [148]. This approach helps create a balanced ratio of classes and speeds up the training phase. However, it can also result in losing valuable information from the majority class by removing some of its instances. To address this, there are intelligent under-sampling techniques that aim to preserve important information for learning while still reducing the number of majority class samples. For example, the

near-miss algorithm, developed by Mani et al. [149], calculates the distance between majority class samples and minority class samples and uses a K-nearest neighbour (KNN) classifier to determine which majority samples to remove based on these distances. This approach has inspired other methods, such as one-sided selection and Wilson's editing. Figure 3.3 provides an overview of under-sampling.

- **Hybrid sampling:** The hybrid methods use a combination of over-sampling and under-sampling to balance the class distribution of the data. These techniques merge the two resampling methods to tackle the challenge of imbalanced data classification [150, 151].

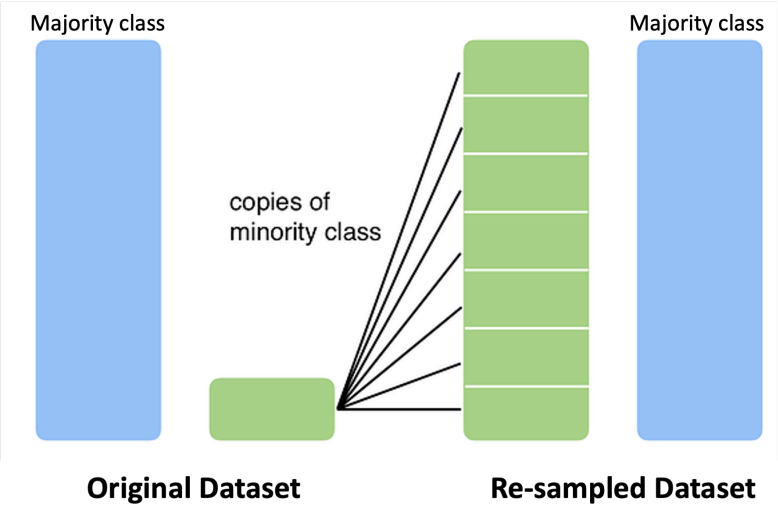

Figure 3.2: An overview of over-sampling

Although sampling techniques have proven to be suitable for imbalanced data, the choice of approach to tackle this issue is highly problem-dependant. There is no guarantee that any single sampling technique will work well for all imbalanced datasets. Therefore, the choice of technique for handling imbalanced data requires careful consideration of the characteristics of the specific problem and dataset, including the distribution of class sizes and the potential impact of losing or duplicating

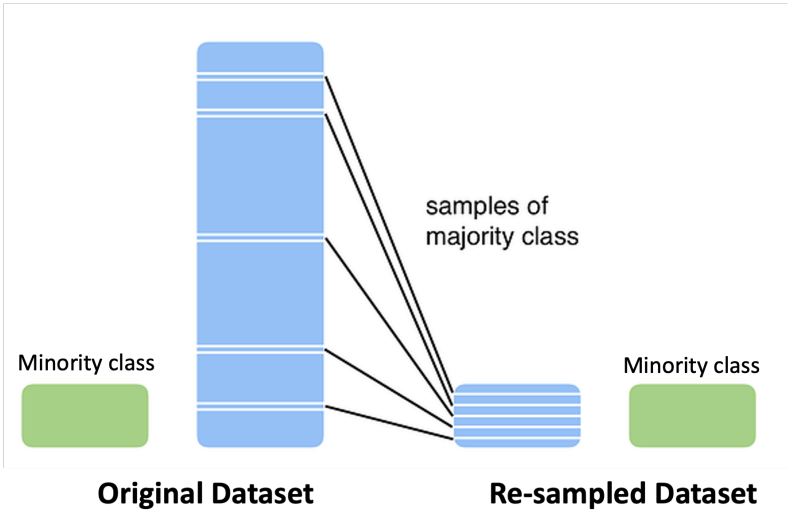

Figure 3.3: An overview of under-sampling

data. It's often necessary to try multiple techniques and compare their results in order to determine the best approach for a given dataset.

Feature Selection

Feature selection is a preprocessing technique that is used to handle imbalanced data in machine learning. The main idea behind feature selection for imbalanced data is to select a subset of the most relevant and informative features from a high-dimensional dataset in order to improve the performance of a classifier on imbalanced data [126, 125, 127]. By removing irrelevant or redundant features, feature selection can help reduce the dimensionality of the dataset and thus make the problem of imbalanced data easier to handle.

There are several feature selection techniques that can be used for handling imbalanced data, such as filter methods, wrapper methods, and embedded methods [152, 153]. The choice of approach is highly dependent on the specific problem and dataset, and there is no guarantee that any one technique will work best for all imbalanced data problems.

While using data-level techniques to handle imbalanced data distributions is a common and simple approach, it also presents various challenges [28]:

- Over-sampling and under-sampling can result in over-fitting and loss of information, respectively.
- Selecting the optimal class distribution in a dataset is crucial because it impacts the performance of the classifier and varies based on the specific dataset.
- Effectively resampling the data is also a challenge, as it may not always be suitable in all situations.

Algorithm-Level Approaches

In contrast to data preprocessing or data-level techniques, algorithm-level methods do not manipulate the data distributions to tackle class imbalance. Instead, they focus on increasing the importance and cost of the minority class in the learning or decision-making process. This can be achieved through techniques such as cost-sensitive learning or threshold adjustments.

Cost-sensitive learning methods assign higher penalties for misclassifying instances belonging to the minority class, thereby reducing the likelihood of such errors [116]. However, this approach is more complex and less common than sampling techniques because the costs associated with misclassification cannot be easily determined from data [154, 155]. These methods can be applied to various machine learning models such as neural networks [155], SVMs [156, 157], decision trees [158], and fuzzy rules [159].

On the other hand, threshold changing or post-processing techniques alter the decision boundary of the classifier to minimize the impact of class imbalance. This method involves converting the learning algorithm from cost-insensitive to cost-sensitive [129].

Hybrid Techniques

Another solution for effectively handling class imbalance is combining both data-level and algorithm-level techniques [130]. These methods typically involve two key steps:

- Performing data sampling to reduce data imbalance and noise.
- Implementing cost-sensitive learning or thresholding to further reduce bias towards the majority class.

For example, Liu et al. developed EasyEnsemble and BalanceCascade, which use combinations of subsets of the majority class and the minority class to create pseudo-balanced training sets for multiple classifiers. Other hybrid models for imbalanced data problems include SMOTEBoost [145], DataBoost-IM [160], and JOUS-Boost [161]. A comprehensive survey of techniques for handling imbalanced class problems, including hybrid methods, was provided by He et al. [117].

3.5 Conclusions

Building a machine learning model for Anti-Money Laundering (AML) requires consideration of three essential factors: the learning algorithm, hyperparameters, and data for training and testing. Previous surveys on AML have focused on categorizing the learning algorithms used in AML research papers. However, these surveys often overlook the significance of data-related aspects of machine learning, which are equally important to the learning algorithm itself. This paper aims to bridge this gap by conducting a data-oriented survey of machine learning-based AML models and presenting a new taxonomy for existing AML solutions.

We have divided the existing AML solutions into two categories based on their output: binary classification and risk assessment. Binary classification models aim to classify transactions or customer accounts as either legitimate or suspicious. On the other hand, risk assessment models assign a risk score to each customer. In our analysis of AML papers, we considered factors such as input and output data, datasets, data processing, and performance evaluation metrics.

Our analysis reveals several key issues in the field of AML. Firstly, there is still a lack of publicly available datasets that are appropriate for AML tasks, leading to

the use of synthetic samples that may not accurately reflect real-world situations. Secondly, the imbalance in the data, with a significantly lower number of money laundering cases compared to normal transactions, is often underestimated in the AML literature. This can result in models that are not robust enough to handle real-world scenarios. Finally, important performance evaluation metrics such as false positive rate, false negative rate, and false alarm rate are not fully reported in many AML papers.

Chapter 4

N2V-GCN: The Proposed AML Model

This chapter provides a detailed description of our AML model, which we have named N2V-GCN, as it incorporates both node2vec and graph convolution network techniques. Figure 4.1 provides an overview of N2V-GCN.

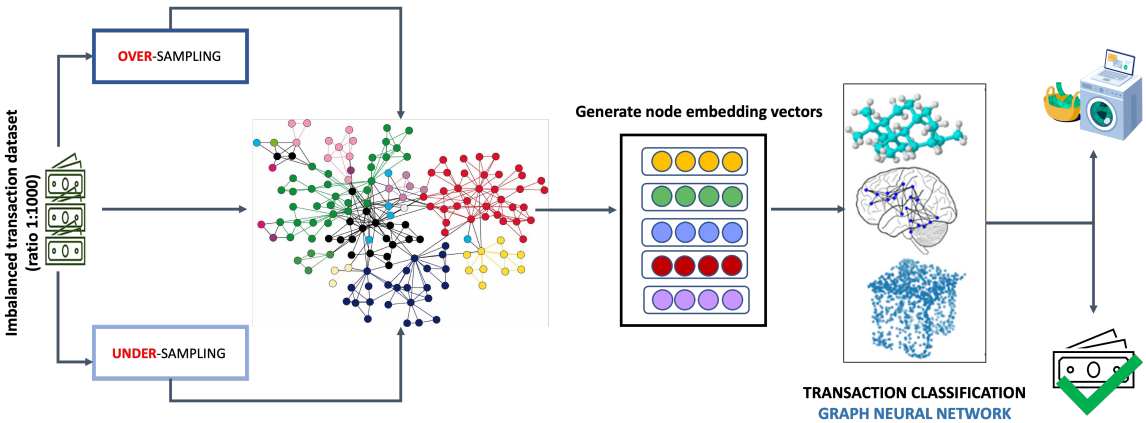

Figure 4.1: A big picture of N2V-GCN

Initially, we will present a detailed explanation of our proposed model, providing a step-by-step account of its intricacies. Subsequently, we will clarify the specifics of our data preparation procedure, including our methodology for managing imbalanced dataset, followed by a discussion of our approach to reporting the results.

4.1 N2V-GCN and Parameters

This section outlines the different components of the N2V-GCN model, which includes: constructing a graph of transactions, generating feature vectors for each node in the graph, and classifying nodes into regular and money laundering classes.

4.1.1 Constructing Transaction Graph

For this thesis, we employed a dataset that contains various attributes of banking transactions, such as transaction amount, type, source and destination account numbers, and account balances before and after the transaction. However, instead of using the raw transaction data, we employed a transaction graph, where each node represents a banking account and the edges between them represent the transactions made by these accounts.

The usage of graph representation is a straightforward yet effective method of storing data that provides a comprehensive overview of money flows, enabling us to identify any suspicious activity for further analysis. As per Battaglia et al. [162], graph networks possess two key capabilities, relational reasoning, and combinatorial generalization, which allow them to mimic human-like abilities. Relational reasoning helps to identify the connections between different entities, while combinatorial generalization provides new inferences and predictions based on the known behavior of the entities.

4.1.2 Generate Graph Embeddings: Node2vec

Once we constructed the transactions graph, we applied a technique called node embedding to transform nodes and relationships in the graph into a set of vectors that can be used as features for the graph convolution network [163]. Node embedding techniques aim to capture all important information such as vertices, vertex-to-vertex relationships, graph topology, and other relevant features.

In this thesis, we used node2vec [164], a widely-used node embedding technique,

to generate embedding vectors for the transactions graph. Node2vec is a modified version of deepwalk, an earlier graph embedding approach. The deepwalk algorithm creates random walks on the graph and uses skip-gram language modeling to generate node embeddings. Node2vec, on the other hand, introduces a flexible notion of node neighborhoods to generate embeddings that capture both local and global information about the graph structure. Compared to other node embedding techniques, node2vec has been shown to be effective in capturing complex relationships between nodes in various types of graphs [163], making it a suitable choice for our proposed model. To gain a better understanding of node2vec, it is necessary to first comprehend the idea behind the deepwalk algorithm.

Deepwalk is a technique that employs graph theory to navigate a graph by following a sequence of walks. These walks involve moving from one node to another as long as they share a common edge. For instance, to explore the area around a particular node, deepwalk starts at a randomly chosen node and walks randomly to a neighboring node until it reaches a maximum path length. To traverse the entire graph, deepwalk performs several random walks based on Equation (4.1).

$$Pr\Big(u_i|\big(\Phi(u_1),\Phi(u_2),...,\Phi(u_{i-1})\big)\Big) \tag{4.1}$$

where u_i denotes the i_{th} node in the graph, Φ is a mapping function that provides the latent representation associated with each node u in the graph, and Pr is the probability of observing the node u_i given all its previous node observations through random walks.

After generating the series of random walks, deepwalk uses the skip-gram [165] approach, which is an embedding technique of natural language processing, to generate the final embedding vector for each node of the graph. In the text content, the input of skip-gram is a sentence. It selects a target word in the sentence to predict its "context" or neighboring words. In our case, the input of skip-gram is a series of random walks containing a sequence of the connected nodes in the graph. Hence, the

skip-gram technique treats a node and a sequence of nodes (series of random walks) as a word and a sentence, respectively.

The idea of deepwalk is easy to understand and implement while it comes with some limitations. One of the primary limitations is that it generates walks in a completely random manner, with no control over the walk path. This technique can result in a high degree of randomness, resulting in suboptimal representations of the nodes in the graph. Moreover, deepwalk may be unable to capture more complicated relationships within the graph, such as transitive or hierarchical relationships.

Grover et al. [164] proposed a solution to the randomness issue of deepwalk, called node2vec. Node2vec is an improved version of deepwalk that employs the concept of biased random walk to alleviate the randomness problem. They accomplished this by introducing a second-order random walk with two parameters:

- **In-out parameter (q).** The in-out parameter (q) controls the probability of moving away from the previous node and allows for inward and outward exploration. If q is greater than 1, the random walk will take an inward approach and focus on local nodes close to the starting node, similar to breadth-first sampling (BFS), which explores the immediate neighborhood of the starting node. If q is less than 1, the random walk will take an outward approach and explore nodes further away from the starting node, similar to depth-first sampling (DFS), which explores the graph globally for neighborhood nodes at increasing distances from the starting node.

- **Return parameter (p):** The return parameter (p) determines the likelihood of revisiting a node during a walk and affects the breadth-first-search (BFS) sampling approach. A high value of $p(> max(q, 1))$ reduces the chance of revisiting a node in the next step, preventing duplicate sampling. Conversely, a low value of $p(< min(q, 1))$ causes the walk to backtrack, resulting in local walks that stay close to the starting node u.

These parameters, in general, serve as guides for the walk and determine the pace at which it explores and moves beyond the starting node's vicinity. Figure 4.2 demonstrates how p and q can regulate the process of walk generation in node2vec. Suppose a walk has just traversed the (t, v) edge in Figure 4.2 and has arrived at node v to decide on its next step. The walk must calculate the transition probabilities π_{vx} for edges (v, x) leading from v. According to Grover et al. [164], they define the transition probability as $\pi_{vx} = \alpha_{pq}(t, x).w_{vx}$, where w_{vx} denotes the edge weight. They determine $\alpha_{pq}(v, x)$ as follows:

$$\alpha_{pq}(v,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{4.2}$$

where d_{tx} represents the distance along the shortest path between nodes t and x. The value of d_{tx} must fall within the range of 0, 1, or 2, so the two parameters (p and q) are essential and sufficient to direct the walk.

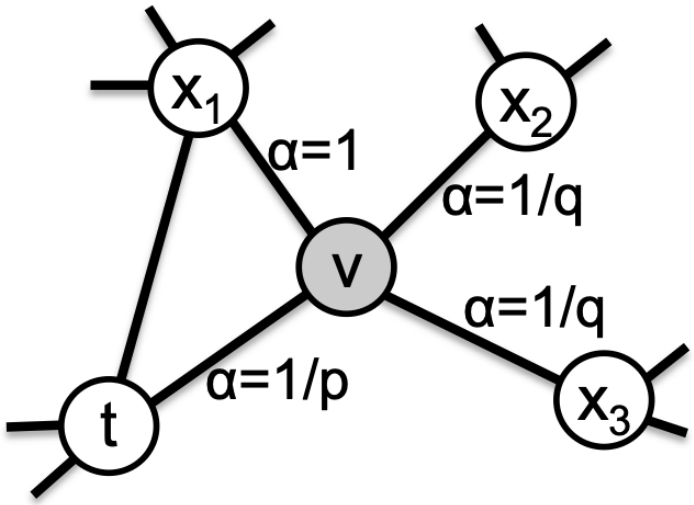

Figure 4.2: Random walk procedure in node2vec. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α [164].

In summary, a large value of p causes the random walks to explore widely, while a

small value keeps them close to the starting node (locally). Similarly, a small value of q encourages exploration, while a large value restricts the walk to a local area [164].

Therefore, using the biased random walks, node2vec interpolates between BFS and DFS to produce sequences of random walks with fixed lengths. Assuming that c_i is the ith node in a walk sequence with $c_0 = u$ as the starting node, node c_i can be generated using the distribution:

$$P\left(c_i = x \mid c_{i-1} = v\right) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

In the output, node2vec produces sequences of nodes that are then used by the skip-gram algorithm to create embedding feature vectors for each node in the graph [165]. The skip-gram algorithm takes each node in the sequence and tries to predict its neighboring nodes based on a fixed-size window of nodes surrounding the input node called the context window. It learns the probability distribution of co-occurrence of nodes within the context window and uses it to create an embedding feature vector of length d for each node in the graph, where d is the dimensionality of the embedding space. The feature vectors are learned by optimizing a negative log-likelihood objective using stochastic gradient descent.

Once the skip-gram algorithm has learned the embedding feature vectors for all the nodes in the graph, they can be used for downstream tasks such as node classification, clustering, and link prediction. In this thesis, the feature vectors were fed into a classifier along with the associated graph as input.

4.1.3 Learning Algorithm: Graph Convolution Network

This section explains the learning algorithm that is used to detect money laundering in detail. This thesis leverages a graph convolution network (GCN) algorithm proposed by Kipf et al. [166] as it is suitable for working with graph-based data. The primary

objective of the GCN algorithm is to classify transactions as normal or suspicious.

The GCN algorithm consists of three main layers: input layer, convolution (or hidden) layer, and output layer, which are illustrated in Figure 4.3. At the input layer, the GCN algorithm takes two inputs: embedding vectors generated using the node2vec algorithm, and the adjacency matrix of the graph. The adjacency matrix is a matrix that represents the connections or relationships between nodes in a graph, and it is commonly used to represent a graph mathematically.

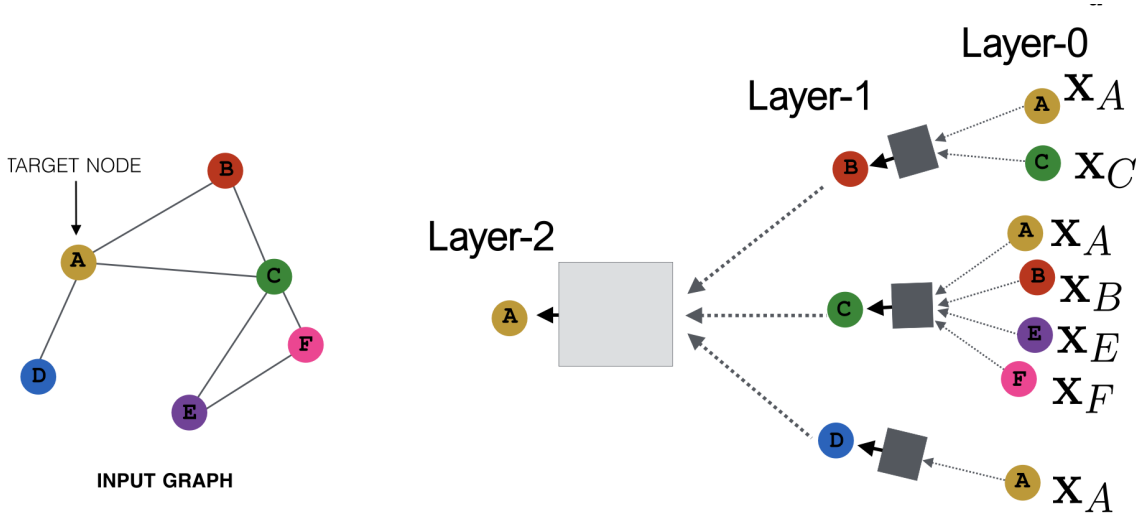

Figure 4.3: Nodes have different embedding in each layer

In our model, the embedding vectors generated by the node2vec algorithm are used as a feature set to classify transactions. These embedding vectors form a matrix of numerical values, where each row represents the embedding vector for a specific node in the graph. This matrix is depicted in Figure 4.4.

The next layer is the convolution layer which is the key component of GCN. Its primary objective is to produce node embeddings based on the local neighborhoods. To achieve this, the node feature matrix goes through a layer-wise neural network in GCN, which aggregates information from neighboring nodes. Figure 4.5 illustrates how GCN aggregates the local neighborhoods of nodes in an input graph using their associated embedding feature vectors.

At each layer of neighborhood aggregation, an embedding vector is generated for

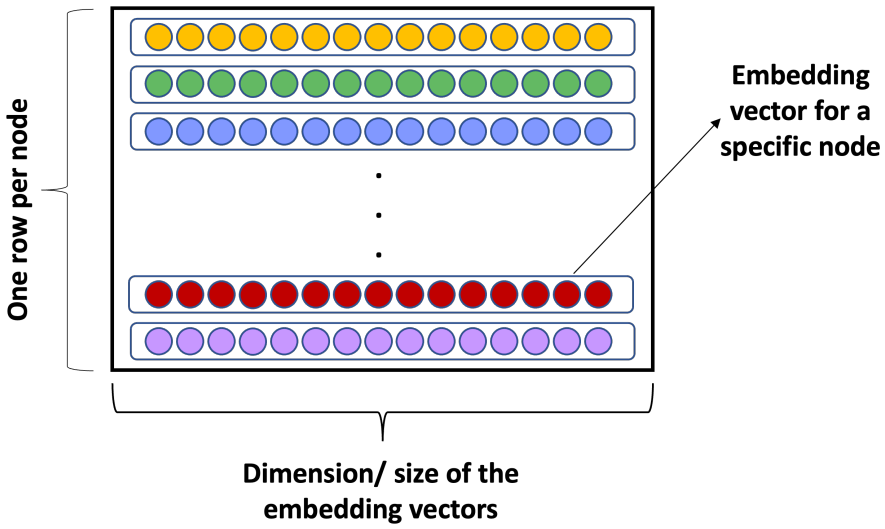

Figure 4.4: Resulting matrix for node's embeddings obtained by node2vec

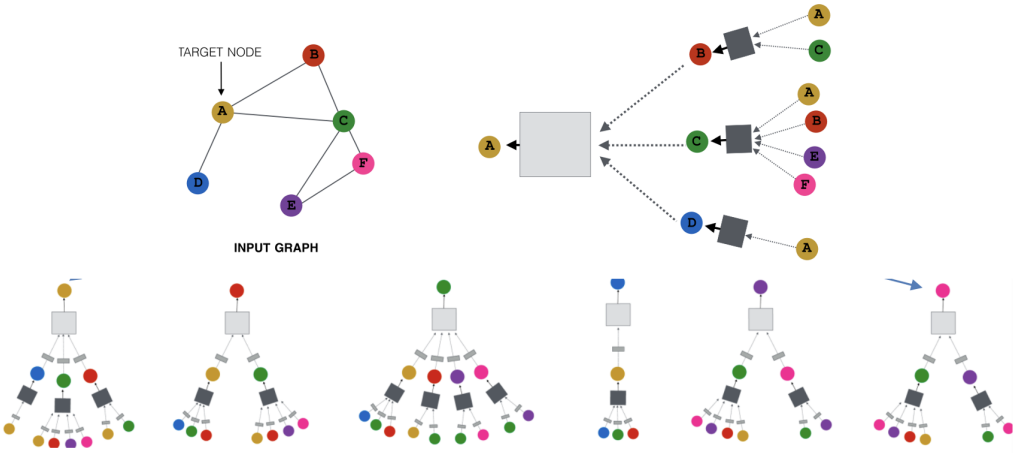

Figure 4.5: For each node, GCN aggregates the local neighborhoods

each node. As a result, nodes have different embeddings at each layer, as depicted in Figure 4.3. For instance, in layer-0 (or the input layer), node embeddings are essentially the input matrix X that represents the initial features of the node network.

In each layer, the neural network performs the propagation step while learning a set of weights for the input data. This process repeats for all layers in GCN, increasing the size of the local neighborhood used to calculate embedding for each node. This type of computation is a first-order approximation of the local spectral filters on the

graph, which improves computational efficiency [167].

Each layer of approximation in GCN is computed by equation 4.4,

$$H^{(i+1)} = \sigma\left(D^{\frac{-1}{2}}AD^{\frac{-1}{2}}H^iW^i\right) \tag{4.4}$$

where $H^{(i+1)}$ refers to the hidden layers in the GCN that compute embeddings by considering the neighbors of each node. W^i is a layer-specific trainable weight matrix, and $\sigma(.)$ is the activation function such that $ReLU(.) = max(0, .)$. D is the diagonal degree matrix, and A is the identity matrix I added to the adjacency matrix A_{adj} (Equation 4.4).

$$A = I_n + A_{adj} \tag{4.5}$$

The role of the identity matrix I is to add self-edges to all nodes, allowing embeddings to be calculated from previous layers. After adding self-loops, we see in Equation 4.4 that the adjacency matrix is normalized by multiplying by the inverse of the diagonal degree matrix $D^{\frac{-1}{2}}AD^{\frac{-1}{2}}$. This process is called spectral filters and ensures that feature vectors maintain the same scale between propagation steps.

Following the convolution layer, a fully-connected layer combines the information from all nodes in the graph to make a final prediction or decision. In the output layer, there are two neurons to decide whether a transaction is suspicious or normal.

4.2 Data Preparation

The effectiveness of a machine learning model is significantly impacted by the quality of the data used during training. Consequently, creating a reliable machine learning model requires careful data preparation. Furthermore, it is essential to address the problem of imbalanced data in AML data because it can significantly affect the performance of a machine learning model. This thesis presents a hybrid approach to mitigate the skewed data distribution in our dataset. The proposed solution com-

prises two parts: data level (pre-processing) and algorithm level (post-processing). First, we will elaborate on the data level component.

4.2.1 Handling Imbalanced Dataset: Pre-processing

As discussed in section 3.4.1, there are several techniques available for mitigating the problem of imbalanced data. However, the selection of an appropriate technique depends on the specific problem at hand [168]. Therefore, there is no fixed rule to determine which technique should be used for a given problem.

In the pre-processing stage, we used sampling methods from the data level techniques to assess the impact of class distribution on the learning model. The original ratio of our dataset was 1/1000, and we modified the natural class distribution to create datasets with nine different class distributions, ranging from 1/1 to 1/500. To achieve this, we used sampling techniques from both under-sampling and over-sampling methods. To avoid random sampling, we applied two of the most commonly used techniques, namely near-miss and Synthetic minority over-sampling technique (SMOTE), for under-sampling and over-sampling, respectively. In the following sections, we will explain how these techniques work.

Under-sampling: Near-miss

To prevent randomness and avoid potential information loss caused by under-sampling, the near-miss algorithm aims to create a balanced dataset by removing samples from the majority group based on the nearest neighbor algorithm. The near-miss algorithm can be summarized in three steps:

1. Calculate the distances between all instances of the majority class and the instances of the minority class.

2. Select **k** instances of the majority class with the smallest distance to the minority samples.

3. Construct the final dataset. For example, if the original dataset has **n** minority samples, then near-miss will result in $k \times n$ instances from the majority group.

Figure 4.6 illustrates the sample selection process in near-miss.

Figure 4.6: The procedure of near-miss algorithm

Over-Sampling: SMOTE

SMOTE stands for synthetic minority oversampling technique. Instead of duplicating minority samples based on existing ones, SMOTE creates new instances based on the nearest neighbors as shown in Figure 4.7. This helps the model avoid overfitting. The SMOTE procedure can be summarized as follows:

1. Randomly select a minority class instance (**a**) and find its k-nearest minority class neighbors.

2. Randomly select one of the k-nearest neighbors (**b**) and connect **a** and **b** to form a line segment in the feature space using linear interpolations.

3. Generate a synthetic instance as a convex combination of the two instances **a** and **b**.

Figure 4.7 depicts the SMOTE procedures. In this thesis, the main reason for applying SMOTE in addition to near-miss is to determine the effectiveness of different

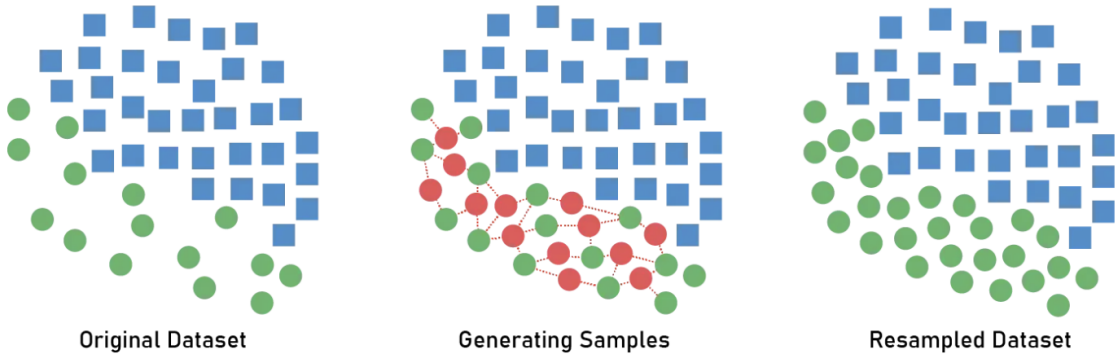

Figure 4.7: The procedure of SMOTE algorithm

sampling techniques on the performance of our model and compare it to the natural distribution of the dataset.

As a post-processing step, we adjusted the threshold of the classifier algorithm. The default threshold may perform poorly for classification issues with a significant class imbalance. Therefore, changing the threshold used to translate probabilities to class labels is a straightforward way to improve the performance of a classifier that predicts probabilities on an imbalanced classification problem. In the following section, we will explain how to find the optimal threshold for N2V-GCN through the post-processing step.

4.3 Post-Processing and Reporting Results

To determine the effectiveness of our model, we conducted different experiments with ten class distribution including the natural distribution of our dataset. As mentioned before, we applied a post-processing step to handle the negative effects of imbalanced data. In this post-processing phase, we performed threshold-moving to find an optimal threshold for our AML model.

Receiver operating characteristics (ROC) curve and precision-recall curve are two of the diagnostic tools that provide interpretation of the decision thresholds for binary (two-class) classification predictive modeling problems. In the following, we explain

ROC curves and precision-recall curves and their applications in detail.

4.3.1 Receiver Operating Characteristic (ROC) Curve

ROC is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for the threshold values in the range of (0.0, 1.0). The shape of the curve provides a lot of helpful information regarding false negative and false positive rates. For example, larger values on the y-axis of the plot show higher true positives and lower false negatives. In comparison, smaller values on the plot's x-axis indicate lower false positives and higher true negatives [169]. According to Figure 4.8, the shape of the ROC curve categorizes classifiers into three groups:

- Random: a random model is unable to make a distinction between the classes and would predict a random class or a constant class all the time. A diagonal line from the bottom left of the ROC curve to the top right depicts a random model at each threshold. Also, point (0.5, 0.5) on the ROC plot refers to the random classifier.

- Skillful: a skillful model assigns a higher probability to a randomly chosen real positive occurrence than a negative occurrence on average. The more skillful classifier is closer to the top left of the ROC plot.

- Perfect: a perfect model is represented at the point (0, 1) on the ROC curve. The curve shape of a perfect model is represented by a line that starts from the bottom left of the plot to the top left and then across the top to the top right.

4.3.2 Precision-Recall Curve

A precision-recall curve plots the precision (y-axis) and the recall (x-axis) for different thresholds. This plot provides a simultaneous review of precision and recall, which is useful in imbalanced data distribution. The main reason is that when the dataset

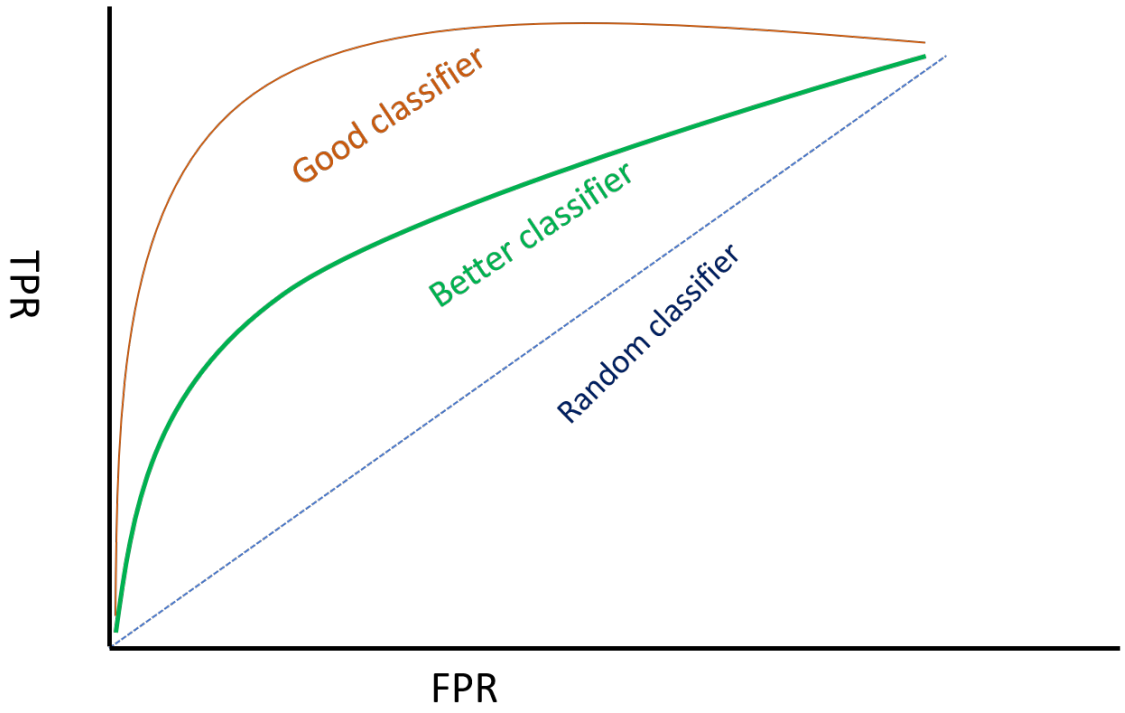

Figure 4.8: The ROC curve

contains a large number of negative samples, we are less interested in the classifier's skill at correctly identifying negative classes. Precision and recall are only concerned with the correct prediction of the minority samples, and they do not consider true negatives in their calculations [169].

Similar to ROC, the shape of the precision-recall curve provides valuable information. As shown in Figure 4.9, the shape of the precision-recall curve is flipped horizontally compared to the ROC curve. Hence, a perfect and skillful model will be in the same position. Only for a random model the line changes based on the distribution of the positive to negative classes. It is a horizontal line with the value of the ratio of positive cases in the dataset. For a balanced dataset, this is 0.5.

In this thesis, we utilized the precision-recall curve to perform threshold-moving and deal with our extremely imbalanced dataset. We consider several points in our selection of the precision-recall curve, which are listed below:

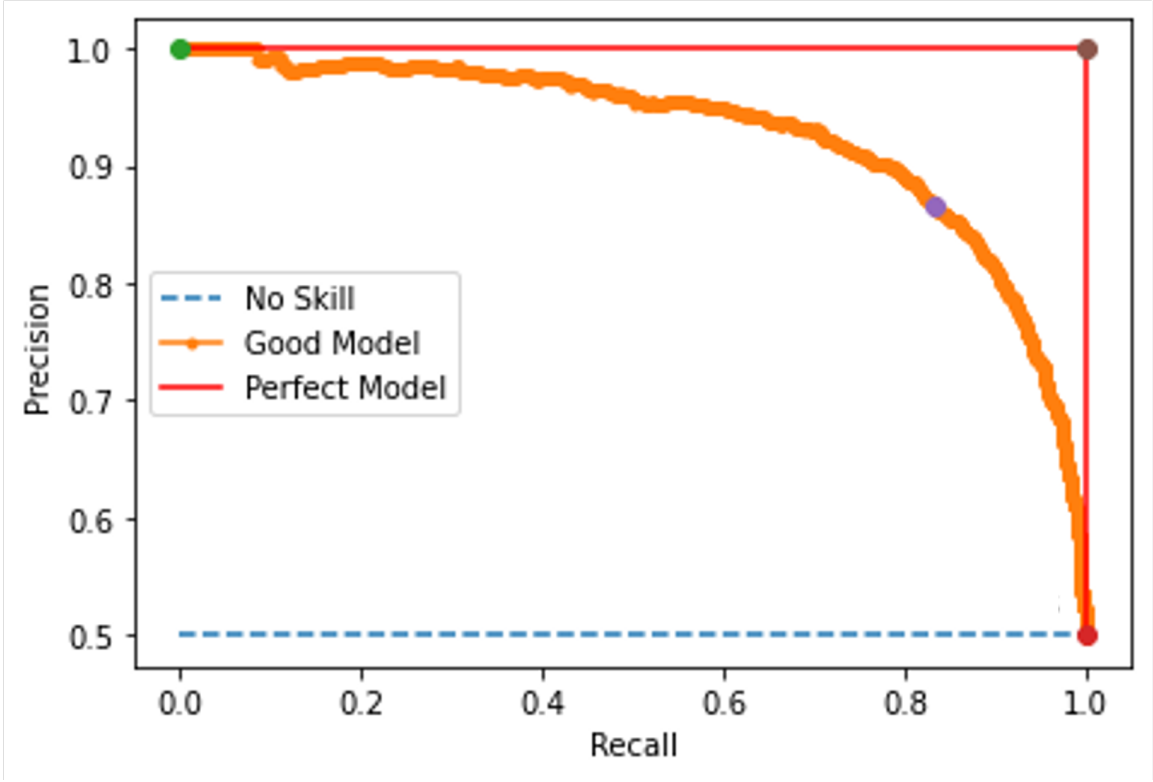

Figure 4.9: The precision-recall curve

- The ROC curve may present an overly optimistic picture of an algorithm's performance if there is a significant skew in the class distribution. The usage of true negatives in the false positive rate in the ROC curve and the deliberate avoidance of this rate in the precision-recall curve are the key factors contributing to this optimistic picture [170].

- The ROC curves will remain unchanged with any change in the ratio of the positive to the negative instance. However, metrics such as accuracy, precision, and F1 score use values from both columns of the confusion matrix. As a class distribution changes, these measures will also change, even if the fundamental classifier performance does not.

- The ROC curve with an imbalanced dataset might be deceptive and lead to incorrect interpretations of the model skill. On the other hand, the precision-recall curve can provide the viewer with an accurate prediction of future classification

performance because they evaluate the fraction of true positives among positive predictions [171].

Chapter 5

Experimental Results

This chapter presents a thorough examination of the dataset utilized in this thesis, along with the performance metrics employed for evaluation. Additionally, various experiments were designed to ascertain the efficacy of our proposed AML model, which we will explain in detail. Finally, we will engage in a comprehensive discussion concerning the results obtained from each experiment.

5.1 Dataset

One of the main challenges that AML researchers face is the lack of real-world datasets for testing and evaluating their solutions. This issue arises due to strict privacy and financial information regulations enforced by governments and regulatory bodies, particularly in democratic nations. To address this issue, researchers have turned to synthetic datasets, which are artificially generated datasets that mimic real-world data patterns and characteristics. Consequently, a significant portion of published research on AML technical solutions has been based on artificial or synthetic data sources [23, 112, 131, 132, 38, 53, 43, 19, 21, 98, 93, 91].

For this thesis, we have also leveraged a synthetic dataset called PaySim [140], which was specifically created for financial fraud detection research and is publicly available on Kaggle. The main reasons that we used PaySim in this thesis are:

- PaySim is publicly available, meaning that anyone can access and use it without

any restrictions. This makes it easier for researchers to access high-quality data without the need for costly and time-consuming data collection efforts. Moreover, the public nature of the dataset allows for greater transparency and reproducibility, ensuring that researchers can validate their findings using the same dataset.

- PaySim is a dataset that is artificially created to mimic real-world financial transactions. This is useful for researchers who need to study financial systems without access to real-world data or for those who want to study how fraud and financial crimes occur in a controlled environment.

- PaySim has gained popularity in academic research due to its realistic nature and the fact that it has been used in several studies [57, 172, 19, 43]. This makes it a valuable resource for researchers who want to build on existing work and make significant contributions to the field of financial security.

- PaySim is large enough for research purposes, providing researchers with a large amount of data to work with. This allows for in-depth analysis and the development of robust models, ensuring that algorithms developed using PaySim are accurate and effective.

PaySim consists of only 1,142 money laundering transactions against 1,047,433 genuine transactions. This means that the minority/majority ratio in this dataset is approximately 1/1000. Despite its synthetic nature, the PaySim dataset has gained widespread usage in AML research [57, 172, 19, 43]. Each transaction in the PaySim dataset is composed of several features and a classification label, as shown in Figure 5.1.

type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0

Figure 5.1: Features and label of a transaction in PaySim dataset

These features are as follows:

- Type of transaction. The dataset includes four transaction types: cash-in, cash-out, debit, and payment.
- Amount of transaction.
- Account number of the source account (nameOrig) who initiated the transaction.
- Balance of the source account before this transaction (oldbalanceOrig).
- Balance of the source account after this transaction (newbalanceOrig).
- Account number of the destination account (nameDest) who received the transaction.
- Balance of the destination account before this transaction (oldbalanceDest).
- Balance of the destination account after this transaction (newbalanceDest).
- A binary label (isFraud) that indicates whether the transaction was fraudulent (label 1) or not (label 0).

As Figure 5.1 shows, we have separate columns for the sender's account, old balance and new balance, as well the transaction amount. While at first observation, it appears that having all three columns is redundant, we should point out that the sender account's old amount subtracted from the transaction amount does not always equates to the sender's account's new amount. Moreover, for some records, the amount column does not correspond to the destination old or new amount, which could constitute a suspicious transaction. For example, for some records, the amount value is non-zero, while the old balance for the destination account is zero.

5.2 Prepare Train and Test Sets

To properly prepare train and test sets from the PaySim dataset, the class imbalance issue must be resolved to ensure that the machine learning model can effectively learn patterns. This requires a sufficient number of positive and negative samples in the training process.

When working with real-world datasets of banking transactions, it is common to split the data based on the transaction dates. For synthetic datasets without transaction dates, a common approach is to randomly divide the data into train and test sets, but this can lead to an uneven distribution of positive samples between the sets.

To address this challenge, we divide the dataset into train and test sets while maintaining the original proportion of positive and negative samples in each set. This involves ensuring a 1/1000 ratio of positive and negative samples in both sets. In this thesis, 60% of normal and money laundering transactions are randomly selected for training the model, 20% for validating the model and fine-tuning parameters, and the remaining 20% are reserved for testing purposes. N2V-GCN was trained using 60% of the transactions, and subsequently, we validated N2V-GCN by using 20% of the data to determine the most effective parameters. Finally, we employed the remaining 20% of the data to test the model.

5.2.1 Addressing the Class Imbalance Issue in Train Set

The PaySim dataset has an imbalanced class distribution, with a ratio of 1/1000 for the minority to the majority class. Training a machine learning model without addressing this class imbalance issue can result in a model that is unable to effectively learn the patterns of money laundering transactions. This is because there are not enough money laundering samples for the model to learn from. Therefore, it is important to find a solution to handle imbalanced data before training a machine learning

model.

In order to address this issue, we employed the SMOTE oversampling technique [173] to increase the number of money laundering transactions. This ensures that the machine learning model has an adequate number of samples from both the positive and negative classes to effectively learn their patterns. Using SMOTE, we generated ten different class distributions, including the natural distribution of PaySim: 1/1000, 1/500, 1/200, 1/100, 1/50, 1/20, 1/10, 1/2, 1/1. These changes were implemented only in the training set to assess the impact of various class distributions on the learning algorithm's performance. The test set remained unmodified, as imbalanced data is typical in real-world AML scenarios.

Following the preparation of the train and test sets, we proceeded to convert the training data into a graph structure where nodes represented customer accounts and edges denoted transactions between them. We used the NetworkX library in Python to perform this conversion, which facilitated the transformation of transaction data into a graph format.

5.3 N2V-GCN Hyperparameter Configurations

In this section, we will define the parameters in each of the two main components of our proposed AML model. These components include generating embedding vectors for each node in the graph using the node2vec approach and classifying transactions through the use of a graph convolution network.

It is important to note that the process of fine-tuning hyperparameters for machine learning algorithms is heavily reliant on the data being used. Therefore, if the N2V-GCN algorithm is to be used in different research areas or with other datasets, it is necessary to once again fine-tune the hyperparameters to ensure optimal performance. This highlights the importance of adapting the algorithm to the unique characteristics of each dataset to achieve the best possible results. It is crucial to remember that what may work well for one dataset may not necessarily be effective for another,

emphasizing the need for careful consideration and thorough experimentation when fine-tuning hyperparameters.

5.3.1 Node2vec Setup

To extract rich information from the graph, we applied the node2vec embedding vector approach. This method enables us to generate low-dimensional vector representations for each node in the graph that capture the underlying structural properties of the graph. By employing node2vec, we can capture the complex relationships and interactions between transactions in the graph and use this information to train our N2V-GCN model effectively.

There are five parameters that can be configured when using node2vec:

- P: It controls the likelihood of the random walk returning to the previous node in the walk.
- Q: It controls the likelihood of the random walk exploring new nodes.
- *Number of walks*: It is equal to the number of visited nodes from the starting node and defines the breadth-first search (BFS) concept.
- *Walk length*: It specifies the number of nodes in each walk or the depth of each walk and defines the depth-first search (DFS) concept.
- *Size of embedding vectors*: It specifies the size of the resulting embedding vectors, which are subsequently used by the graph convolution network.

A higher number of walks with a shorter walk length explores the local neighbourhoods of nodes more, while a higher number of walks with a longer walk length explores the global structure of the graph more. Node2vec generates a series of random walks by exchanging between BFS and DFS and then converting them to an embedding vector for each node.

To obtain the best possible results in our experiments, we conducted tests with various sets of parameters and ultimately selected the combination that yielded the most favourable outcomes. The grid search method was employed to determine the optimal set of hyperparameters for Node2vec, which we identified as follows:

- P: 1, selected from a range of [0.25, 0.5, 1, 1.25, 1.5, 2].
- Q: 2, selected from a range of [0.25, 0.5, 1, 1.25, 1.5, 2].
- *Number of walks*: 15, chosen from options of [5, 10, 15]. We explored values both smaller and larger than the default value of 10 in node2vec.
- *Walk length*: 32, choose from options of [16, 32, 64, 80]. We tested four different values, including the default value of 80 in node2vec.
- *Size of embedding vectors*: 128, chosen from a range of [64, 128, 256]. It is worth noting that 128 is also the default value in node2vec.

5.3.2 Graph Convolution Network Setup

We evaluated multiple parameter sets to determine the optimal combination for achieving the best performance in our experiments based on their design. To obtain the best combination of hyperparameters for the graph convolution network, we applied the grid search method using the validation set. The resulting hyperparameters, which gave the best performance, are listed below:

- *Network structure*: two convolutional layers that use 16 filters and have a kernel size of 3, which determines the number of neighbours to aggregate. The ReLU activation function is applied, and a fully connected sigmoid layer is used for the output.
- *Learning rate*: the model is trained using Adam optimizer [174] while the learning rate is set to 0.01.

- *Batch size*: 16.
- *Number of epochs*: 20.

5.4 Evaluation Metrics

The choice of evaluation performance metrics for machine learning algorithms is problem-dependent. As mentioned in section 3.2.4, the most important metrics for AML are false negative rate, false alarm rate, false positive rate, recall, and precision. Therefore, we determined the effectiveness of N2V-GCN and reported results using these metrics. Our goal is to minimize the number of false negatives to as close to zero as possible while keeping the occurrence of false positives and false alarms at the lowest possible. Although accuracy has been commonly used in AML research papers, it is not well-suited for evaluating AML performance due to the imbalanced nature of AML data.

For example, let's say we have a dataset containing financial transactions where the number of legitimate transactions far exceeds the number of fraudulent transactions, with a ratio of 1/1000. If we train an AML system on this imbalanced dataset, it may achieve a high accuracy of 99.99%, which suggests that it is performing very well. However, this high accuracy can be misleading and does not necessarily indicate that the AML system is effective in detecting fraudulent transactions. In fact, in this scenario, the AML system is missing all of the fraudulent transactions, which is a significant issue in the context of AML. This means that the system is failing to identify and flag potential money laundering activities, which could have severe consequences for banks and their customers, such as financial and reputational damages.

5.5 Experimental Results and Discussions

This section will provide a detailed explanation of the various experiments we conducted to assess the effectiveness of N2V-GCN. Our initial experiment aimed to iden-

tify the optimal threshold of N2V-GCN using the PaySim dataset. This was in line with our objective to minimize the FNR to zero while maintaining the FAR as low as possible. Subsequently, we analyzed the effectiveness of our model and examined the influence of different class distributions on its performance to ascertain the most suitable class distribution for N2V-GCN in relation to our dataset. Additionally, we conducted two more experiments to investigate the impact of the node2vec technique in general and varying values for each of its parameters. Finally, we compared the performance of N2V-GCN with commonly used machine learning algorithms in AML research.

5.5.1 Experiment #1: Finding the Optimal Classifier Threshold for N2V-GCN

In classification algorithms, a threshold is used to convert a model's predicted probability into a binary class label. The default threshold is often set to 0.5, but it may not be appropriate in some cases, such as in AML, where the cost of missing a true positive is much higher than the cost of incorrectly classifying a negative observation as positive. Adjusting the threshold can help achieve a better balance between false positives and false negatives and improve the overall performance of the model. However, the choice of the threshold depends on the specific performance metric, and in AML, the goal is to minimize false negatives while keeping false positives and false alarm rates low.

As mentioned in Section 5.2.1, we employed SMOTE to increase the fraudulent transactions and modify the natural class distribution of the PaySim dataset, which is 1/1000. After that, we plotted a precision-recall curve to improve our understanding of how N2V-GCN's precision and recall change at different thresholds. In the precision-recall curve, the x-axis represents the recall, and the y-axis represents the precision. The curve starts from the bottom-left corner, corresponding to a threshold of 1.0, and moves towards the top-right corner, corresponding to a threshold of 0.

Figure 5.2 shows the precision-recall curve for N2V-GCN when the class distribution of the PaySim dataset is 1/1.

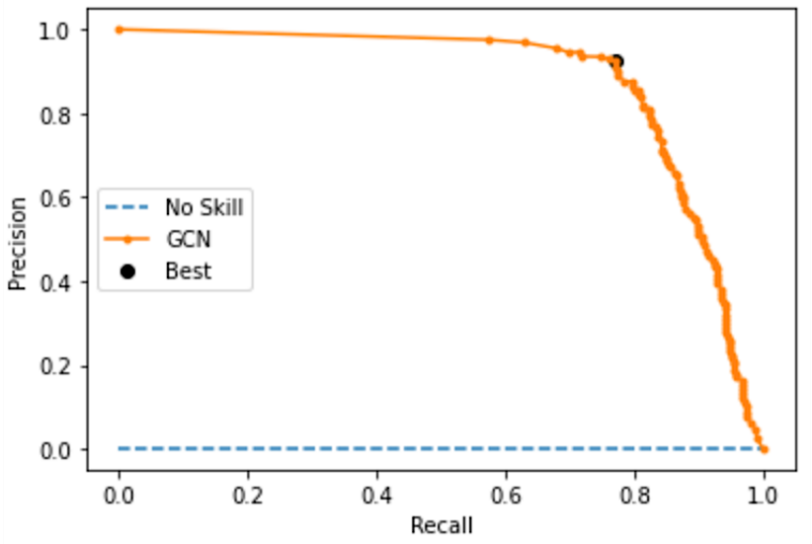

Figure 5.2: Precision-recall curve of N2V-GCN for the balanced dataset. The optimal threshold suggested by this curve is not the best threshold for AML.

Figure 5.2 suggests that the optimal result can be obtained at a threshold of 0.8, where there is a trade-off between recall and precision. However, in AML, our priority is to minimize the false negative rate, ideally to zero, while keeping the false alarm rate as low as possible. Therefore, the precision-recall curve is not an appropriate tool for evaluating AML performance. As a result, we employed a grid search method to determine the optimal threshold within the range of [0.3,0.6] that would minimize false negatives. After conducting the search, we identified 0.3 as the best threshold.

To better understand how adjusting the threshold affects our model's performance, we conducted an experiment where we examined all the thresholds between 0.30 and 0.40. The experiment was carried out using the PaySim dataset with a class distribution of 1/1, and we used the best parameters for N2V-GCN, which we explained in Sections 5.3.1 and 5.3.2. The results of this experiment are presented in Table 5.1. We found that from the 0.30 to the 0.32 threshold, the false negative rate remained constant. Therefore, we recommend choosing the 0.32 threshold as it produced lower

false positive and false alarm rates, making it the best threshold for N2V-GCN.

Table 5.1: Results of the experiment that investigated the impact of adjusting the threshold for a balanced dataset in N2V-GCN. The best threshold for N2V-GCN is highlighted in green. Thresholds that show a sudden increase in FNR are highlighted in blue.

Threshold	FNR	#FN	Recall	FPR	FAR	Precision
0.30	0	0	100	0.2434	69.10	30.89
0.31	0	0	100	0.2148	66.37	33.63
0.32	0	0	100	0.1909	63.70	36.30
0.33	0.87	2	99.12	0.1742	61.76	38.24
0.34	0.87	2	99.12	0.1623	60.07	39.93
0.35	1.31	3	98.68	0.1565	59.31	40.69
0.36	1.75	4	98.25	0.1432	57.25	42.75
0.37	1.75	4	98.25	0.1346	55.73	44.27
0.38	1.75	4	98.25	0.1293	54.75	45.25
0.39	1.75	4	98.25	0.1193	52.74	47.26
0.40	2.19	5	97.80	0.1121	51.31	48.69

In addition, it is evident from Table 5.1 that we have four peaks in threshold values (0.33, 0.35, 0.36, and 0.40) where FNR increases suddenly.

It can be seen that when the threshold was set at 0.32, our N2V-GCN model could detect all of the money laundering transactions. However, increasing the threshold to 0.33 resulted in the model missing two money laundering transactions. On the other hand, increasing the threshold to 0.40 has led to a significant increase in the number of false negatives, with the model missing five money laundering transactions.

5.5.2 Experiment #2: Evaluating the Impact of Different Class Distributions on the Performance of N2V-GCN

In this section, we will explore how different class distributions and imbalanced data affect N2V-GCN's performance in terms of FNR, FAR, FPR, recall, and precision. We conducted all of these experiments with the optimal parameter for N2V-GCN, as

previously mentioned in Section 5.3.1 and 5.3.2 and the best threshold of N2V-GCN which is 0.32. Table 5.2 shows the results of N2V-GCN's performance for each class distribution.

Table 5.2: Results of the experiment examining the effect of various class distributions on N2V-GCN. The best result is obtained by 1/1 class distribution, which is highlighted in green.

Class distributions	FNR	#FN	FPR	FAR	Precision	Recall
1/1000	11.40	26	0.025	20.78	79.22	88.60
1/500	8.77	20	0.0381	27.78	72.22	91.23
1/200	6.57	15	0.0458	31.07	68.93	93.42
1/100	4.82	11	0.0467	31.11	68.89	95.18
1/50	4.82	11	0.0480	31.76	68.24	95.18
1/20	3.50	8	0.0610	36.96	63.04	96.50
1/10	2.19	5	0.0840	44.39	55.61	97.80
1/5	1.75	4	0.1002	48.39	51.61	98.25
1/2	0.43	1	0.1847	63.03	36.97	99.56
1/1	0	0	0.1909	63.70	36.30	100

When comparing a dataset with a 1/1 balanced ratio to one with a natural distribution of 1/1000, it was found that the FNR increased by over 11.40 percentage points, which is considered too high for AML. This increase in FNR occurs as the class distribution becomes more imbalanced, causing the FNR to increase while the FPR and FAR decrease. The reason for this is that in a 1/1 ratio, the learning algorithm has sufficient positive samples to learn their patterns, but as the number of positive samples decreases in the training data, the model's ability to detect the pattern of money laundering transactions becomes weaker.

On the other hand, minimizing FNR means maximizing recall, as these two metrics are equivalent ($FNR = 1 - recall$). Hence, we are looking for a class distribution that has the lowest FNR (highest recall) with acceptable FPR, FAR, and precision. This is because false negatives are more problematic than false positives and false alarms

in AML. Our proposed AML model achieves a false negative rate of zero when trained with the balanced class distribution of the PaySim dataset. The changes in FNR and recall are demonstrated in Figures 5.3 and 5.4, respectively, as the dataset transitions from a balanced state (1/1) to a significantly imbalanced one (1/1000).

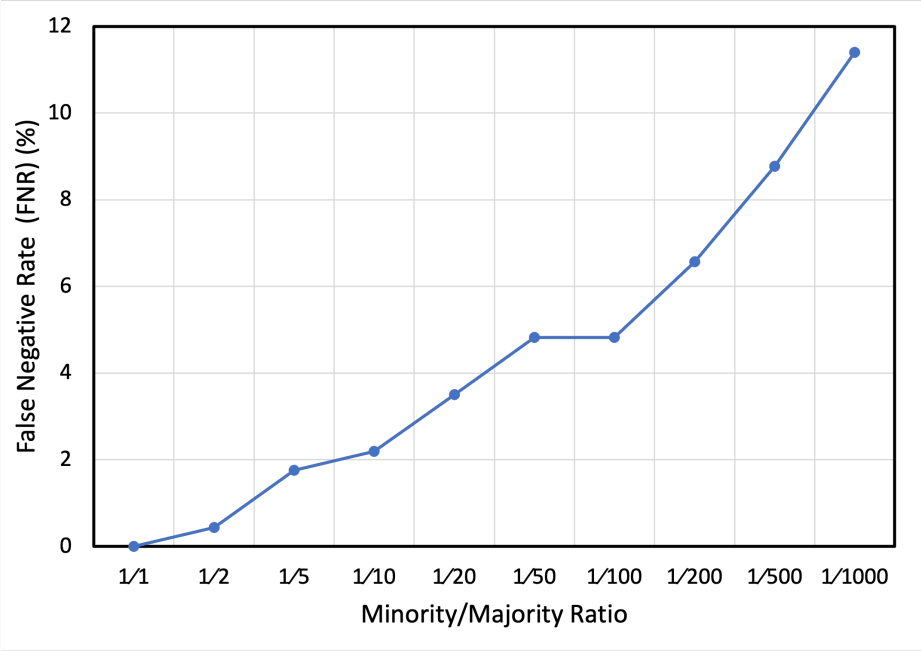

Figure 5.3: The variation in FNR with respect to different class distributions.

In terms of false alarm rate (FAR), the balanced dataset has a rate of 63.70%, which is considered reasonable when compared to real-world false alarm rates of approximately 90% (cite). While an imbalanced dataset results in lower FAR and FPR, it is not desirable to prioritize FPR and FAR at the expense of FNR, as missing money laundering transactions come with a much higher cost than having more false alarms and false positives. The trend of FAR and FPR as the dataset becomes more imbalanced is illustrated in Figure 5.5 and 5.6, respectively.

5.5.3 Experiment #3: Evaluating the Effectiveness of Node2vec Embedding Vectors

The aim of this set of experiments is to assess the effectiveness of node2vec vectors from various perspectives. The initial focus is on evaluating the internal parameter

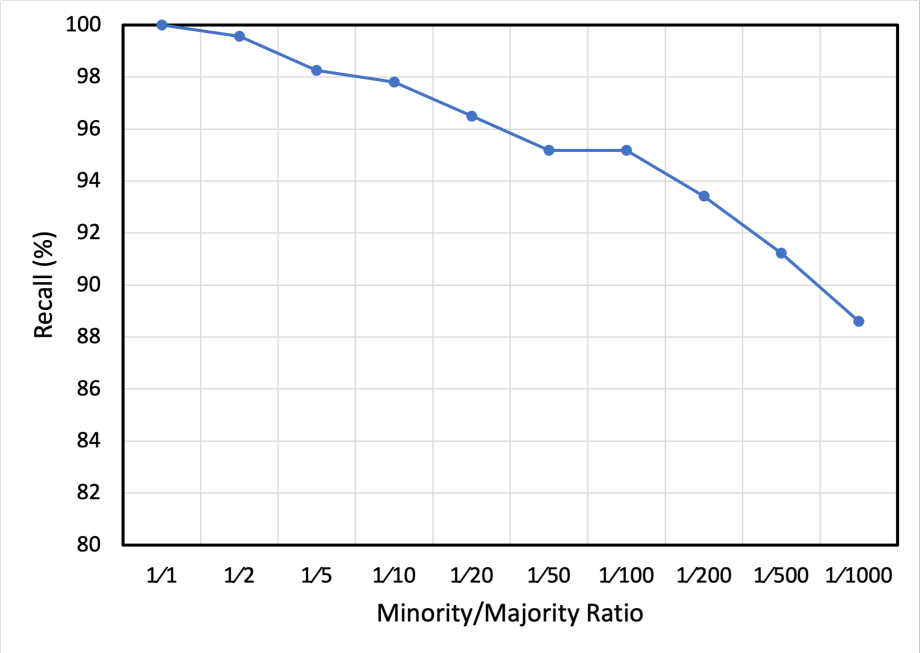

Figure 5.4: The variation in recall with respect to different class distributions.

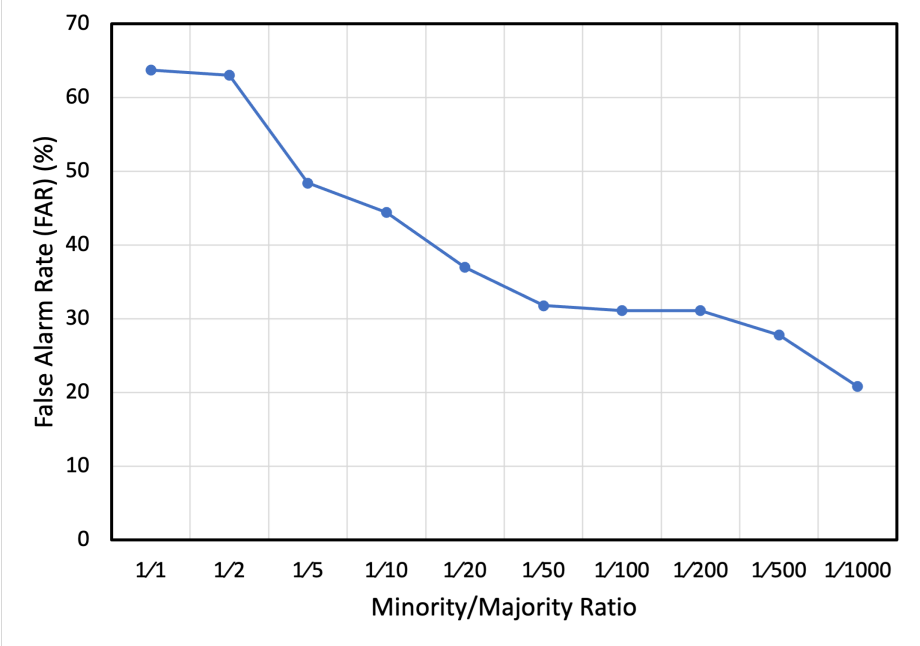

Figure 5.5: The variation in FAR with respect to different class distributions.

of node2vec. The two critical parameters that influence the properties of the random walks in node2vec are the number of walks and the walk length. The number of walks specifies the number of random walks that are generated from each node in the

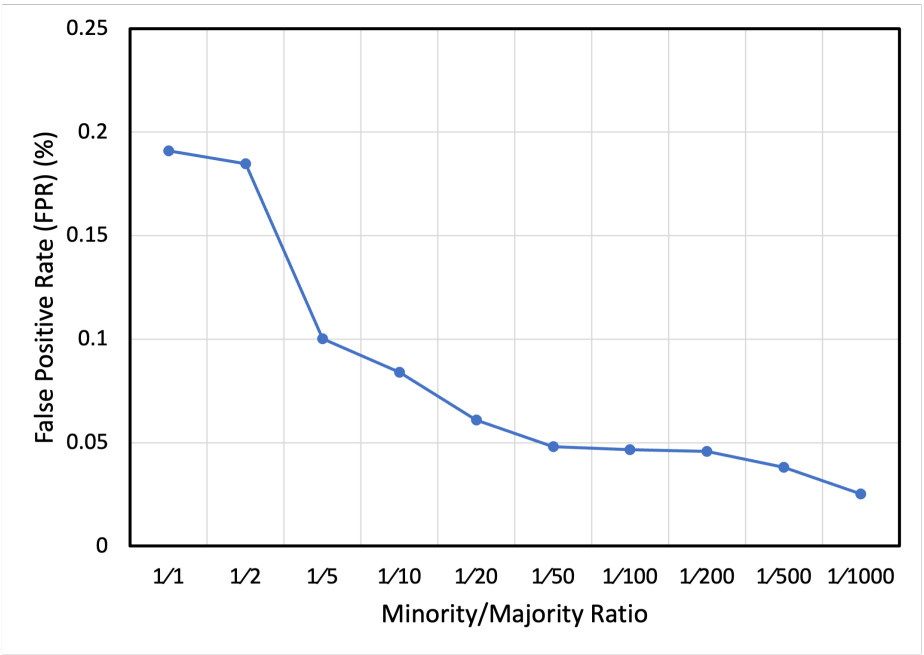

Figure 5.6: The variation in FPR with respect to different class distributions.

graph (implies the concept of breadth-first search, while the walk length determines the length of each random walk (implies the concept of depth-first search).

Experiment #3.1: Impact of the number of walks parameter. The objective of this experiment is to examine the effectiveness of varying the number of walks on the performance metrics (FNR, FAR, FPR, recall, and precision) of the model. It has been noted that the best performance of N2V-GCN is achieved with 15 walks in node2vec. In order to examine the impact of this parameter on the process of transaction identification, we conducted an experiment where we fixed all other parameters and varied the number of walks from 5 to 10. By doing so, we evaluated the effectiveness of this parameter and investigated how changes in the number of walks may affect our AML model's performance. In this experiment, we kept all the other parameters fixed and only changed the number of walks. The results of this experiment are provided in Table 5.3.

Table 5.3: Results of experiments that examined how the number of walk parameter in node2vec affects N2V-GCN

Number of Walks	FNR	FPR	FAR	Recall	Precision
5	1.31	0.3532	76.68	98.68	23.31
10	0.43	0.2959	73.20	99.56	26.80
15	0	0.1909	63.70	100	36.30

It can be seen that increasing the number of walks parameter in the N2V-GCN generally leads to improved performance. Specifically, as the number of walks increases from 5 to 10 and 15, the model achieves lower FNR and FPR, indicating a better classification of positive and negative instances, respectively. Notably, the FNR of the model with five walks decreases by 1.31 percentage points when the number of walks is increased to 15. Additionally, the model's recall metric improves while increasing the number of walks, indicating better identification of all positive instances. This improvement in the recall is consistent with the reduction in FNR, as they are equivalent metrics. More precisely, the recall increases gradually from 98.68 percentage points with five walks to 100 percentage points with 15 walks. Figure 5.7 shows the changing trend of FNR and FPR with increasing the number of walks in N2V-GCN.

Furthermore, the results also show that increasing the number of walks can lead to a lower FAR, which means that the method becomes less likely to incorrectly classify negative instances as positive. This is particularly noticeable in the results for the highest number of walks (15), where the FAR is lower than the 5 number of walks (63.70 vs. 76.68 percentage points). Figure 5.8 show the trend of FAR and recall while changing the number of walks in the node2vec.

Finally, the precision metric shows an interesting trend where it increases moderately from 5 to 10 walks (23.31 vs. 26.80 percentage points) and then increases dramatically from 10 to 15 walks (23.31 vs. 36.30 percentage points). This indicates that the method is better at correctly classifying positive instances when the number

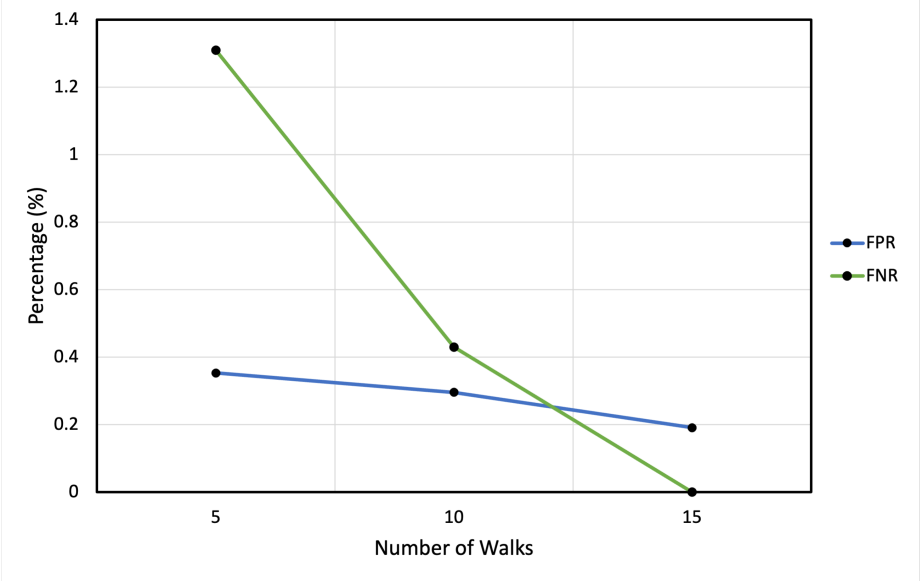

Figure 5.7: The variation in FNR and FPR with respect to a different number of walks in node2vec.

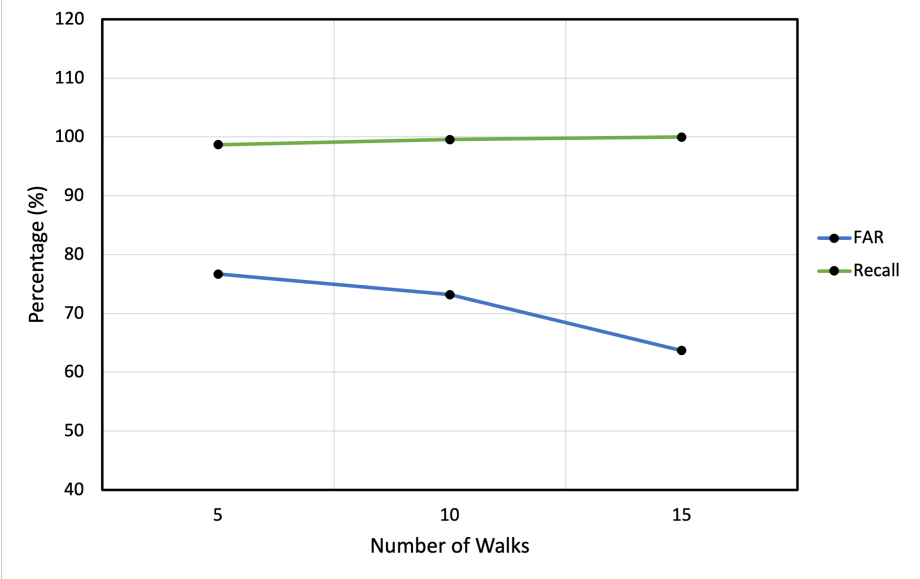

Figure 5.8: The variation in FAR and recall with respect to a different number of walks in node2vec.

of walks is increased.

Overall, these results suggest that choosing an appropriate number of walks is important for achieving good performance with the N2V-GCN model. In this particular experiment, the best results were obtained with a number of walks equal to 15.

Experiment #3.2: Impact of the walk length parameter. Similar to the number of walks, the N2V-GCN model's ability to detect money laundering cases is influenced by the length of the walk. In order to examine the impact of this parameter on the process of money laundering detection, we conducted an experiment where we fixed all other parameters and varied the walk length from 16 to 64. By doing so, we evaluated the effectiveness of this parameter and investigated how changes in the walk length may affect our AML model's performance.

Table 5.4 summarizes the results of N2V-GCN using various walk lengths.

Table 5.4: Results of experiments that examined how the walk length parameter in node2vec affects N2V-GCN

Walk length	**FNR**	**FPR**	**FAR**	**Recall**	**Precision**
16	0.43	0.1002	48.05	99.56	51.95
32	0	0.1909	63.70	100	36.30
64	1.75	0.7423	87.40	98.25	12.59

Based on the results, longer walk lengths tend to yield better performance, but only up to a certain point. A comparison between walk lengths 16 and 32 shows a minor but important reduction in FNR (0.43 vs. 0 percentage points). Although this change is not substantial, it is considered significant in the context of AML because FNR is critical. The 64 walk length results in a slight decrease in all performance metrics and an increase in FNR, which may be due to an overfitting issue. Figure 5.9 shows the variations in FNR and FPR while increasing the walk length. Also, Figure 5.10 provides an insight into the changes of FAR and recall by changing the walk length in node2vec.

5.5.4 Experiment #4: Effectiveness of Node2vec Approach

This section involves comparing the outcomes of our AML model with and without node2vec. To accomplish this, we used the transaction graph produced from the

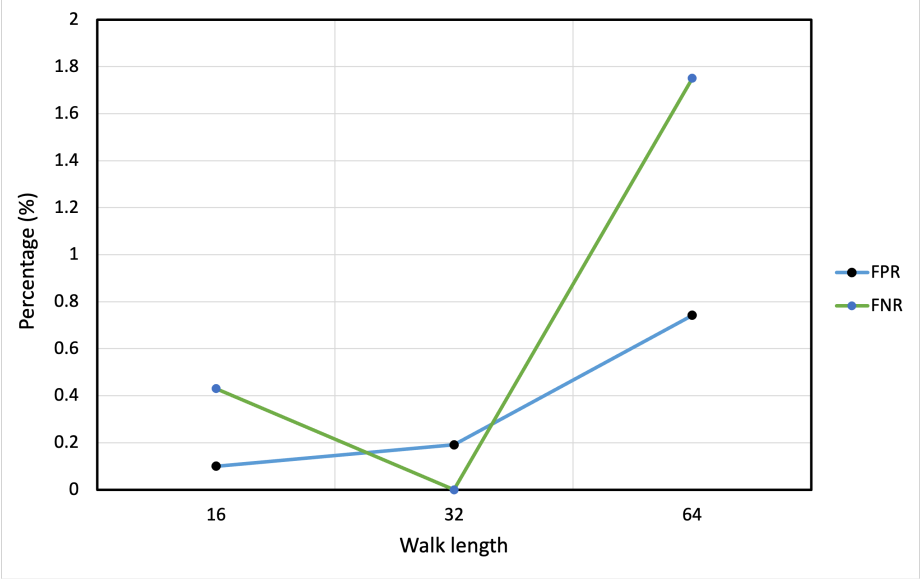

Figure 5.9: The variation in FNR and FPR with respect to different walk lengths in node2vec.

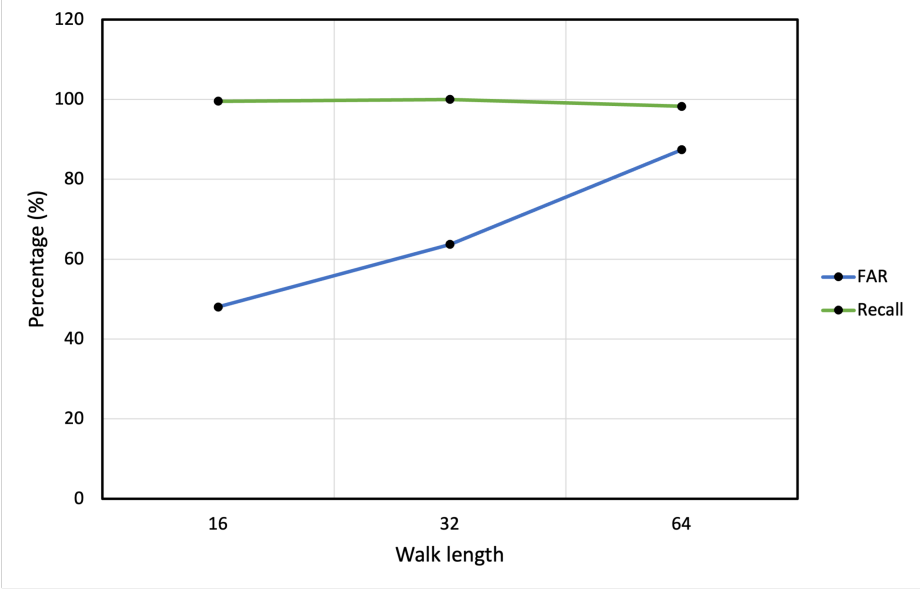

Figure 5.10: The variation in FAR and recall with respect to different walk lengths in node2vec.

PaySim dataset, with a balanced class distribution (1/1) for training, as the input to the graph convolution network, using the same parameters mentioned in Section 5.3.2. Additionally, we maintained the best threshold of 0.37 throughout this experiment. The findings of this experiment are presented in Table 5.5.

Table 5.5: The outcomes of comparing our AML model when utilizing the node2vec technique versus when it is not used.

	FNR	**FPR**	**FAR**	**Recall**	**Precision**
N2V-GCN	0	0.1909	63.70	100	36.30
GCN	3.49	0.29	73.54	96.50	26.46

Upon analyzing the results, it is evident that the N2V-GCN model surpasses the GCN model in all the evaluation metrics. The N2V-GCN model exhibits a considerable decrease in FNR by 3.49 percentage points (0 vs. 3.49), which indicates its superior capability in identifying positive samples, a critical aspect in the context of AML. Furthermore, the N2V-GCN model demonstrates a nearly 10% reduction in false alarm rate as compared to the GCN model. This leads to a decrease in costs associated with the human investigation of false alarms produced by the AML model. Moreover, the N2V-GCN model has considerably higher precision, implying that the positive predictions made by the model are more likely to be accurate. Overall, these findings indicate that utilizing node2vec alongside a graph convolutional network can significantly enhance the performance of classification models for this specific application.

5.5.5 Experiment #5: Comparing N2V-GCN with Other Classification Algorithms

In this experiment, we compared the performance of N2V-GCN with other commonly used classification algorithms in AML research. We used the optimal parameters generated by node2vec to create embedding vectors, as mentioned earlier (Section 5.3.1). For the graph convolution network, we kept all the parameters at their optimal values (as described in section 5.3.2), with a threshold of 0.32 using the balanced class distribution (1/1) of the PaySim dataset. For the other algorithms, we used their default values and only changed the classifier threshold to the optimal ones, which

are 0.35, 0.40, and 0.41 for SVM, random forest, and logistic regression, respectively. The results of the experiment can be found in Table 5.6.

Table 5.6: Comparison results of N2V-GCN with other classification algorithms.

	FNR	**FPR**	**FAR**	**Recall**	**Precision**
N2V-GCN	0	0.1909	63.70	100	36.30
Random forest	7.87	0.46	82.37	92.13	17.63
Logistic regression	4.37	30.62	99.66	95.63	0.33
SVM	10.49	0.33	77.21	89.5	22.79

Table 5.6 reveals that the N2V-GCN model outperforms the other models since it has the lowest FNR (0), FPR (0.1909%), and FAR (63.70%), and the highest recall (100%). On the other hand, the SVM model has a low FPR (0.33%) but a high FNR (10.49%), implying that it is not as proficient in detecting positive instances. The random forest model has a lower recall (92.13%) and higher FPR (0.46%) compared to N2V-GCN, but it has a lower FNR (7.87%) than SVM. Lastly, the logistic regression model has high FPR (30.62%) and FAR (99.66%), suggesting that it generates numerous false alarms. Although its recall (95.63%) is quite high, it is not sufficient to compensate for its inadequate FPR and FAR.

5.6 Discussion of Existing AML Solutions

In this section, we give a brief overview of different AML approaches that have used the PaySim dataset, which is the same dataset used in our thesis. Although many AML research papers use synthetic datasets, only a few mention the actual name of the dataset they use. We briefly describe the studies that employ PaySim to develop their AML models.

- **Raiter et al.[19]:** They evaluated four different machine learning models - random forest, SVM, artificial neural network, and logistic regression - to deter-

mine their effectiveness in identifying potentially fraudulent money laundering transactions. They found that the random forest model outperformed the other techniques, while the artificial neural network was the least accurate model.

- **Tundis et al.[43]:** They developed a money laundering detection model based on transaction features. They used the PaySim dataset to define a new set of nine features for each transaction, including balance difference, incoming domestic amount, outgoing domestic amount, incoming foreign amount, outgoing foreign amount, incoming domestic count, outgoing domestic count, incoming foreign count, and outgoing foreign count, in order to improve the performance of their classifier. They tested their model using five different classifiers: random forest, decision tree, SVM, linear regression, and Naive Bayes. According to their findings, the random forest classifier provided the best results compared to the other classifiers.

- **Pambudi et al.[175]:** They employed a technique called random under-sampling to overcome the problem of the imbalanced dataset in fraud detection. Additionally, they optimized the performance of the SVM by adjusting its kernels and hyperparameters.

- **Kumar et al.[38]:** They applied big data analytics to analyze transactions. To this end, they utilized logical operators and if-else conditions to analyze the connections between different attributes in the dataset. Ultimately, they employed the Naive Bayes classifier to detect instances of money laundering.

The outcomes of these models, as reported by their authors, are summarized in Table 5.7.

Table 5.7: Results of the comparison between our AML model and other related existing works

Model	**Accuracy**	**F1 score**	**FPR**	**FNR**
Raiter et al.[19]	99.00	36.00	-	-
Tundis et al.[43]	95.44	95.89	6.70	2.70
Pambudi et al.[175]	88.00	90.00	-	-
Kumar et al.[38]	81.25	-	-	-
N2V-GCN	-	-	0.1909	0

The results indicate that among the studies using the PaySim dataset, Tundis et al. [43] is the only one that reported both the FNR and FPR, which are crucial metrics for AML. However, their FNR is 2.41 percentage points higher than that of N2V-GCN, and their FPR is 6.10 higher than our model. This suggests that N2V-GCN's results are closer to the AML objective of minimizing FNR to zero while keeping FPR and FAR as low as possible.

The other studies mainly focused on accuracy, which is not a reliable metric for AML models due to the imbalanced nature of the PaySim dataset. Additionally, Tundis et al. [43], and Pambudi et al. [175] showed almost balanced recall and precision, which is not a good sign for AML. The goal is to minimize FNR to zero, or in other words, maximize recall because these two metrics are equivalent. Hence, their results cannot be used to compare with our model. Although Pambudi et al. [175] addressed the imbalanced data issue by using undersampling, their results are incomplete, and it is difficult to assess how well their model detects positive samples.

Chapter 6

Conclusion and Future Work

6.1 Summary

Money laundering, which involves turning illegal funds into legitimate ones, is the third largest industry worldwide and can impact many individuals, industries, and companies. As a result, financial institutions such as banks must establish regulations to fight against it. Anti-money laundering refers to a set of policies and rules aimed at preventing criminals from disguising their illicit funds as legitimate income. Governments and their relevant organizations, like FINTRAC in Canada and FinCen in the US, are responsible for implementing these AML compliance laws.

Banks currently use rule-based techniques to detect money laundering. However, these methods have several major flaws that render them inadequate for accurately identifying money laundering. These limitations include but are not limited to a lack of generalizability, an inability to detect sophisticated criminal behaviour, and a failure to recognize unseen patterns. As a result, machine learning models have emerged as an alternative to traditional rule-based methods. The key distinction between machine learning and rule-based approaches is the ability of the former to learn from data and identify previously unseen money laundering activities within transactions. These and other machine learning capabilities broadly address the shortcomings of rule-based models.

Deep learning is a subfield of machine learning that utilizes complex algorithms

based on neural networks to handle unstructured data and recognize more intricate patterns in the data. This study employs a robust deep learning model called graph convolution network (GCN) to categorize transactions as usual or money laundering. Additionally, we utilized the node2vec graph embedding method by converting raw transactions into a graph. The node2vec technique generates rich feature vectors for each node in the graph representing customer accounts. Hence, our anti-money laundering (AML) model, named N2V-GCN, involves the following three steps:

- Transform transaction data into a graph, with each node and edge representing customer accounts and their transactions.
- Use node2vec to create embedding vectors for each node in the graph, which the learning algorithm utilizes as feature vectors. The node2vec approach has two parameters: the number and length of walks. We report N2V-GCN results using three combinations of number and length of walks to assess their impact on performance.
- Apply GCN to the feature vectors generated by node2vec and the transaction graph to classify transactions.

Data is essential to any machine learning algorithm, so providing high-quality data for training the model is crucial. This thesis utilized the PaySim dataset, a publicly available dataset for fraud detection, including money laundering. The PaySim dataset is highly imbalanced, with a ratio of 1/1000 (positive or money laundering class to negative or regular class). Training a machine learning model with such an imbalanced dataset can result in inaccurate results with high accuracy due to the algorithm's bias towards the negative class. In this scenario, the detection model would correctly identify nearly all normal transactions, leading to high accuracy but with a low rate of false positives and false negatives. Thus, we focused mainly on the false positive and false negative rates in our results due to their significance in the anti-money laundering field.

This thesis employed a hybrid model to deal with the class imbalance problem. Our hybrid model has two parts: sampling and thresholding. In sampling, we applied SMOTE, an over-sampling technique, to modify the class distribution of the dataset. Our main goal is to evaluate the effectiveness of different class distributions on the decision of the learning algorithm. In thresholding, we utilized the precision-recall curve to find the optimal threshold for our N2V-GCN. After modifying the class distribution of PaySim, we trained N2V-GCN using ten different class distributions, including the original one. We reported results for each class distribution. Our results indicate that the optimal class distribution for our data is the balanced one (1/1).

Our main contributions are:

- conducting a comprehensive survey analyzing more than 50 AML research papers that employed machine learning models. The survey focused on three main aspects: datasets used, metrics employed for evaluating learning algorithms, and approaches used to deal with class imbalance. By reviewing these aspects, the survey aimed to provide a comprehensive understanding of the current machine learning-based models for AML purposes, identify strengths and weaknesses of different datasets, and guide the development of more effective and efficient AML systems to detect and prevent financial crimes.

- developing an AML model called N2V-GCN to effectively detect money laundering transactions by combining node2vec graph embedding technique and graph convolution network. It is designed to process graph-based data structures to capture the underlying relationships between transactions, enabling it to identify patterns indicative of money laundering. The model has shown promising results, with a false negative rate of zero in our experiments and a low false alarm rate of 63.70% compared to the industry rate of 90% to 95%. Also, it has the potential to be used in other financial industry applications, such as fraud detection.

- employing a data-level approach using an oversampling technique called SMOTE to handle the class imbalance issue of the PaySim dataset. We applied this technique to modify the class distribution of the PaySim dataset, with the aim of determining its effectiveness and identifying the optimal class distribution for our model.

The limitations of our proposed AML model, N2V-GCN, are as follows:

- Any machine learning model is data-dependent, and our model is no exception. This means that the performance of the machine learning model heavily relies on the quality and quantity of the training data used to build the model. In this case, the N2V-GCN model is designed and optimized based on the PaySim dataset. If this model were to be applied to a different dataset, it would be necessary to optimize its hyperparameters according to that dataset.

- PaySim is an old dataset of financial transactions which may not include the updated patterns of money laundering transactions. The PaySim dataset was released in 2018 and may not reflect the current landscape of money laundering activities. Criminals are constantly changing their methods to avoid detection; hence, it's crucial to use an up-to-date dataset of financial transactions to ensure that the model can detect new patterns of money laundering.

- PaySim only contains mobile money transactions. Mobile money transactions are only one of the several banking services that may contain money laundering activities. Training the model on different types of banking transactions enables the model to learn various patterns of money laundering.

- PaySim does not include the date of each transaction. The date of the transaction is an important feature in financial transactions, as money laundering activities occur over time. Using dates as a feature of transactions can help us build a dynamic graph and detect money laundering patterns over time.

6.2 Research Directions

The following are several potential research directions for future work:

- Provide an explanation for the decisions made by the learning algorithm. While N2V-GCN achieves a very low false negative rate, it can be difficult to understand how it arrives at its predictions. This lack of transparency may make it challenging for end-users to trust and adopt the model. One way to address this challenge is to provide explanations for the predictions of N2V-GCN. Explanations can help increase the transparency of the model by revealing how it makes decisions. With explanations, end-users can understand the reasoning behind the predictions, identify potential biases, and gain insights into the underlying data patterns. There are different methods to provide explanations for machine learning models, and some may be more suitable for N2V-GCN than others. For example, one approach could be to use local interpretability techniques such as LIME (Local Interpretable Model-Agnostic Explanations) or SHAP (SHapley Additive exPlanations) to identify the most important features that contribute to the model's predictions for a particular instance. Another approach could be to use global interpretability techniques such as partial dependence plots or feature importance rankings to provide an overview of how the model behaves across the entire dataset.

- Apply a one-class graph neural network to perform outlier detection for imbalanced transaction datasets. A one-class graph neural network is a type of neural network that is trained on only one class of data, in this case, normal transactions. The model learns the patterns and characteristics of the regular transactions, allowing it to identify anomalies or outliers, i.e., money laundering transactions. Therefore, the model is trained using unsupervised learning, which does not require labeled data, making it quick and economical to adopt new datasets for training.

- Use other datasets of financial transactions containing dates of transactions to validate and evaluate the proposed model. Dates of transactions feature in financial transaction datasets are valuable resources when training machine learning algorithms. It enables the algorithm to identify trends and patterns that can inform business decisions and predict future trends. By analyzing dates, the algorithm can identify seasonality, create subsets of the data, and detect outliers or anomalies. Additionally, by combining multiple datasets that contain date features, the algorithm can gain a more comprehensive understanding of the data and make more accurate predictions. Overall, the date of transaction feature is a crucial aspect of financial transaction datasets that can improve the performance of machine learning algorithms and provide valuable insights into AML classification.

- Use datasets that contain more customer-related data to validate and evaluate the proposed model. This data can provide valuable insights into customer behaviour, preferences, and potential risks. Incorporating geographical location, risk score, businesses associated with the customer, services used, and devices used for transactions can help identify patterns specific to a region, high-risk individuals or businesses, potential conflicts of interest, areas for additional services or improvements, and potential security risks. By using this additional data, machine learning algorithms can gain a deeper understanding of customer behaviour, enabling financial institutions to make more informed decisions, provide better services, and mitigate risks more effectively.

Bibliography

[1] Thomas Pietschmann and John Walker. *Estimating illicit financial flows resulting from drug trafficking and other transnational organized crimes*. UNODC, United Nations Office of Drugs and Crime, 2011.

[2] *Official website of United Nations Office on Drugs and Crime [Online]*. URL: https://www.unodc.org/unodc/en/money-laundering/overview.html (visited on 09/19/2022).

[3] Michael Levi and Peter Reuter. "Money laundering". In: *Crime and justice* 34.1 (2006), pp. 289–375.

[4] Friedrich Schneider and Ursula Windischbauer. "Money laundering: some facts". In: *European Journal of Law and Economics* 26.3 (2008), pp. 387–404.

[5] Joras Ferwerda. "The economics of crime and money laundering: does anti-money laundering policy reduce crime?" In: *Review of Law & Economics* 5.2 (2009), pp. 903–929.

[6] Matthew R Hall. "An emerging duty to report criminal conduct: banks, money laundering, and the suspicious activity report". In: *Ky. LJ* 84 (1995), p. 643.

[7] Dennis Cox. *Handbook of anti-money laundering*. John Wiley & Sons, 2014.

[8] Wouter H Muller, Christian H Kalin, and John G Goldsworth. *Anti-Money Laundering: international law and practice*. John Wiley & Sons, 2007.

[9] Doug Hopton. *Money laundering: a concise guide for all business*. Routledge, 2020.

[10] Emmanuel Senanu Mekpor, Anthony Aboagye, and Jonathan Welbeck. "The determinants of anti-money laundering compliance among the Financial Action Task Force (FATF) member states". In: *Journal of Financial Regulation and Compliance* (2018).

[11] Stuart Ross and Michelle Hannan. "Money laundering regulation and risk-based decision-making". In: *Journal of Money Laundering Control* (2007).

[12] Kamlesh D Rohit and Dharmesh B Patel. "Review on detection of suspicious transaction in anti-money laundering using data mining framework". In: *International Journal for Innovative Research in Science & Technology* 1.8 (2015), pp. 129–133.

[13] Ahmad Salehi, Mehdi Ghazanfari, and Mohammed Fathian. "Data mining techniques for anti money laundering". In: *International Journal of Applied Engineering Research* 12.20 (2017), pp. 10084–10094.

[14] Shijia Gao et al. "Intelligent anti-money laundering system". In: *2006 IEEE International Conference on Service Operations and Logistics, and Informatics*. IEEE. 2006, pp. 851–856.

[15] Martin Jullum et al. "Detecting money laundering transactions with machine learning". In: *Journal of Money Laundering Control* 23.1 (2020), pp. 173–186.

[16] Yan Zhang and Peter Trubey. "Machine learning and sampling scheme: An empirical study of money laundering detection". In: *Computational Economics* 54.3 (2019), pp. 1043–1063.

[17] Chih-Hua Tai and Tai-Jung Kan. "Identifying money laundering accounts". In: *2019 International Conference on System Science and Engineering (ICSSE)*. IEEE. 2019, pp. 379–382.

[18] Ana Isabel Canhoto. "Leveraging machine learning in the global fight against money laundering and terrorism financing: An affordances perspective". In: *Journal of business research* 131 (2021), pp. 441–452.

[19] Omri Raiter. "Applying Supervised Machine Learning Algorithms for Fraud Detection in Anti-Money Laundering". In: *Journal of Modern Issues in Business Research* 1.1 (2021), pp. 14–26.

[20] Jenny Domashova and Natalia Mikhailina. "Usage of machine learning methods for early detection of money laundering schemes". In: *Procedia Computer Science* 190 (2021), pp. 184–192.

[21] Mark E Lokanan. "Predicting Money Laundering Using Machine Learning and Artificial Neural Networks Algorithms in Banks". In: *Journal of Applied Security Research* (2022), pp. 1–25.

[22] Habiba Nasir Mohammed et al. "Machine Learning Approach to Anti-Money Laundering: A Review". In: *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*. IEEE. 2022, pp. 1–5.

[23] Edgar Alonso Lopez-Rojas and Stefan Axelsson. "Money laundering detection using synthetic data". In: *Annual workshop of the Swedish Artificial Intelligence Society (SAIS)*. Linköping University Electronic Press, Linköpings universitet. 2012.

[24] Bart Van Liebergen et al. "Machine learning: a revolution in risk management and compliance?" In: *Journal of Financial Transformation* 45 (2017), pp. 60–67.

[25] Jingguang Han et al. "Artificial intelligence for anti-money laundering: a review and extension". In: *Digital Finance* 2.3 (2020), pp. 211–239.

[26] Zhiyuan Chen et al. “Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review”. In: *Knowledge and Information Systems* 57.2 (2018), pp. 245–285.

[27] Colin King. “Anti-money laundering: An overview”. In: *The Palgrave Handbook of Criminal and Terrorism Financing Law* (2018), pp. 15–31.

[28] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. “A systematic review on imbalanced data challenges in machine learning: Applications and solutions”. In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–36.

[29] Nevine Makram Labib, Mohammed Abo Rizka, and Amr Ehab Muhammed Shokry. “Survey of machine learning approaches of anti-money laundering techniques to counter terrorism finance”. In: *Internet of Things—Applications and Future*. Springer, 2020, pp. 73–87.

[30] Alhanouf Abdulrahman Saleh Alsuwailem and Abdul Khader Jilani Saudagar. “Anti-money laundering systems: A systematic literature review”. In: *Journal of Money Laundering Control* 23.4 (2020), pp. 833–848.

[31] Clifton Phua et al. “A comprehensive survey of data mining-based fraud detection research”. In: *arXiv preprint arXiv:1009.6119* (2010).

[32] Ted E Senator et al. “Financial crimes enforcement network AI system (FAIS) identifying potential money laundering from reports of large cash transactions”. In: *AI magazine* 16.4 (1995), pp. 21–21.

[33] Mark Pieth and Gemma Aiolfi. “The private sector become active: The Wolfsberg process”. In: *Journal of Financial Crime* (2003).

[34] Jun Tang and Jian Yin. “Developing an intelligent data discriminating system of anti-money laundering based on SVM”. In: *2005 International conference on machine learning and cybernetics*. Vol. 6. IEEE. 2005, pp. 3453–3457.

[35] Tom M Mitchell and Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[36] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. “Handling imbalanced datasets: A review”. In: *GESTS international transactions on computer science and engineering* 30.1 (2006), pp. 25–36.

[37] Irina Rish et al. “An empirical study of the naive Bayes classifier”. In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. 2001, pp. 41–46.

[38] Ashwini Kumar, Sanjoy Das, and Vishu Tyagi. “Anti money laundering detection using Naıve Bayes classifier”. In: *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*. IEEE. 2020, pp. 568–572.

[39] Saleha Raza and Sajjad Haider. “Suspicious activity reporting using dynamic bayesian networks”. In: *Procedia Computer Science* 3 (2011), pp. 987–991.

[40] Nida S Khan et al. "A Bayesian approach for suspicious financial activity reporting". In: *International Journal of Computers and Applications* 35.4 (2013), pp. 181–187.

[41] Xinwei Deng et al. "Active learning through sequential design, with applications to detection of money laundering". In: *Journal of the American Statistical Association* 104.487 (2009), pp. 969–981.

[42] Mohannad Alkhalili, Mahmoud H Qutqut, and Fadi Almasalha. "Investigation of applying machine learning for watch-list filtering in anti-money laundering". In: *IEEE Access* 9 (2021), pp. 18481–18496.

[43] Andrea Tundis, Soujanya Nemalikanti, and Max Mühlhäuser. "Fighting organized crime by automatically detecting money laundering-related financial transactions". In: *The 16th International Conference on Availability, Reliability and Security*. 2021, pp. 1–10.

[44] S Rasoul Safavian and David Landgrebe. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.

[45] Micheline Kamber et al. "Generalization and decision tree induction: efficient classification in data mining". In: *Proceedings Seventh International Workshop on Research Issues in Data Engineering. High Performance Database Management for Large-Scale Applications*. IEEE. 1997, pp. 111–120.

[46] Vikas Jayasree and RV Siva Balan. "Money laundering regulatory risk evaluation using bitmap index-based decision tree". In: *Journal of the Association of Arab Universities for Basic and Applied Sciences* 23 (2017), pp. 96–102.

[47] Xuan Liu, Pengzhu Zhang, and Dajun Zeng. "Sequence matching for suspicious activity detection in anti-money laundering". In: *International conference on intelligence and security informatics*. Springer. 2008, pp. 50–61.

[48] Su-Nan Wang and Jian-Gang Yang. "A money laundering risk evaluation method based on decision tree". In: *2007 International conference on machine learning and cybernetics*. Vol. 1. IEEE. 2007, pp. 283–286.

[49] Ting-Hsuan Chen. "Do you know your customer? Bank risk assessment based on machine learning". In: *Applied Soft Computing* 86 (2020), p. 105779.

[50] Krzysztof Michalak and Jerzy Korczak. "Graph mining approach to suspicious transaction detection". In: *2011 Federated conference on computer science and information systems (FedCSIS)*. IEEE. 2011, pp. 69–75.

[51] Rafal Drezewski, Jan Sepielak, and Wojciech Filipkowski. "System supporting money laundering detection." In: *Digit. Investig.* 9.1 (2012), pp. 8–21.

[52] Daniel Borrajo, Manuela Veloso, and Sameena Shah. "Simulating and classifying behavior in adversarial environments based on action-state traces: An application to money laundering". In: *Proceedings of the First ACM International Conference on AI in Finance*. 2020, pp. 1–8.

[53] Arman Zand, James Orwell, and Eckhard Pfluegel. "A secure framework for anti-money-laundering using machine learning and secret sharing". In: *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE. 2020, pp. 1–7.

[54] Marti A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[55] Liu Keyan and Yu Tingting. "An improved support-vector network model for anti-money laundering". In: *2011 Fifth International Conference on Management of e-Commerce and e-Government*. IEEE. 2011, pp. 193–196.

[56] David Savage et al. "Detection of money laundering groups using supervised learning in networks". In: *arXiv preprint arXiv:1608.00708* (2016).

[57] Huyen Vu. "Machine learning in money laundering detection". In: *Chair's Message* (), p. 59.

[58] Tang Jun. "A peer dataset comparison outlier detection model applied to financial surveillance". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 4. IEEE. 2006, pp. 900–903.

[59] Miguel Agustın Villalobos and Eliud Silva. "A statistical and machine learning model to detect money laundering: an application". In: *Actuarial Sci. Dept. Anahuac Univ., Tech. Rep* (2017).

[60] Lin-Tao Lv, Na Ji, and Jiu-Long Zhang. "A RBF neural network model for anti-money laundering". In: *2008 International conference on wavelet analysis and pattern recognition*. Vol. 1. IEEE. 2008, pp. 209–215.

[61] Edgar Alonso Lopez-Rojas and Stefan Axelsson. "Multi agent based simulation (mabs) of financial transactions for anti money laundering (aml)". In: *Nordic Conference on Secure IT Systems*. Blekinge Institute of Technology. 2012.

[62] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[63] Changjie Lu et al. "Compare Shallow Neural Network and Conventional Machine Learning in Predicting Money Laundering Crimes". In: ().

[64] Linyuan Lü and Tao Zhou. "Link prediction in complex networks: A survey". In: *Physica A: statistical mechanics and its applications* 390.6 (2011), pp. 1150–1170.

[65] Yadong Zhou et al. "Analyzing and detecting money-laundering accounts in online social networks". In: *IEEE Network* 32.3 (2017), pp. 115–121.

[66] Utku Görkem Ketenci et al. "A Time-Frequency Based Suspicious Activity Detection for Anti-Money Laundering". In: *IEEE Access* 9 (2021), pp. 59957–59967.

[67] Oren Anava and Kfir Levy. "k*-nearest neighbors: From global to local". In: *Advances in neural information processing systems*. 2016, pp. 4916–4924.

[68] N Malini and M Pushpa. "Analysis on credit card fraud identification techniques based on KNN and outlier detection". In: *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*. IEEE. 2017, pp. 255–258.

[69] Xin Yao. "Evolving artificial neural networks". In: *Proceedings of the IEEE* 87.9 (1999), pp. 1423–1447.

[70] Jingguang Han et al. "Nextgen aml: Distributed deep learning based language technologies to augment anti money laundering investigation". In: Association for Computational Linguistics. 2018.

[71] Neda Heidarinia, Ali Harounabadi, and Mehdi Sadeghzadeh. "An intelligent anti-money laundering method for detecting risky users in the banking systems". In: *International Journal of Computer Applications* 97.22 (2014).

[72] Qifeng Yang, Bin Feng, and Ping Song. "Study on anti-money laundering service system of online payment based on union-bank mode". In: *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE. 2007, pp. 4991–4994.

[73] Pavlo Tertychnyi et al. "Scalable and imbalance-resistant machine learning models for anti-money laundering: A two-layered approach". In: *International Workshop on Enterprise Applications, Markets and Services in the Finance Industry*. Springer. 2020, pp. 43–58.

[74] Lina Zhou et al. "Machine learning on big data: Opportunities and challenges". In: *Neurocomputing* 237 (2017), pp. 350–361.

[75] Daniel A Harris et al. "Using real-world transaction data to identify money laundering: Leveraging traditional regression and machine learning techniques". In: *STEM Fellowship Journal* 7.1 (2022), pp. 21–32.

[76] Xuan Liu and Pengzhu Zhang. "A scan statistics based suspicious transactions detection model for anti-money laundering (AML) in financial institutions". In: *2010 International Conference on Multimedia Communications*. IEEE. 2010, pp. 210–213.

[77] Florian Hahne et al. "Unsupervised machine learning". In: *Bioconductor case studies* (2008), pp. 137–157.

[78] Monika Kalra, Niranjan Lal, and Samimul Qamar. "K-mean clustering algorithm approach for data mining of heterogeneous data". In: *Information and Communication Technology for Sustainable Development*. Springer, 2018, pp. 61–70.

[79] Zhiyuan Chen et al. "Exploration of the effectiveness of expectation maximization algorithm for suspicious transaction detection in anti-money laundering". In: *2014 IEEE Conference on Open Systems (ICOS)*. IEEE. 2014, pp. 145–149.

[80] Rui Liu et al. "Research on anti-money laundering based on core decision tree algorithm". In: *2011 Chinese Control and Decision Conference (CCDC)*. IEEE. 2011, pp. 4322–4325.

[81] Claudio Alexandre and João Balsa. "Integrating client profiling in an anti-money laundering multi-agent based system". In: *New Advances in Information Systems and Technologies*. Springer, 2016, pp. 931–941.

[82] Mercedes Fernández-Redondo, Joaquın Torres-Sospedra, and Carlos Hernández-Espinosa. "Training RBFs networks: A comparison among supervised and not supervised algorithms". In: *International Conference on Neural Information Processing*. Springer. 2006, pp. 477–486.

[83] Claudio Alexandre and Joao Balsa. "Client profiling for an anti-money laundering system". In: *arXiv preprint arXiv:1510.00878* (2015).

[84] Nhien-An Le-Khac, Sammer Markos, and Mohand-Tahar Kechadi. "Towards a new data mining-based approach for anti-money laundering in an international investment bank". In: *International Conference on Digital Forensics and Cyber Crime*. Springer. 2009, pp. 77–84.

[85] Nhien An Le Khac and M-Tahar Kechadi. "Application of data mining for anti-money laundering detection: A case study". In: *2010 IEEE international conference on data mining workshops*. IEEE. 2010, pp. 577–584.

[86] Rafał Dreżewski, Jan Sepielak, and Wojciech Filipkowski. "The application of social network analysis algorithms in a system supporting money laundering detection". In: *Information Sciences* 295 (2015), pp. 18–32.

[87] Asma S Larik and Sajjad Haider. "Clustering based anomalous transaction reporting". In: *Procedia Computer Science* 3 (2011), pp. 606–610.

[88] Victoria Hodge and Jim Austin. "A survey of outlier detection methodologies". In: *Artificial intelligence review* 22.2 (2004), pp. 85–126.

[89] Thai Pham and Steven Lee. "Anomaly detection in bitcoin network using unsupervised learning methods". In: *arXiv preprint arXiv:1611.03941* (2016).

[90] Ebberth L Paula et al. "Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering". In: *2016 15th ieee international conference on machine learning and applications (icmla)*. IEEE. 2016, pp. 954–960.

[91] Danilo Labanca et al. "Amaretto: an active learning framework for money laundering detection". In: *IEEE Access* 10 (2022), pp. 41720–41739.

[92] Sarah M Erfani et al. "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning". In: *Pattern Recognition* 58 (2016), pp. 121–134.

[93] Amr Ehab Muhammed Shokry, Mohammed Abo Rizka, and Nevine Makram Labib. "Counter terrorism finance by detecting money laundering hidden networks using unsupervised machine learning algorithm". In: *Proc. 13th IADIS Int. Conf.(ICT), Soc. Hum. Beings (ICT), 6th IADIS Int. Conf. Connected Smart Cities (CSC), 17th IADIS Int. Conf. Web Based Communities Social Media (WBC), 14th Multi Conf. Comput. Sci. Inf. Syst.(MCCSIS), 13th*

IADIS Int. Conf. ICT, Soc. Hum. Beings (ICT), 6th IADIS Int. Conf. Connected Smart Cities (CSC), 17th IADIS Int. Conf. Web Based Communities Social Media (WBC), 14th Multi Conf. Comput. Sci. Inf. Syst.(MCCSIS IADIS). 2020, pp. 89–97.

[94] Ramiro Daniel Camino et al. "Finding suspicious activities in financial transactions and distributed ledgers". In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2017, pp. 787–796.

[95] Kandan Chitra and Balakrishnan Subashini. "Data mining techniques and its applications in banking sector". In: *International Journal of Emerging Technology and Advanced Engineering* 3.8 (2013), pp. 219–226.

[96] Dang Khoa Cao and Phuc Do. "Applying data mining in money laundering detection for the Vietnamese banking industry". In: *Asian Conference on Intelligent Information and Database Systems*. Springer. 2012, pp. 207–216.

[97] Xingqi Wang and Guang Dong. "Research on money laundering detection based on improved minimum spanning tree clustering and its application". In: *2009 Second international symposium on knowledge acquisition and modeling*. Vol. 2. IEEE. 2009, pp. 62–64.

[98] Reza Soltani et al. "A new algorithm for money laundering detection based on structural similarity". In: *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE. 2016, pp. 1–7.

[99] Zengan Gao. "Application of cluster-based local outlier factor algorithm in anti-money laundering". In: *2009 International Conference on Management and Service Science*. IEEE. 2009, pp. 1–4.

[100] Tianqing Zhu. "Suspicious financial transaction detection based on empirical mode decomposition method". In: *2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*. IEEE. 2006, pp. 300–304.

[101] Zhiyuan Chen et al. "Variational Autoencoders and Wasserstein Generative Adversarial Networks for Improving the Anti-Money Laundering Process". In: *IEEE Access* 9 (2021), pp. 83762–83785.

[102] Andrea Fronzetti Colladon and Elisa Remondi. "Using social network analysis to prevent money laundering". In: *Expert Systems with Applications* 67 (2017), pp. 49–58.

[103] Catarina Oliveira et al. "GuiltyWalker: Distance to illicit nodes in the Bitcoin network". In: *arXiv preprint arXiv:2102.05373* (2021).

[104] Xiaobing Sun et al. "CubeFlow: Money laundering detection with coupled tensors". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2021, pp. 78–90.

[105] Xurui Li et al. "Intelligent anti-money laundering solution based upon novel community detection in massive transaction networks on spark". In: *2017 fifth international conference on advanced cloud and big data (CBD)*. IEEE. 2017, pp. 176–181.

[106] Ahmad Naser Eddin et al. "Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs". In: *arXiv e-prints* (2021), arXiv–2112.

[107] Pariwat Ongsulee. "Artificial intelligence, machine learning and deep learning". In: *2017 15th international conference on ICT and knowledge engineering (ICT&KE)*. IEEE. 2017, pp. 1–6.

[108] Giang Nguyen et al. "Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey". In: *Artificial Intelligence Review* 52.1 (2019), pp. 77–124.

[109] Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. "Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain". In: *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*. 2020, pp. 23–27.

[110] Wai Weng Lo, Siamak Layeghy, and Marius Portmann. "Inspection-L: Practical GNN-Based Money Laundering Detection System for Bitcoin". In: *arXiv preprint arXiv:2203.10465* (2022).

[111] Pingfan Xia et al. "A Novel Spatiotemporal Prediction Approach Based on Graph Convolution Neural Networks and Long Short-Term Memory for Money Laundering Fraud". In: *Arabian Journal for Science and Engineering* 47.2 (2022), pp. 1921–1937.

[112] Mark Weber et al. "Scalable graph learning for anti-money laundering: A first look". In: *arXiv preprint arXiv:1812.00076* (2018).

[113] Nathalie Japkowicz and Shaju Stephen. "The class imbalance problem: A systematic study". In: *Intelligent data analysis* 6.5 (2002), pp. 429–449.

[114] R Bharat Rao, Sriram Krishnan, and Radu Stefan Niculescu. "Data mining for improved cardiac care". In: *Acm Sigkdd Explorations Newsletter* 8.1 (2006), pp. 3–10.

[115] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. "Predicting disease risks from highly imbalanced data using random forest". In: *BMC medical informatics and decision making* 11.1 (2011), pp. 1–13.

[116] Bartosz Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* 5.4 (2016), pp. 221–232.

[117] Haibo He and Edwardo A Garcia. "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[118] Wei Wei et al. "Effective detection of sophisticated online banking fraud on extremely imbalanced data". In: *World Wide Web* 16.4 (2013), pp. 449–475.

[119] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. "Experimental perspectives on learning from imbalanced data". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 935–942.

[120] Shivani Tyagi and Sangeeta Mittal. "Sampling approaches for imbalanced data classification problem in machine learning". In: *Proceedings of ICRIC 2019*. Springer, 2020, pp. 209–221.

[121] Thanh Cong Tran and Tran Khanh Dang. "Machine learning for prediction of imbalanced data: Credit fraud detection". In: *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. IEEE. 2021, pp. 1–7.

[122] Amit Singh, Ranjeet Kumar Ranjan, and Abhishek Tiwari. "Credit card fraud detection under extreme imbalanced data: a comparative study of data-level algorithms". In: *Journal of Experimental & Theoretical Artificial Intelligence* 34.4 (2022), pp. 571–598.

[123] Sara Makki et al. "An experimental study with imbalanced classification approaches for credit card fraud detection". In: *IEEE Access* 7 (2019), pp. 93010–93022.

[124] Junfei Qiu et al. "A survey of machine learning for big data processing". In: *EURASIP Journal on Advances in Signal Processing* 2016 (2016), pp. 1–16.

[125] Sebastián Maldonado, Richard Weber, and Fazel Famili. "Feature selection for high-dimensional class-imbalanced data sets using support vector machines". In: *Information sciences* 286 (2014), pp. 228–246.

[126] Ilnaz Jamali, Mohammad Bazmara, and Shahram Jafari. "Feature Selection in Imbalance data sets". In: *International Journal of Computer Science Issues (IJCSI)* 9.3 (2012), p. 42.

[127] Jason Van Hulse et al. "Feature selection with high-dimensional imbalanced data". In: *2009 IEEE International Conference on Data Mining Workshops*. IEEE. 2009, pp. 507–514.

[128] Bee Wah Yap et al. "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets". In: *Proceedings of the first international conference on advanced data and information engineering (DaEng-2013)*. Springer. 2014, pp. 13–22.

[129] Charles X Ling and Victor S Sheng. "Cost-sensitive learning and the class imbalance problem". In: *Encyclopedia of machine learning* 2011 (2008), pp. 231–235.

[130] Justin M Johnson and Taghi M Khoshgoftaar. "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6.1 (2019), pp. 1–54.

[131] Andra Baltoiu, Andrei Patrascu, and Paul Irofti. "Community-level anomaly detection for anti-money laundering". In: *arXiv preprint arXiv:1910.11313* (2019).

[132] Toyotaro Suzumura et al. "Towards federated graph learning for collaborative financial crimes detection". In: *arXiv preprint arXiv:1909.12946* (2019).

[133] Md Rezaul Karim et al. "Catch Me If You Can: Semi-supervised Graph Learning for Spotting Money Laundering". In: *arXiv e-prints* (2023), arXiv–2302.

[134] Ahmad Naser Eddin et al. "Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs". In: *arXiv preprint arXiv:2112.07508* (2021).

[135] Pavlo Tertychnyi et al. "Time-aware and interpretable predictive monitoring system for Anti-Money Laundering". In: *Machine Learning with Applications* 8 (2022), p. 100306.

[136] Yusarina Mat Isa et al. "Money laundering risk: From the bankers' and regulators perspectives". In: *Procedia Economics and Finance* 28 (2015), pp. 7–13.

[137] Esman Kurum. "RegTech solutions and AML compliance: what future for financial crime?" In: *Journal of Financial Crime* (2020).

[138] Anna Simonova. "The risk-based approach to anti-money laundering: problems and solutions". In: *Journal of Money Laundering Control* 14.4 (2011), pp. 346–358.

[139] Mark Weber et al. "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics". In: *arXiv preprint arXiv:1908.02591* (2019).

[140] Edgar Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. "PaySim: A financial mobile money simulator for fraud detection". In: *28th European Modeling and Simulation Symposium, EMSS, Larnaca*. Dime University of Genoa. 2016, pp. 249–255.

[141] Toyotaro Suzumura and Hiroki Kanezashi. *Anti-Money Laundering Datasets: InPlusLab Anti-Money Laundering DataDatasets*. http://github.com/IBM/AMLSim/. 2021.

[142] Aldo Pareja et al. "Evolvegcn: Evolving graph convolutional networks for dynamic graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5363–5370.

[143] Daniel Berrar. *Cross-Validation*. 2019.

[144] Guang-Hui Fu, Lun-Zhao Yi, and Jianxin Pan. "Tuning model parameters in class-imbalanced learning with precision-recall curve". In: *Biometrical Journal* 61.3 (2019), pp. 652–664.

[145] Nitesh V Chawla. "Data mining for imbalanced datasets: An overview". In: *Data mining and knowledge discovery handbook* (2009), pp. 875–886.

[146] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning". In: *International conference on intelligent computing*. Springer. 2005, pp. 878–887.

[147] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2009, pp. 475–482.

[148] Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. "A comparative study on sampling techniques for handling class imbalance in streaming data". In: *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*. IEEE. 2012, pp. 1762–1767.

[149] Inderjeet Mani and I Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction". In: *Proceedings of workshop on learning from imbalanced datasets*. Vol. 126. ICML. 2003, pp. 1–7.

[150] Qiang Wang. "A hybrid sampling SVM approach to imbalanced data classification". In: *Abstract and Applied Analysis*. Vol. 2014. Hindawi. 2014.

[151] Yun Qian et al. "A resampling ensemble algorithm for classification of imbalance problems". In: *Neurocomputing* 143 (2014), pp. 57–67.

[152] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[153] Alan Jović, Karla Brkić, and Nikola Bogunović. "A review of feature selection methods with applications". In: *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. Ieee. 2015, pp. 1200–1205.

[154] Yanmin Sun et al. "Cost-sensitive boosting for classification of imbalanced data". In: *Pattern recognition* 40.12 (2007), pp. 3358–3378.

[155] Salman H Khan et al. "Cost-sensitive learning of deep feature representations from imbalanced data". In: *IEEE transactions on neural networks and learning systems* 29.8 (2017), pp. 3573–3587.

[156] Peng Cao, Dazhe Zhao, and Osmar Zaiane. "An optimized cost-sensitive SVM for imbalanced data learning". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 280–292.

[157] Sauptik Dhar and Vladimir Cherkassky. "Development and evaluation of cost-sensitive universum-SVM". In: *IEEE transactions on cybernetics* 45.4 (2014), pp. 806–818.

[158] Chen Qiu, Liangxiao Jiang, and Chaoqun Li. "Randomly selected decision tree for test-cost sensitive learning". In: *Applied Soft Computing* 53 (2017), pp. 27–33.

[159] Ana Palacios et al. "Cost-sensitive learning of fuzzy rules for imbalanced classification problems using FURIA". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 22.05 (2014), pp. 643–675.

[160] Hongyu Guo and Herna L Viktor. "Learning from imbalanced data sets with boosting and data generation: the databoost-im approach". In: *ACM Sigkdd Explorations Newsletter* 6.1 (2004), pp. 30–39.

[161] David Mease, Abraham J Wyner, and Andreas Buja. "Boosted classification trees and class probability/quantile estimation." In: *Journal of Machine Learning Research* 8.3 (2007).

[162] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[163] Palash Goyal and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018), pp. 78–94.

[164] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[165] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[166] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[167] Rahul Palamuttam and William Chen. "Evaluating Network Embeddings: Node2Vec vs Spectral Clustering vs GCN". In: ().

[168] Gary M Weiss and Foster Provost. *The effect of class distribution on classifier learning: an empirical study*. Tech. rep. Rutgers University, 2001.

[169] Takaya Saito and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one* 10.3 (2015), e0118432.

[170] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 233–240.

[171] Tom Fawcett. "ROC graphs: Notes and practical considerations for researchers". In: *Machine learning* 31.1 (2004), pp. 1–38.

[172] Xingrong Luo. "Suspicious transaction detection for anti-money laundering". In: *International Journal of Security and Its Applications* 8.2 (2014), pp. 157–166.

[173] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[174] Zijun Zhang. "Improved adam optimizer for deep neural networks". In: *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. Ieee. 2018, pp. 1–2.

[175] Bayu Nur Pambudi, Indriana Hidayah, and Silmi Fauziati. "Improving money laundering detection using optimized support vector machine". In: *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE. 2019, pp. 273–278.