**Erasmus University Rotterdam**
**Erasmus School of Economics**

**Thesis Msc Data Science & Marketing Analytics**

# Money Laundering and Bitcoins: Detecting Fraudulent Transactions Using Machine Learning

"*A lot of people think Bitcoin is highly anonymous and untraceable, that it is outside the view of regulators and authorities, but once you start digging into it, it's surprising how much sense you can make of it..*" Putnins (2019):

**Name:** Victor de Andrade
**Student number:** 427707
**Supervisor:** E. Raviv
**Second Assessor:** A.C.D. Donkers

## Abstract

This study explores a novel approach to detecting money laundering in Bitcoin transactions using machine learning models and sampling techniques. The study focuses on class imbalance and explores the performance of traditional classification models compared to a novel model that combines Gradient Boosted Decision Trees with Graph Neural Networks. The results show that the XGBoost model with all features outperforms other models in terms of the F1 score, despite the expectation that the novel model would benefit from the graph structure of the data. Furthermore, the study finds that Generative Adversarial Networks produce more realistic samples than traditional sampling methods. This finding is significant in research on money laundering detection due to the class imbalance nature of this context. The study also investigates the possibility of concept drift and explores offline and online learning models. The findings suggest no significant sign of concept drift. Still, the XGBoost model shows a high variance in the ROC curve, indicating the need for further research on regularization techniques, online learning models, and concept drift detection. Overall, the study provides insights into the importance of addressing class imbalance and the effectiveness of various machine learning models and sampling techniques in detecting money laundering in Bitcoin transactions. The study suggests that dimensionality reduction techniques such as PCA, t-SNE, and LDA can be applied to large datasets to improve model performance. Further research can extend the study by combining fraud detection at the transaction and account levels.

## Table of Contents

## 1. Introduction

*Tuesday, February 8, 2022*. "Two arrested for alleged conspiracy to launder $4.5 billion in stolen cryptocurrency" (U.S. Department of Justice Office of Public Affairs, 2022). The U.S. Department of Justice confiscated $3.6 billion in Bitcoin, making it the largest capture in its existence. In this lawsuit, the individuals were arrested for alleged conspiracy to launder money. Years before the arrest, hackers stole 119,757 Bitcoins from the cryptocurrency exchange Bitfinex during a significant security breach. Extracting large withdrawals right after the hack would have caused alarm bells to fire immediately. Thus, the suspects started routing small amounts of money over a six-year period using digital wallets designed to prevent tracing and state-of-the-art money laundering techniques (Chow, 2022). Deputy Attorney General L. O. Monaco on the case: "In a futile effort to maintain digital anonymity, the defendants laundered stolen funds through a labyrinth of cryptocurrency transactions. Thanks to the meticulous work of law enforcement, the department once again showed how it can and will follow the money, no matter what form it takes." (U.S. Department of Justice Office of Public Affairs, 2022).

The practice of money laundering has been around for thousands of years. Seagrave (2010) describes money laundering in its earliest forms; ancient Chinese merchants around 2000 B.C. used to hide their wealth from their rulers because the regional authorities banned various forms of private businesses. As a result, the merchants invested their capital into offshore businesses. The term "money laundering" originated more recently, during the roaring twenties in the 20th century. The infamous American gangster Al Capone aka "Scarface" used launderettes to conceal the proceeds of his empire of crime and convert these proceeds into a legitimate source of income (Duyne et al., 2003). Al Capone was later successfully prosecuted for tax evasion (Storm, 2013).

The first prominent law on anti-money laundering was introduced almost half a century later, with the U.S. 1970 Bank Secrecy Act. The U.S. Bank Secrecy Act requires banks and financial institutions to keep records, file reports, and report suspicious activity to detect and prevent money laundering (Buchanan, 2004). Later, in 1989, the G7 summit formed the Financial Action Task Force (FATF) as an international watchdog on money laundering. The awareness of money laundering and the efforts against it have intensified significantly ever since. The FATF is now the largest intergovernmental anti-money laundering body worldwide, with 205 associated jurisdictions (FATF, 2021). Throughout the years, several incidents have fueled further tightening of the screws for financial institutions. Some instances that illustrate this phenomenon include the 2008 financial crisis which accelerated the enactment of the Dodd-Frank Wall Street Reform and Consumer Protection Act, as well as the Foreign Account Tax Compliance Act (FACTA). Additionally, the Panama Papers scandal caused policymakers across the globe to raise regulatory measures for financial institutions, including the implementation of more demanding Know Your Customer (KYC) policies, such as the enactment of the Corporate Transparency Act. Last year, President Biden's administration heightened the bar even

further for financial institutions through rigorous compliance expectations (Vanderford, 2022). Regulators expect financial institutions to have the latest and most sophisticated machine learning systems to flag suspicious transactions. If financial institutions do not meet the high standards imposed by regulators, hefty fines are issued. For example, Dutch bank ABN AMRO had to reach a $574 million settlement after serious shortcomings in combating money laundering (Deutsch & Meijer, 2021). Ultimately, the evolving regulatory landscape highlights the urgent demand for advanced machine learning algorithms to effectively prevent money laundering.

With the creation of Bitcoin in 2009, the world's first cryptocurrency, the financial industry ushered into a new era of innovations. It was in 2018 that the cryptocurrency market caught traction with the large public, resulting in a push that led to a total market capitalization of around $730 billion. The total market capitalization of cryptocurrencies peaked in 2021 at a valuation of approximately $2.8 trillion (CoinMarketCap, 2022), which is larger than the size of the subprime mortgage market ($1.3 trillion) when it caused the global financial crisis in 2007 (European Central Bank, 2022). With the surges in popularity, participation of institutional investors, and exorbitant market capitalization figures, the crypto market is past its infancy. As a result, regulators have a new problem child. Foley et al. (2019) estimate that around $76 billion of illegal activity involve Bitcoin or 46% of all Bitcoin transactions in their sample, nearing the same size as the total narcotics market of the United States and Europe combined. The FATF warns about cryptocurrencies. The speed, global reach, and (pseudo) anonymity of these virtual assets threaten the world economy (FATF, 2020). This disruptive field within the finance industry attracts criminals who want to escape authorities' scrutiny. Without established regulation and oversight, this field is still referred to as the "Wild West" (European Central Bank, 2022).

Academic research has explored advanced Machine Learning techniques for detecting suspicious transactions based on data provided by traditional financial institutions (e.g., Tang & Yin, 2005; Lv et al., 2008; Zhang & Trubey, 2019). However, the decentralized and anonymous nature of cryptocurrencies, combined with multiple digital wallets and addresses to obscure the source of funds, presents a unique challenge for detecting illicit transactions. Additionally, the lack of regulation and oversight makes it difficult for authorities to detect and prevent money laundering. Moreover, the widespread use and the danger of the potential fraudsters lurking behind crypto's (pseudo)anonymity require attention. Therefore, further research is necessary to develop effective methods for detecting money laundering in Bitcoin transactions.

In current academic research on money laundering detection, the Gradient Boosted Decision Tree (GBDT) models are among the best in class classification models for money laundering detection (e.g. Ahmed, 2021; Lin et al., 2019; Jullum et al., 2021; Vassallo et al., 2021). On the other hand, recent academic literature also explores the usage of Graph Convolutional Neural networks (GCN) that

leverage the power of graph data and neural networks in the context of money laundering (Alarab et al., 2020; Weber et al., 2019). As money laundering involves complex transactions that are connected through a network-like structure, models performed on graph data might be able to reveal hidden relationships that are not detected by conventional classification models. Given the strengths of both Gradient Boosted Decision Trees and the potential of Graph Neural Networks, Ivanov and Prokhorenkova (2021) recently proposed a model that combines the best of both worlds in a model called the BGNN or GBDT+GNN. This model can be interpreted as a GNN with an embedding layer of the GBDT or as a GBDT with a parametric loss function (Ivanov & Prokhorenkova, 2021). This model shows promising results, but has not been tested in the context of money laundering detection yet.

Next to optimization of classification models, a predominant challenge in fraud detection is data imbalance, since illicit transactions are inherently rare and vastly outnumbered by licit transactions. This can lead to biased models with a high accuracy at identifying licit transactions, however failing to identify the cases of interest, the illicit transactions. To address the problem of class imbalance, researchers have developed various sampling methods. Common methods used in the context of money laundering include Under- and Oversampling and SMOTE sampling (e.g. Vassallo et al., 2021; Zhang & Trubey, 2019). However, recently a novel method for sampling was proposed by Goodfellow et al. (2014): Generative Adversarial Networks (GANs). The GAN sampling method is an innovative kind of sampling that makes use of the generative capabilities of neural networks, resulting in realistic data samples (Wang et al., 2017). Research on this novel sampling method shows promising results. However, the GAN sampling method has not been applied in the context of money laundering, and specifically in Bitcoin transactions. Given the fundamental necessity of sampling methods in the research of money laundering detection and the powerful potential of the GAN sampling approach, this sampling method deserves to be investigated in this context.

As a result, this paper aims to analyze the effectiveness of novel machine learning and sampling methods in the detection of money laundering in Bitcoin transactions. The *scope* of the research are supervised classification models and various sampling methods. The *aim* of the paper is to compare conventional and novel classification models and sampling methods. The paper focuses on the current state of literature on machine learning classifiers and sampling methods used to detect money laundering in Bitcoin transactions. The paper covers both the mechanics of the classification and sampling methods, as well as their respective advantages and disadvantages. The paper covers two main research questions:

Research question based on the novel sampling method:

R.Q.1: *How do Generative Adversarial Networks (GANs) impact the performance of money laundering detection models in Bitcoin transactions, compared to traditional sampling methods?*

Research question based on the novel classification model:

R.Q.2: *How does the performance of the Gradient Boosted Decision Tree + Graph Neural Network model compare to conventional models in detecting money laundering in Bitcoin transactions?*

This paper holds both academic relevance and yield practical implications. The paper contributes to the existing literature on money laundering, digital currencies, and machine learning methods. Furthermore, it advances the understanding of financial crimes and the implications of digital currencies. Finally, in terms of originality, it examines novel machine learning and sampling methods in combating money laundering, which has implications for academic research, regulators, and financial institutions.

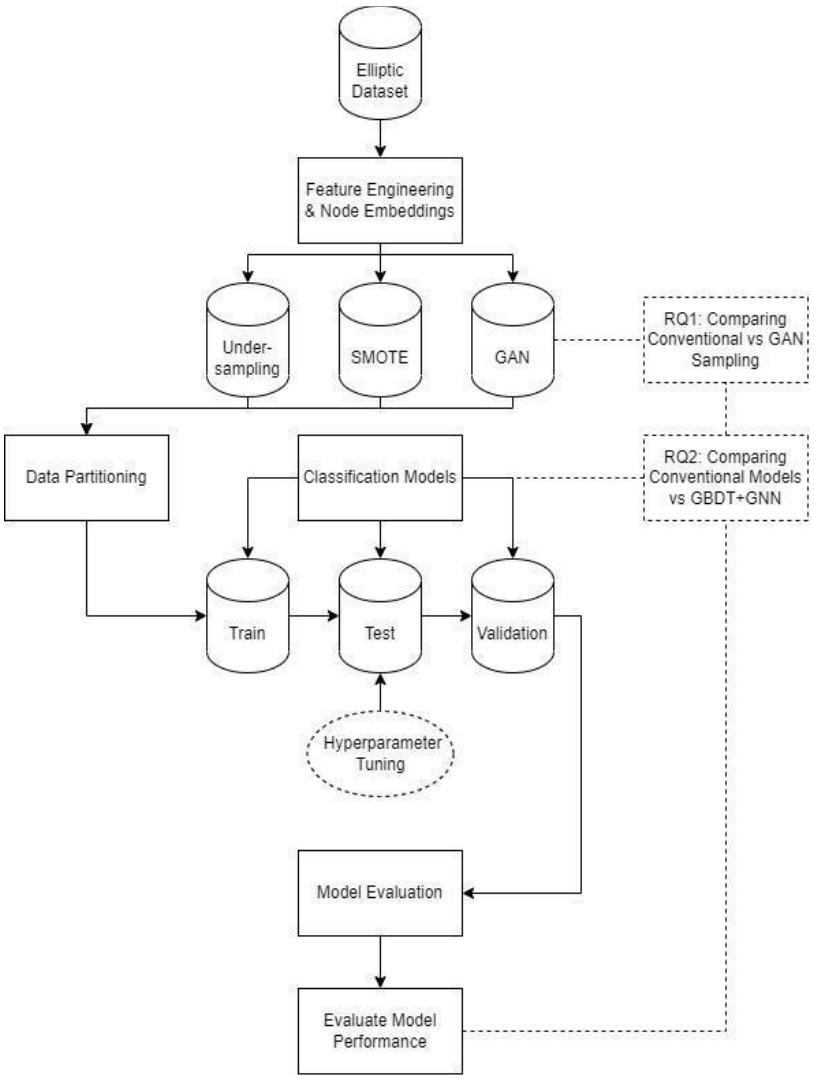


**Figure 1: Conceptual Model for this Research on comparing Sampling Methods and Classification Models**

# 2. Literature Review

With the increasing societal impact of money laundering and Bitcoin, academic research on these subjects and their intersection is expanding. This section explores related literature, starting with the definition of money laundering. The subsequent section gives an overview of Bitcoin and the technologies brought together in this financial innovation. The section concludes with the intersection of money laundering and Bitcoin; an analysis of the mechanics of money laundering in Bitcoin. Next, the literature review discusses the methods used to detect money laundering, from traditional to advanced machine learning methods. The review analyzes previous findings regarding the effectiveness of various techniques and considers the advantages and disadvantages of these methods. At last, a novel advanced machine learning method is discussed, Gradient Boosted Decision Trees with Graph Neural Networks, which shows potential to be a powerful method in detecting money laundering in Bitcoin transactions.

## 2.1 Background on Money Laundering and Bitcoin

### 2.1.1 Definition of Money Laundering

Money laundering is the practice of disguising the proceeds of criminal activity through transactions so that it appears to originate from an appropriate source (FATF, 2006).

Money laundering is a three-step process (Richardson et al., 2019): The first step is *placement*, which involves introducing illicit funds into the financial system. Placement can be done by depositing small, non-rounded amounts to multiple accounts. The second step is *layering*, which involves moving the funds through various financial transactions to disguise the source and ownership of the funds. For example, using unrelated people in different locations to transfer resources between accounts. The third step is *integration*, which involves using the funds for legitimate purposes. For instance, purchase tangible assets like real estate. The detection of money laundering attempts to identify suspicious transactions across these layers and trace funds derived from illicit sources.

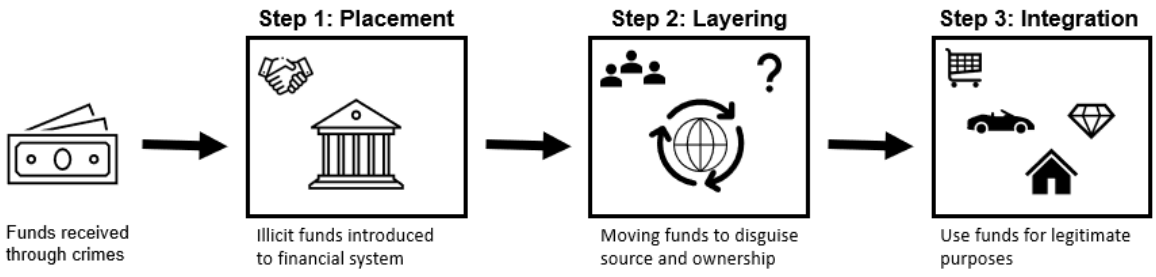


**Figure 2: Schematic Overview of the Money Laundering Process showing the Placement, Layering and Integration of Funds**

The following section gives an overview of Bitcoin and explains how the distributed ledger in Bitcoin records the placement, layering, and integration of the transactions. This transparency enables research on money laundering in Bitcoin transactions.

### 2.1.2 Overview of Bitcoin

In 2008, Satoshi Nakamoto published the whitepaper "Bitcoin: A Peer-to-Peer Electronic Cash System" (Nakamoto, 2008). The goal of this paper was to allow online payments without the need for a third party (Nakamoto, 2008). With this paper, Nakamoto brought together four innovations: digital signatures, the distributed ledger, blockchain technology, and the proof-of-work concept.

*Digital signatures* were invented in 1970 by the British intelligence agency Government Communications Headquarters (GCHQ) to have secure communication and check whether messages arrived unaltered (Vatra, 2009). Within Bitcoin, double-key cryptography ensures secure transfers of assets between digital wallets. This mechanism guarantees that a Bitcoin can only be transacted by the account that owns the Bitcoin.

*Blockchain* is a technology that records new data in blocks, sequentially, in a write-only, digitally *distributed ledger* proposed by Nakamoto (2008), that brings together several concepts developed since the 1970s (e.g., Wong, 1997; Merkle, 1978; Haber & Stornetta, 1990). The blockchain consists of data packages (blocks), where a block contains data on multiple transactions (Nofer et al., 2017). This technology makes changing the ledger impossible, as one must rewrite the entire blockchain to change one entry.

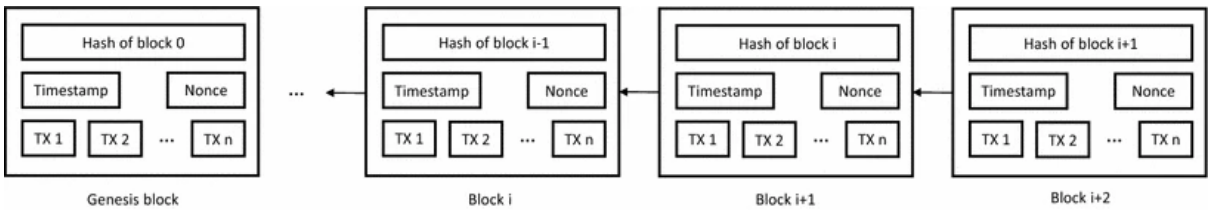


**Figure 3: Example of a Blockchain consisting where each Block contains Data on Multiple Transactions (Zheng et al., 2016)**

The distributed ledger replaces the need for a third party to guarantee correct data. Furthermore, the proposed technology is a crowd-sourcing solution where network members (nodes) compete in a proof-of-work competition. In proof-of-work, nodes solve a puzzle that requires computing power. The first node to solve this puzzle is rewarded if the network accepts a block. Proof-of-work validates transactions and prevents double spending of Bitcoins.

The result is a fast and easy-to-use cryptocurrency that allows users to execute transactions between digital wallets without needing a third party. Blockchain technology lowers transaction

uncertainty through identity management, asset tracking, and reneging deals. Transaction fees are low, and users are (allegedly) anonymous. However, in the current information age, anonymity generally involves pseudonymity and unlinkability (Nissenbaum, 1999). Bitcoin allows for pseudonymity as users can hide behind their digital wallets. But, as transactions are recorded on the indelible blockchain, Bitcoin does not offer unlinkability. This linkability enables institutions to track the sources of illicit funds and label transactions accordingly. Linkability is a vital property enabling research on money laundering regarding Bitcoin transactions, as labeled data is needed to apply supervised machine learning methods.

The following section discusses the intersection of money laundering and Bitcoin. The section outlines why Bitcoin is attractive for money laundering and how fraudsters execute money laundering in Bitcoin.

### 2.1.3 Money Laundering in Bitcoin

Bitcoin offers numerous benefits for fraudsters seeking to launder money. Some of the core concepts of Bitcoin as described by the whitepaper of Nakamoto (2008), are directly related to the advantages that Bitcoin provides in a money laundering context. One of the core aspects of Bitcoin is the pseudonymity it offers its users. Fraudsters are not required to disclose personal details to execute transactions. Consequently, it is challenging for regulators to find, trace, and link evidence to fraudsters. Next, as Bitcoin transactions are considerably faster than conventional bank payments, funds can be transferred rapidly and discretely. Additionally, Bitcoin transactions involve lower costs than conventional methods, making it an attractive money laundering technique. Furthermore, as Bitcoin is accessible everywhere, there are no constraints on transferring funds across borders. At last, the decentralized aspect of Bitcoin ensures there is no public authority or organization that will obstruct the (illicit) transactions. These considerations make Bitcoin a desirable option for fraudsters to launder money. Gaining an understanding of the mechanics of money laundering in Bitcoin is an integral part of the detection. The following paragraphs discuss the mechanics of the three-step money laundering process in the context of Bitcoin transactions: the placement, layering, and integration of illicit funds.

Fraudsters have multiple options to place illicit funds in the Bitcoin ecosystem. The first placement options involve cash transactions. Firstly, criminals can offer people cash in a real-life, face-to-face transaction. Subsequently, the counterparty transfers Bitcoins to the criminal's digital Bitcoin wallet. A second option for criminals is to buy stored-value cards, such as gift vouchers or phone cards, and buy Bitcoin through these cards.

At last, criminals can opt to deposit cash at a Bitcoin ATM. Across the Netherlands, 19 Bitcoin ATMs are located across large cities where people can deposit cash in exchange for cryptocurrencies (CoinATMRadar, 2023). Hyman (2015) describes the risks of these Bitcoin ATMs and proposes

possible solutions to detect and prevent money laundering through these machines. Among the proposed solutions by Hyman (2015) are the implementation of an I.D. card scanner, a real-time camera to capture images of the customer, and the integration of (facial recognition) software that compares data from the I.D. card with governmental libraries. Next to cash options for placement, it is also possible for criminals to place illegitimate digital money in the Bitcoin ecosystem. This could be done by digitally transferring money for Bitcoins in a peer-to-peer transaction or through (illegal) exchanges that do not collect personal information.

Subsequently, the layering step disguises the sources of the illicit funds. Using several digital wallets and exchanges not regulated by similar laws as conventional banks is one of the main ways to layer money in Bitcoin. Fraudsters can camouflage the sources and pathways of funds through these systems. In addition to this, criminals have unique crypto money laundering tools at their disposal, such as mixers and tumblers. Mixers and tumblers facilitate transactions by combining funds from different sources (Silva Ramalho & Igreja Matos, 2021). By combining funds, the mixer and tumblers disguise and 'mix' the funds' various sources and transaction histories. Subsequently, the service executes the desired transactions. Mixers and tumblers can be a beneficial tool for privacy protection and preventing third-party attacks. However, these tools also enable criminals to disguise the source of money that has been gained unlawfully and 'layer' it into the Bitcoin system (Silva Ramalho & Igreja Matos, 2021).

The last step in the money laundering process is the integration of Bitcoin. Once placed and layered, fraudsters can use legal crypto exchanges to cash out their proceeds. Wegberg et al. (2018) highlight that using crypto exchanges to cash out money laundering proceeds is user-friendly and cost-efficient relative to traditional money laundering methods. The cost margin of money laundering through Bitcoin mixers and cashing out through exchanges is estimated to be 15%, a significant cost reduction to the cost margin of up to 50% for traditional methods (Wegberg et al., 2018). These benefits strengthen the case for both interferences by law enforcement and the demand for research on money laundering detection methods in Bitcoin.

### 2.2 Context on Methods to Detect Money Laundering

This section of the literature review examines the literature on both traditional methods and machine learning methods to detect money laundering. Despite being used for some time, conventional methods are becoming less reliable in current financial practices. Due to the increasing complexity of modern economic systems and suspects becoming more refined, these methods may only be partially effective in anti-money laundering operations. On the other hand, research on sophisticated machine learning techniques is expanding, and novel methods show possible advantages and improvements in effectiveness.

### 2.2.1 Traditional Methods Employed in Money Laundering Detection

#### 2.2.1.1 Rule-Based Method

The rule-based method uses a set of predetermined rules and guidelines to identify potentially suspicious activity (e.g., Levi et al., 2014). In the Netherlands, for example, banks will flag cash deposits over 10 thousand euros (Zuurmond, 2022). This is a rule-based method related to the Dutch Anti-Money Laundering and Anti-Terrorist Financing Act (Wwft). The advantage of this method is that it is simple to use and interpret. This method's drawbacks include its high rate of false positives, poor thresholds, impractical data processing, and failure to automatically detect money laundering classifications and practical rules requiring expert knowledge in the respective domain (Vassallo et al. 2021).

Furthermore, money launderers actively seek loopholes in the set rules and adjust their approaches accordingly (Zhang & Trubey, 2019). This makes this method unsuitable for detecting money laundering in Bitcoin transactions. Because of the shortages of rule-based systems, the FATF started advocating for financial institutions to shift away from rule-based systems and towards a more risk-based approach since 2012 (FATF, 2012).

#### 2.2.1.2 Social Network Analysis

Social Network Analysis (SNA) can produce meaningful insights into the structure and dynamics of social networks. In SNA, an individual is represented by a *node*, while the connections between nodes are represented by *edges*. By identifying all people and relationships in a network, SNA can provide many statistics and insights into the network (Golbeck, 2013).

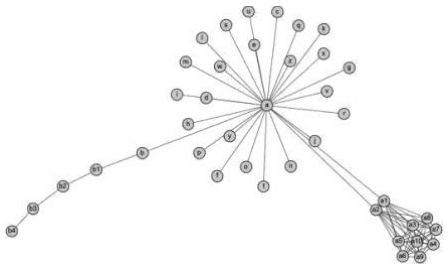

**Figure 4. Social Network Consisting of Nodes and Edges (Golbeck, 2013)**

In the detection of money laundering, SNA can serve as an important component of the detection system. By studying the structure of a network, it is possible to identify suspicious patterns of behavior that may indicate suspicious transactions (Dreżewski et al., 2015). For example, if a node performs many transactions in a short period, this may indicate money laundering. Also, nodes with many connections to other nodes may show a suspicious pattern of transactions. By uncovering

networks of nodes, it is possible to identify clusters of individuals that may be involved in money laundering.

One of the disadvantages of SNA is the large amount of data required to discover patterns, which can be challenging to obtain in the context of money laundering (Colladon & Remondi, 2017). Also, it may be difficult to distinguish between legitimate and suspicious activities, as patterns can be similar. At last, the accuracy of the SNA is limited by the accuracy of the data. This disadvantage is a significant obstacle in research regarding money laundering, as transactions in real data can only be labeled after the money laundering was confirmed, resulting in misleading data (Buchanan, 2004). Colladon and Remondi (2017) suggest combining network analysis with machine learning algorithms to detect suspicious nodes and prevent money laundering.

### 2.2.2 Machine Learning Methods Employed in Money Laundering Detection

This section discusses machine learning methods used for money laundering detection. Machine learning has become a powerful tool in many industries and is especially useful in detecting money laundering. Machine learning can analyze large and unstructured data quickly and accurately compared to traditional models. Traditional models rely on linear rules, whereas machine learning methods can capture complex patterns like nonlinear relationships. Moreover, in order to deliver accurate outcomes, machine learning models can learn from prior estimations and adapt to new inputs. The following subsections investigate existing literature on the effectiveness, pros, and cons of current detection methods applied to traditional money and cryptocurrencies. An explanation of the model mechanics of the standard machine learning models is delegated to Appendix A. A brief overview of the operating mechanisms is given in the text for the advanced models.

#### 2.2.2.1 Logistic Regression

Logistic regression is an often-used statistical approach that models a binary outcome using a number of independent variables by applying a logistic function. Current academic research on the detection of money laundering aims to decrease false positives while not increasing false negatives, as the goal is to allow legitimate transactions while detecting illicit transactions. In academic research regarding the concerned topic, Logistic Regression, and Random Forests are often used as benchmark methods (Weber et al., 2019). Additionally, Harris et al. (2019) use real-world transaction data to compare the performance of different algorithms in money laundering detection. Their findings suggest that the traditional logistic regression significantly outperforms other supervised machine learning methods (Harris et al., 2019). On the contrary, Zhang and Trubey (2019) find that Support Vector

Machine and Random Forest models performed as well or better than the logistic regression in detecting money laundering based on actual U.S. transaction data provided by a financial institution.

#### 2.2.2.2 Support Vector Machine

The Support Vector Machine (SVM) method is a robust classification model. The SVM has some strong properties, amongst others; the model excels in the classification of non-linearly distinguishable categories and does not need extensive training datasets, is less prone to overfitting compared to other models (e.g., decision trees), and it converges to a single and unified solution (Sudjianto et al., 2010; Ray, 2019; Zhang & Trubey, 2019). However, the drawbacks of SVM include that the results are difficult to understand, numerous algorithm parameters must be identified, and it requires a large amount of computing power in large datasets (Sudjianto et al., 2010; Zhang & Trubey, 2019). The most relevant shortcoming of SVM in money laundering detection is that the model appears to experience lacking performance in the classification of rare events (Zhang & Trubey, 2019). This is a possible explanation for the limited research on the performance of SVM compared to other methods in the context of money laundering detection. Şahin and Duman (2011) compared the performance of different SVM models with varying trees of decision models in the context of credit card fraud detection. The findings of Şahin and Duman (2011) suggest the decision tree models outperform the SVM-based models in accuracy. However, Şahin and Duman (2011) discuss that accuracy is not the only important metric in money laundering, as accuracy reflects the percentage of correct classifications, independent of whether they are true licit or true illicit classifications. This is a critical observation, as money laundering detection works with highly unbalanced datasets since the occurrence of 'regular' transactions is much higher than 'fraudulent' transactions. Thus, other performance metrics, such as Precision, Recall, F1, and Area Under the Curve (AUC), are required for a fair performance evaluation. Unfortunately, these performance metrics were not included in the paper by Şahin and Duman (2011).

On the other hand, Zhang and Trubey (2019) do include these different performance measures in their comparison of the performance of SVM compared to other models in a money laundering detection study. In their study, Zhang and Trubey (2019) also investigate the effects of under- and oversampling on the performance of classification methods in this context because of the inherent data imbalances in money laundering detection cases. Findings by Zhang and Trubey (2019) suggest that SVM performed worst across all models (i.e., Decision Tree, Random Forest, and ANN) on the data without sampling and gained the most performance from sampling. After sampling, findings suggest that SVM outperformed the Logistic Regression and Random Forest regarding the Area Under the Curve metric for specific low event rate ranges and even exceeded ANN in high event regimes when oversampling was used (Zhang & Trubey, 2019).

#### 2.2.2.3 Decision Trees

Decision trees work by drawing a flowchart-like structure of decisions and their outcomes based on a set of input data. In early research on money laundering detection, Senator et al. (1995) asserted that tree-based models displayed powerful capabilities in the respective context. However, as the unavailability of labeled data has been a continuous bottleneck in research on the concerning topic, proving the capabilities of the methods in this context was particularly challenging at the time. Nonetheless, decision trees offer several advantages over other techniques. Decision trees are easy to understand and interpret, and provide clear visualizations

Additionally, they need minimal data preparation, are capable of dealing with multicollinearity, and can handle both numerical and categorical data (de Oña et al., 2014; Ray, 2019). Furthermore, decision trees indicate which fields are important for prediction or classification. However, decision trees are also prone to overfitting and can be unstable due to variations in the data (Ray, 2019). Some supervision is then required to make sure the model is detecting patterns rather than noises in the data (Zhang & Trubey, 2019). Moreover, decision trees can generate biased trees in cases where some classes dominate. Kumar et al. (2021) compare the use of Decision Trees and Support Vector Machines in money laundering detection on a dataset consisting of raw data collected from secured sites. In their research, Kumar et al. (2021) find that the Decision Tree outperforms the Support Vector Machine in terms of precision, accuracy, and recall values, possibly due to the ability of Decision Trees to handle multicollinearity efficiently.

#### 2.2.2.4 Random Forests

Random Forests are combinations of decision tree predictors, each depending on a randomly sampled vector, a concept introduced by Breiman (2001). Compared to decision trees, the random forest method is more precise in calculating the error rate; Breiman (2001) established that the error rate converges as the number of trees grows. On the other hand, Schonlau and Zou (2020) argue this is a tradeoff, as random forests lose interpretability compared to 'weak' decision trees that are intuitive to interpret since random forests aggregate multiple decision trees. Weber et al. (2019) emphasize the power of jointly using logistic regression and random forests as benchmarks in money laundering, logistic regression for its explainability, and random forests for its accuracy.

In academic research on random forests in the case of money laundering detection, random forest models have previously been described as "best in class" for fraud detection (Bartoletti, 2018). Findings by Raiter (2021) confirm this description, as Raiter (2021) finds that compared to other techniques (e.g., Logistic regression, Support Vector Machines and Artificial Neural Networks), random forests perform best in terms of both accuracy and interpretability. It is important to note that in their research, Raiter (2019) uses a synthetic transaction data set due to a lack of publicly accessible

data. In contrast, Monamo et al. (2016) use a real Bitcoin transaction dataset accommodated by the Laboratory for Computational Biology at the University of Illinois. In their comparison of supervised detection models for money laundering, Monamo et al. (2016) found that the random forest model was the most successful classifier compared to GLM logistic regression and boosted logistic regression, irrespective of class imbalances. The findings by Weber et al. (2019) reinforce the idea of the random forest as an accurate classifier. In their paper, Weber et al. (2019) compare the performances of logistic regression, random forest, Multilayer Perceptrons, and Graph Convolutional Networks on real-world bitcoin transaction data provided by the Elliptic dataset. In this study, random forests outperformed the other models, even when graph convolutional networks leverage the power of graph data. The Graph Neural Networks section of the literature review further elaborates on the findings of the paper and the suggestions for future research by Weber et al. (2019).

#### 2.2.2.5 Gradient Boosted Decision Trees

Gradient Boosted Decision Trees (GBDT) are, similar to random forests, a decision tree ensemble method. However, the GBDT method uses a boosting approach, whereas random forest uses a bagging approach. In GBDT, the trees are built sequentially to correct the mistakes of prior trees (Vassallo et al., 2021). Vassallo et al. (2021) note that tree-based ensemble methods are among the most popular methods in money laundering detection. The advantages of the GBDT method are the model's cutting-edge results in classification problems, scalability, and being less prone to overfitting due to the combination of weak learners (Chen & Guestrin, 2016; Vassallo et al., 2021). Yet, which ensemble method is best at identifying illegal activity in cryptocurrency transactions is undetermined.

Jullum et al. (2021) note that, despite the apparent necessity for research on machine learning methods in anti-money laundering, the literature on detection methods is still scarce. A recurring theme in the literature on money laundering detection is the lack of real-world data. However, Jullum et al. (2021) were able to access real-world transaction data provided by Norway's largest bank, the DNB. Experiments on this real-world transaction dataset suggest the potential of Gradient Boosted Decision Trees in the context of money laundering detection through the XGBoost library presented by Chen and Guestrin (2016). Jullum et al. (2021) highlight the models' efficiency, scalability, and ability to decrease training time compared to traditional methods of money laundering detection (i.e., rule-based systems).

Research by Lin et al. (2019) on real-world Bitcoin transaction data gathered by Toyoda & Ohtsuki (2018) shows the power of GBDT models as classifiers by leveraging transaction history to identify abnormal Bitcoin addresses. In their paper, Lin et al. (2019) compare the model performance of, amongst others, logistic regression, SVM, random forest, GBDT, and Neural Networks. Lin et al. (2019) adopt the stratified random sampling method to address the imbalance in the data. Lin et al.

(2019) find that random forests, GBDT models, and neural network models are the best-performing models out of the eight tested machine learning methods. Here, the GBDT model (Light GBM) shows the most strong consistent performance, and the Neural Network produces the most significant outcome but is less reliable across the tests.

Previous experiments on the Elliptic dataset research the power of Gradient Boosted Decision Trees in the context of money laundering in Bitcoin transactions. Vassallo et al. (2021) compared the performance of different GBDT models (i.e., XGBoost, Light Gradient Boosting Machine (LGBM), and CatBoost) against the performance of Random Forests. Furthermore, Vassallo et al. (2021) highlight the importance of research on data-sampling techniques in the context of money laundering detection (in cryptocurrencies) to offset skewed class distributions. As a result, the paper additionally investigates model performance on different sampling methods (i.e., Synthetic Minority Over-Sampling (SMOTE), Neighbourhood Cleaning Rule (NCL), and NCL-SMOTE). The results show differences in model performance ranking across the different sampling methods, emphasizing the importance of data sampling in the given context. Analyzing various sampling techniques, such as the hybrid of under- and oversampling (i.e., NCL-Smote), demonstrates false negatives may be reduced (Vassallo et al., 2021). Furthermore, the results of the paper by Vassallo et al. (2021) give no conclusive evidence on which model performs better (GBDT or R.F.). Still, findings show the underperformance of Catboost compared to XGBoost and LGBM, providing a reason to prefer the XGBoost or LGBM model for this paper over the CatBoost model.

On the other hand, Ahmed (2021) finds evidence for GBDT models (Light Gradient Boosting Algorithm and XGboost) outperforming the random forest model on the Elliptic dataset. In their paper, Ahmed (2021) compares similar data sampling methods as Vassallo et al. (2021) (i.e., NCL, SMOTE, NCL-SMOTE), confirming the finding that employing advanced sampling techniques can reduce false negatives. The results show GBDT models were as good as the R.F. model in precision and outperformed R.F. in accuracy, recall, and F1 score. The difference in outcomes between the papers of Vassallo et al. (2021) and Ahmed (2021) underscores the relevance of hyperparameter tuning. The paper's similarity in terms of dataset and data sampling results in comparable outcomes for the models before hyperparameter tuning. However, Ahmed (2021) accredits the model improvements to the Bayesian hyper-parameter optimization technique proposed by Xia et al. (2017). In contrast, Vassallo et al. (2021) adopted a variation named the Tree Structured Parzen Estimator (TPE), suggested by Bergstra et al. (2011), for its effectiveness in handling high dimensionality.

#### 2.2.2.6 Graph Neural Networks

As described in the previous paragraphs, various methods have been created to locate anomalies in Euclidean data. However, as graph data becomes more prevalent due to rich relation information among elements, machine learning approaches for structured graph data receive increasingly more interest (Heidari et al., 2022). Graphs are a type of data structure that depicts a collection of objects (nodes) and the relationship (edges) between them, similarly as previously illustrated in the SNA in figure 4 (Zhou et al., 2020). Graph data contains up to four types of information; nodes, edges, global context, and connectivity (Sanchez-Lengeling et al., 2021). Graph Neural Networks (GNNs) are a class of Neural Network-based machine learning algorithms designed to operate on graph data.

Compared to the Euclidean data with a "fixed-size input and output" used in the previously discussed models (L.R., D.T., R.F., and SVM), graphs can be irregular and may have variable unordered nodes and edges and varying numbers of neighbors (Wu et al., 2020). As a result, GNNs take on a "graph-in, graph-out" design, where the existing node and edge embeddings are transformed without altering the connectivity of the input graph (Sanchez-Lengeling et al., 2021) (see schematic overview in figure 5). Furthermore, Wu et al. (2020) argue that another difference between the existing models (e.g., L.R., D.T., R.F., and SVM) and graph-based models (i.e., GNN) is the assumption of independence of observations. In contrast to the previously discussed model, this assumption no longer holds for graph-based models, as nodes are indeed related through edges and describe, for instance, social relations or interactions (Wu et al., 2020).

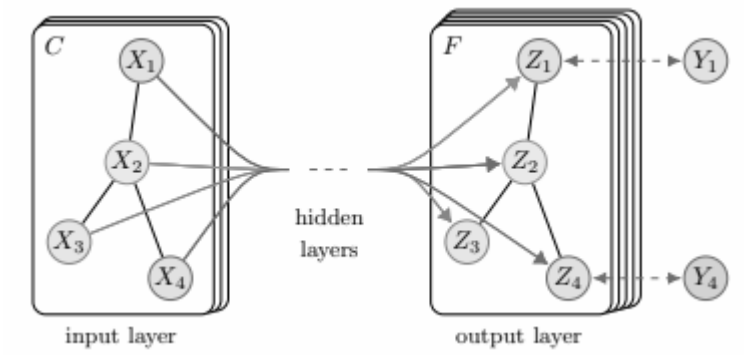


**Figure 5: Schematic Overview of "Graph-in Graph-out" Design in a Multi-Layer GCN (Kipf & Welling, 2017)**

Applying traditional Neural Networks methods to graph data is challenging, as these models are designed to take fixed, grid-like arrays as input (Sanchez-Lengeling et al., 2021). To handle this issue, commonly used concepts in GNN are the message-passing mechanism and convolution, as applied in the Graph Convolutional Network (GCN) method. The message-passing mechanism updates node representations by passing messages between nodes based on aggregated information from adjacent nodes (Sanchez-Lengeling et al., 2021; Zhou et al., 2020). At the same time, convolution is the

concept of aggregating the local information of each node. The process of updating the graph and computing new messages is repeated multiple times in the Neural Network's layers until all nodes' representations have been updated; see the schematic overview in figure 6.

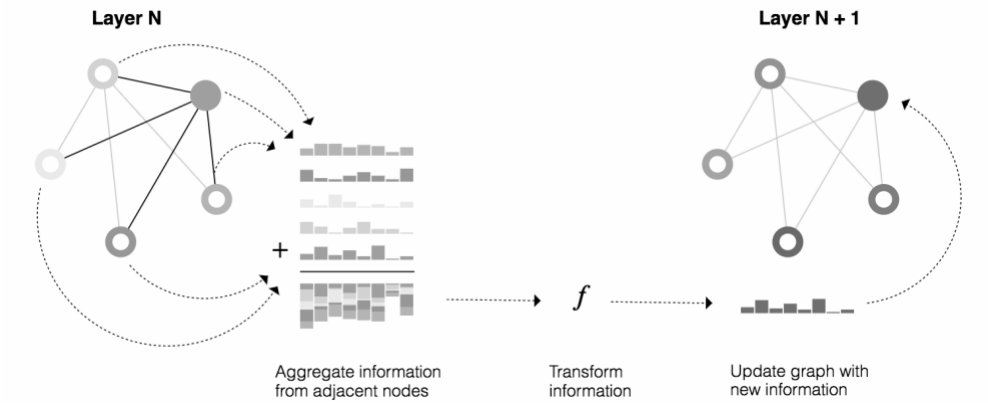


**Figure 6: Message Passing Mechanism and Convolution in GCN (Sanchez-Lengeling et al., 2021)**

Recent academic research focuses on GCN to detect money laundering in Bitcoin transactions, as this version of GNN emphasizes the representations of nodes and edges of the graph based on the local structure and relationships (Weber et al., 2019; Alarab et al., 2020; Alarab & Prakoonwit, 2022). This emphasis is an advantage in money laundering detection, as illicit transactions can be identified through the connections and patterns between nodes in the network. On the other hand, among the disadvantages of GCN are the high complexity, the model is computationally intensive, which makes it less scalable, and low interpretability as the learned representations are highly dimensional and abstract.

Academic research on advanced machine learning applications to detect money laundering in Bitcoin received great stimulus with the public release of the Elliptic dataset, the most sizable dataset containing labeled data on money laundering in cryptocurrencies to date. The dataset was released to stimulate research and development of machine learning techniques to detect money laundering in cryptocurrencies (Robinson, 2019). Along with the dataset's publication, Weber et al. (2019), a collective of researchers from Elliptic Ltd. and the MIT-IBM Watson AI Lab, published a paper on experiments using GCN to detect money laundering in Bitcoin transactions using the Elliptic dataset. When evaluating the performance of GCN to L.R. and RF, Weber et al. (2019) find that R.F. outperforms both L.R. and GCN. These results might be surprising, as GCN leverages the power of graph data. However, Weber et al. (2019) argue that GCN can be seen as a nontrivial generalization of L.R., as GCN uses L.R. as the final output layer. As a result, Weber et al. (2019) suggest future research to explore the possibilities of combining R.F. with GCN to get the best of both worlds.

In succession of the paper by Weber et al. (2019), Alarab et al. (2020) propose an extension of GCN to detect money laundering in Bitcoin transactions using the Elliptic dataset. The novel approach is modeled by combining linear layers and the already-existing GCN; A single hidden layer created by the linear transformation of the node feature matrix is combined with node embeddings acquired from graph convolutional layers in the approach, which is then followed by a multilayer perceptron (Alarab et al., 2020). The result is a method that surpasses the performance previously achieved by Weber et al. (2019). For future work, Alarab et al. (2020) suggest real-time linked data as a potential feature to add to the GCN input.

#### 2.2.2.7 Gradient Boosted Decision Tree with Graph Neural Networks

In learning from graph data, GNNs have been demonstrated to be a powerful tool (Zhou et al., 2020). The GNN framework effectively transforms input data into meaningful representations to produce significant outcomes. However, real-world data often consists of tabular data with extensive information rich in semantics. For example, in money laundering research, data consists of personal information regarding sex, age, and geography. For that reason, Ivanov and Prokhorenkova (2021) explore the possibilities of combining GNN with Gradient Boosted Decision Trees (GBDT), as the latter has proven to be successful for tasks involving tabular data (Bentéjac et al., 2021).

The reasons why GBDT is particularly effective for tabular data, according to Ivanov and Prokhorenkova (2021), include the following; GBDT effectively learns decision spaces with borders resembling hyperplanes, which are frequent in tabular data; GBDT is suitable for dealing with higher cardinality parameters, features with missing data and features of various scales; GBDT offers decision trees a meaningful explanation; even for extensive volumes of data, GBDT often converges more quickly in pragmatic situations. On the other hand, Ivanov and Prokhorenkova (2021) argue GNN has several advantages over GBDT; to generate projections, GNN considers both the nodes' local information and their characteristics, whereas GBDT only considers the latter. Furthermore, it is theoretically demonstrated that message-passing GNNs can calculate any Turing machine-computable function on its graph input (Keriven & Peyré, 2019; Maron et al., 2019).

Finally, Ivanov and Prokhorenkova (2021) assert several benefits to gradient-based neural network learning over tree-based methods; GNNs with relational inductive bias eliminate the requirement to design features to represent the network architecture explicitly.

Ivanov and Prokhorenkova (2021) propose a model that yields the best of both worlds from GBDT and GNN. The presented model merges GBDT learning on tabular node characteristics with GNNs prediction-refinement using the structure of graphs. The solution is a Boosted Graph Neural Network (BGNN) that benefits from the representation learning and end-to-end training of GNN and

the heterogeneous learning and interpretability of gradient-boosted techniques (Ivanov and Prokhorenkova, 2021).

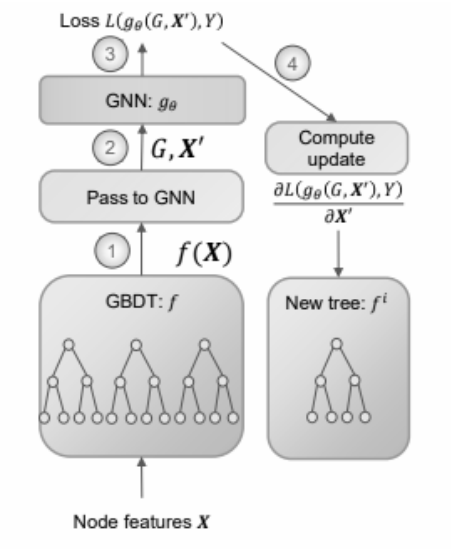


**Figure 7: Schematic Overview of End-to-End Training of GBDT and GNN (Ivanov & Prokhorenkova, 2021)**

The proposed model has two interpretations; it can be viewed as a GNN with an embedding layer of the GBDT or as a GBDT with a parametric loss function. Figure 7 displays a schematic overview of the presented BGNN model. By enabling the updated trees to match the gradient modifications of GNN, end-to-end optimization improves the model (Ivanov & Prokhorenkova, 2021).

Application of the BGNN model on classification tasks shows promising results. In their paper, Ivanov & Prokhorenkova (2021) find that the proposed BGNN outperforms existing alternatives (GBDT, GNN, and N.N.) regarding the accuracy of predictions and training time. These results make this novel model a compelling candidate for this research on the detection of money laundering in Bitcoin transactions.

### 2.3 Data Imbalance in Money Laundering Detection

Data imbalance is a significant concern in the research of money laundering as data imbalance is inherent to the topic of fraud detection. As fraud is naturally a rare event, datasets on money laundering often consist of a small minority class and a large majority class.

The occurrence of imbalance in a dataset causes problems in money laundering detection models. Classification models are developed to reduce misclassification rates and increase model accuracy. These models assume that the studied data includes approximately an equal number of observations for each class (Thabtah et al., 2020). As a result, the classification models tend to generate

a bias towards the majority class in settings with unbalanced data, deteriorating the performance. Moreover, classifiers aim to improve the model's accuracy in traditional settings. However, as previously discussed in the literature review, Şahin and Duman (2011) highlight that accuracy is not the most appropriate metric in a money laundering detection setting. The model accuracy might perform well on the majority class and, by extension, the overall dataset, while the importance in this context lies in classifying the minority class (Thabtah et al., 2020).

A potential solution to the problem of dataset imbalance is data sampling. The majority of studies in the field of money laundering detection concentrate on non-heuristic (e.g., random under- and over-sampling) or conventional methods (e.g., SMOTE) (Vassallo et al., 2021). The following sections discuss these standard methods for data sampling in anti-money laundering research and consider a novel approach for data sampling to be applied in this context, Generative Adversarial Networks (GAN).

### 2.3.1 Under- and Oversampling

A basic method to tackle imbalance is under or oversampling the dataset. Undersampling refers to dropping samples from the majority class, whereas oversampling duplicates samples from the minority class (Barandela et al., 2004). As oversampling duplicates existing observations, this method holds the risk of overfitting (Zhang & Trubey, 2019). The generated data might include instances where the majority and minority classes are similar, which results in models that become extremely complicated in classification tasks (Huang et al., 2021). As a result, Huang et al. (2021) argue models trained on oversampled data possibly perform equivalent or worse compared to models trained on imbalanced data.

Furthermore, Zhang and Trubey (2019) note that oversampling will increase the size of the dataset, which demands increased computational power to train the models. On the other hand, undersampling the dataset will be less computationally demanding. However, a disadvantage of undersampling is that it may cause valuable information to be removed, as it only captures a subset of the available data (Zhang & Trubey, 2019). Drummond and Holte (2003) research the effectiveness of both sampling methods and the interaction with a decision tree classifier on four different datasets. Findings suggest that undersampling beats over-sampling, as undersampling obtains reasonable sensitivity to variations in misclassification costs and class distributions. In contrast, oversampling was inefficient and frequently resulted in no changes in performance measures (in the case of default settings) (Drummond & Holte, 2003).

Zhang and Trubey (2019) researched under and oversampling in a money laundering detection context. Their findings suggest that the ANN, SVM, and R.F. models benefit from both sampling methods (Zhang & Trubey, 2019). Furthermore, the models show similar performance for under- and oversampling, except for the D.T. and SVM, as these models are more sensitive to event rate changes for oversampling than undersampling.

### 2.3.2 Synthetic Minority Oversampling Technique

The Synthetic Minority Oversampling Technique (SMOTE) was introduced by Chawla et al. (2002) to develop classification models on datasets with class imbalances. Contrary to the previously discussed oversampling technique, which duplicates existing samples from the minority class, SMOTE is an augmentation technique that synthesizes new samples based on the minority class, applying a K-nearest neighbors approach (Chawla et al., 2002). As a result, the overfitting problems induced by oversampling are minimized while there is no loss of important information. Nevertheless, a drawback is that it could increase data noise and can be ineffective in high-dimensional settings. Furthermore, a critique on SMOTE is that it occasionally adds minority-class samples identical to majority-class ones (He & Garcia, 2009). However, applying SMOTE on imbalanced datasets may modify the model to produce fewer false negatives but more false positives (Vassallo et al., 2021). As a result, SMOTE can result in higher recall at the expense of decreased accuracy.

### 2.3.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a novel type of sampling that leverages the power of neural networks for generative sampling. Initially proposed by Goodfellow et al. (2014), GANs produce realistic data by challenging two separate neural networks against one another in a zero-sum game (Wang et al., 2017). GANs have recently gained attention due to their capacity to produce accurate results based on limited training data. The networks consist of a discriminator and a generator trained concurrently (Wang et al., 2017). The discriminator's classifier attempts to distinguish between the true and false samples, whereas the generator produces new samples (see figure 8). Combined, the neural networks learn using an adversarial approach with the discriminator, improving the detection of false cases, whereas the generator attempts to deceive the discriminator with random variables (noise). The outcome is a model that generates unique and realistic samples. Conventional techniques often increase noise, resulting in overfitting as classification models attempt to fit this noise (Huang et al., 2021).

On the other hand, GANs attempt to disregard this noise and instead recognize the pattern of the minority class. For this reason, GANs are expected to have a lower probability of leading to overfitting and thus are expected to boost the performance of classification models (Huang et al., 2021). However, one of the drawbacks of GANs is the difficulty of training the model, as the discriminator and generator are continuously contending with one another, possibly leading to slower and unstable training (Huang et al., 2021).

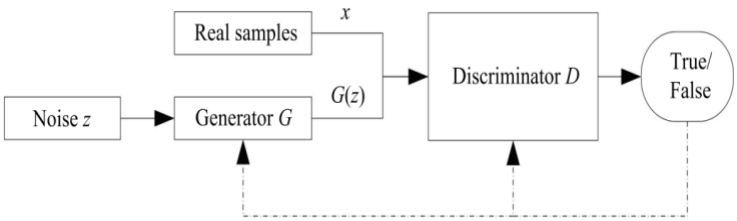


**Figure 8: Schematic Overview of the GAN Procedure (Wang et al., 2017)**

## 3. Data

One of the primary challenges in the research on money laundering detection is the lack of available real-world data. Fortunately, as discussed in the previous chapter, Elliptic Ltd. released the Elliptic data set used in this research. The Elliptic dataset is the largest publicly available dataset consisting of labeled transaction data on cryptocurrency transactions (Robinson, 2019). The dataset consists of 203,769 Bitcoin transactions worth six billion dollars when the data was published (Robinson, 2019). The transactions are labeled "licit", "illicit" or unlabelled based on publicly available data. This results in 4,545 illicit transactions (2% of the sample) and 42,019 licit transactions (21% of the sample). The rest of the sample is unlabeled, and for this research, these observations are dropped as the study focuses on supervised models. Next, the dataset is assumed to be imbalanced since it consists of 4,545 illicit and 42,019 licit transactions. This assumption is straightforward, as fraud detection is inherent to modeling a rare event and, thus, skewness in the data classes. The dataset is based on real-world transaction data, so features are masked and scaled to protect Bitcoin user privacy.

Figure 9 displays the ratio and moving average of illicit to licit transactions over the 49 timesteps of the dataset to gain insight on the prevalence of illicit transactions in the dataset. The 49 timesteps are spread over two weeks. Weber et al. (2019) note that the Elliptic dataset includes a closure of a dark marketplace after timestep 43, displayed by a significant drop in the volume of illicit transactions after timestep 43. Through this volume drop, combined with various time plot tests and the Augmented Dickey-Fuller test, Vassallo et al. (2020) raise concerns regarding potential concept drift (changes in statistical properties over time) in the dataset.

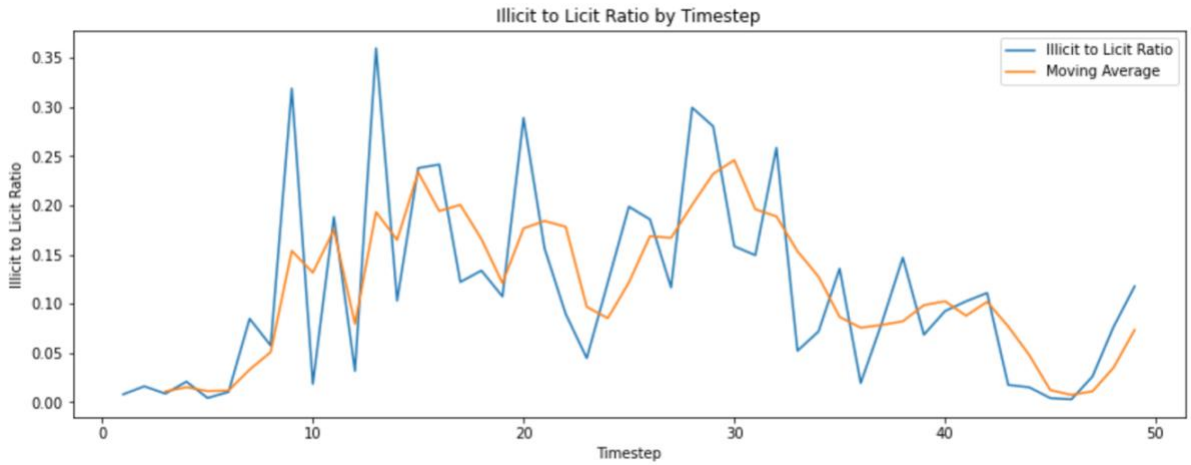


**Figure 9: Illicit to Licit Ratio by Timestep:** The illicit to licit ratio and moving average appear subject to changes over time, especially with a sharp decline after t=43 due to a black market shutdown. Changes in these metrics over time might indicate concept drift

Each observation in the Elliptic dataset has 166 features. The features are split into local features and aggregated features. The 94 features local features consist of local details on the transaction

such as transaction ID, timestamp, amount of inputs and outputs, service charge, the average quantity of Bitcoin sent or received by sender or receiver, as well as the average amount of transactions that come in or go out for the related addresses (Weber et al., 2019). All features, except the transaction ID, licit/illicit, and timestamp, were pre-scaled to zero mean and unit variance. In addition to the Local Features, the dataset consists of 72 aggregated features. These features are derived by combining the neighboring transactions' minimums, maximums, variances, and covariances for the same metadata, moving one step backward or forward relative to the transaction (Weber et al., 2019).

| Feature | Explanation | Measure |
|---|---|---|
| TxtID | Four to nine number transaction ID | 4 to 9 figure ID |
| Timestamp | Transactions are spread over two weeks | 1 to 49 |
| Class | Transactions are licit (1), illicit (2), or unknown | "1", "2", "unknown" |
| Local | Info on transaction, scaled to zero-mean and unit variance | -1 to 1 |
| Aggregated | Info on one-step neighbors, scaled to zero-mean and unit variance | -1 to 1 |

**Table 1: Overview of All Features in the Dataset**

Researchers were able to deanonymize a large part of the dataset by rescaling a few features and identifying those on the blockchain (Benzik, 2019). As a result, the masked transaction I.D. can be converted to transaction hashes on the blockchain and find real information on the transaction. For example, the transaction I.D. of the first transaction in the dataset is "230425980". Through scaling, the corresponding transaction hash can be found and the related transaction information can be traced using a blockchain tracker:

"74d9bb85c6bbc471c6e18f409d23c3ef1191725bdb90376fdff66fd31da41043"

Entering this hash in blockchair.com will yield that the amount transacted was 0.23 BTC, equalling around 102 USD at the time, with a transaction fee of 0.04 USD. Furthermore, this transaction took place on January 1, 2016, and the related sender and recipient addresses are given. Tracking and validating the transactions confirms the dataset is related to real-world transactions. Connecting the real-world features to the existing dataset falls out of the scope of this paper. However, the possibility of linking the real-world features to the current dataset opens doors for future research on feature importance in money laundering detection in Bitcoin. The transaction receipt is included in figure 10 in Appendix B for reference.

Next, an Uniform Manifold Approximation and Projection (UMAP) figure is created to visualize the high-dimensional data in a reduced, two-dimensional space, see figure 11. Creating the UMAP projection provides a way to detect potential patterns, clusters, and outliers. The projection below shows some potential clusters of illicit transactions, indicating clusters of comparable data points

in the original high-dimensional space. These clusters potentially show the presence of groups of illicit transactions with distinct behavioral patterns. These observations give insight into the underlying structure of the dataset and invite research on the detection of fraudulent transactions based on the patterns in the dataset in the following sections of this paper.

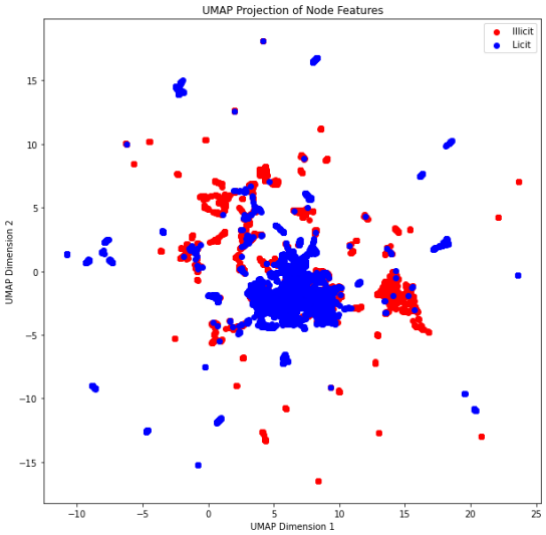


**Figure 11: UMAP Visualization: Clusters of Illicit Transactions Potentially Indicate Patterns in High-Dimensional Settings**

# 4. Methodology

This section first discusses the sampling methods applied to the datasets. The different sampling methods are compared in their performance of tackling the problem of dataset imbalance. Subsequently, graph data is introduced and node embeddings are added to the datasets to enable research on graph-based models. Then, the machine models are applied to the sampled datasets. At last, the evaluation metrics are discussed to assess the performance of the sampling methods and machine learning models.

## 4.1 Data Imbalance

Firstly, the sampling methods to tackle the imbalance of the dataset are discussed. I note that for the scope of this paper, I exclude the timestamps in the sampling process as preserving the temporal relationships while creating synthetic data yields challenges beyond the scope of this research. This opens doors for further research on online learning models, which I touch upon in the discussion section.

For this paper, I opt for undersampling instead of oversampling for further comparison of sampling methods. The rationale behind undersampling instead of oversampling is the high probability of overfitting in oversampling. As the dataset is highly imbalanced, duplicating the minority class towards a balanced dataset will most likely yield significant overfitting. Additionally, the undersampling method will decrease the size of the dataset and will be computationally more efficient. This is an advantage in the context of this research, as the classification models combined with various sampled datasets will require significant computational time. At last, I will also apply SMOTE sampling in the dataset, which outperforms random oversampling in imbalanced class distributions without leading to overfitting (Yen & Lee, 2009). Thus, for the undersampled dataset, I randomly undersample the data to have a balanced dataset of 4,500:4,500 licit to illicit observations.

For the train, validation, and test splits, an 80:10:10 ratio is used for all sampling methods. This train-test-validation split is a common approach in machine learning. As the dataset is large, I expect this split to allow for model evaluation and hyperparameter tuning while providing sufficient data for model training. With this split, I expect to have a sufficient amount of data to avoid overfitting and produce generalizable results through the use of a validation set. I anticipate that the sample is large enough to cover the underlying patterns of the high-dimensional datasets for the undersampling, SMOTE, and GAN methods. The training split consists of the balanced dataset, whereas the test and validation split set will follow the imbalanced data distribution. Using the balanced data for the training will ensure the models are not ‘lazy’ and will not simply classify all observations as the majority class. The balanced training set forces the model to understand the underlying patterns of licit and illicit transactions and classify the observations accordingly. However, if I use a balanced data distribution for the validation split, the model might be too specialized, and this will limit the potential to generalize

the results. Therefore, I choose the original, imbalanced distribution for the training and validation splits. Subsequently, the models will yield generalizable results.

Next to the undersampling method, I explore the performance of the SMOTE sampling method. The SMOTE method is commonly researched in the context of imbalanced data and research on money laundering detection (e.g., Ahmed, 2021; Vassallo et al., 2021), with the advantage of SMOTE reducing model overfitting when balancing datasets compared to randomly oversampling the minority class. I import the imblearn library to access the SMOTE sampling function to sample the training set. I split the majority class into 33,647:4,218:4,154 for train, test, and validation and the minority class into 3,604:438:503, respectively. After applying the SMOTE technique, the minority in training sets gets oversampled, so the balanced ratio becomes 33,647:33,647 in the training set.

At last, I compare the previous data sampling methods with the novel GAN sampling method. As the GAN sampling method can learn patterns from the minority class to generate realistic synthetic data points, I expect this method to produce superior results and reduce overfitting. To perform the GAN data augmentation, I apply the GAN pipeline as developed in the paper by Huang et al. (2021). I use six layers to build the GAN generator and seven layers to build the GAN discriminator. Furthermore, I use the selu activation in all hidden layers and sigmoid for the output layers. This activation function has led to better training convergence and generalization performance (Klambauer et al., 2017). I apply the same train, test, and split ratio and balanced training dataset as the SMOTE sampling method. This results in 33,647:4,218:4,154 and 3,604:438:503 splits for the majority and minority classes, respectively. After applying the GAN framework to sample the dataset, the balanced training dataset consists of 33,647:33,647 licit to illicit observations.

| | Undersampling | | SMOTE | | GAN | |
|---|---|---|---|---|---|---|
| Split | Licit | Illicit | Licit | Illicit | Licit | Illicit |
| Train | 3.600 | 3.600 | 33.647 | 33.647 | 33.647 | 33.647 |
| Test | 455 | 455 | 4.218 | 438 | 4.218 | 438 |
| Validation | 445 | 445 | 4.154 | 503 | 4.154 | 503 |
| Total | 4.500 | 4.500 | 42.019 | 34.588 | 42.019 | 34.588 |

**Table 2: Overview of Train, Test and Validation Splits per Sampling Method**

### 4.2 Graph Data

The edge list provided by Elliptic Ltd. was utilized in conjunction with the *NetworkX* library to create a graph for the Elliptic dataset. I use this approach to retrieve the graph data for the undersampled dataset. However, to generate synthetic data through SMOTE and GAN, I take the k-nearest neighbors

approach to create nodes and edges for the sampled data. I use the k-nearest neighbors approach over the Euclidean distance method for computational efficiencies. This approach slightly limits the potential of the graph network, as it limits the data on the relative distance of the nodes. Future research can explore different approaches' effects on accuracy and computational times in creating edge lists. In the realm of money laundering detection in the real world, there is a trade-off between model accuracy and computational time that holds significant value.

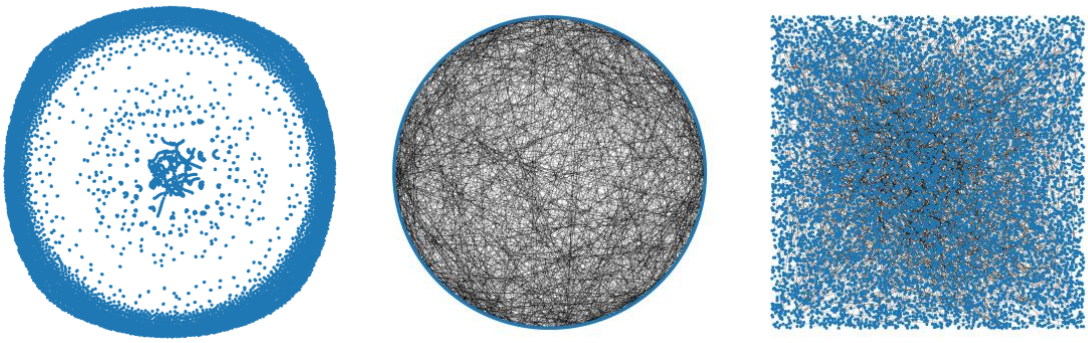

**Figure 12: Visualizations of the Graph Network Retrieved From the Elliptic Dataset**

### 4.3 Node Embeddings

As I am interested in the power of graph data and the potential of models that leverage this graph data, I take into account the embedded relationship between the observations. The dataset does not provide the node embeddings (NE). I use the graph networks I created previously for the undersampled, SMOTE, and GAN datasets to generate the node embeddings. I use the *node2vec* library that Grover and Leskovec (2016) proposed to generate the node embeddings. I develop an extra dataset with 64 enhanced node embedding features for each sampling method.

At last, I separate two categories of datasets. The data consists of local features, one-step forwards, and one-step backwards aggregated features. I create LF (Local Features) sets, consisting of only the first 94 features, and AF (All Features) sets consisting of all features (local and aggregated). This results in four categories of datasets for each sampling method: *LF, LF + NE, AF, AF + NE*.

### 4.4 Classification Models

This paper uses seven classification models to analyze their performance on previously acquired datasets. Specifically, I first apply the Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine models. To implement these models, I access the scikit-learn (sklearn) library and import the relevant functions, including *LogisticRegression* (LR), *DecisionTreeClassifier* (DT), *RandomForestClassifier* (RF), and *SVC* (SVM). Additionally, I apply a gradient-boosted decision

tree model in this study. To implement this model, I import the XGBClassifier from the *xgboost* library. Finally, I implement both the GNN and GBDT + GNN pipeline as proposed by Ivanov and Prokhorenkova (2021) and published by Ivanov (2021).

### 4.5 Evaluation Metrics

I employ a range of evaluation metrics to evaluate the discussed sampling methods and classification models. i access the *sklearn* library to get the performance measure for the models in terms of *accuracy*, *precision*, *recall*, *F1*, and *AUC-ROC* score. The F1 score will be the primary metric of interest, allowing for a balanced evaluation. As the F1 score is the harmonic mean of the precision and recall scores, focussing on this score will allow to assess the models' effectiveness in detecting illicit transactions while minimizing false positives. The focus on the F1 score as the primary metric is backed by previous studies in the given context (e.g., Vassallo et al., 2020; Weber et al., 2019). The formulas for the basic metrics are provided by the following mathematical equations, where the abbreviations are True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN):

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FN} \quad (2)$$

$$Recall = \frac{TP}{FP+FN} \quad (3)$$

$$F1 = \frac{2TP}{2TP+FP+FN} \quad (4)$$

Furthermore, I use the *Kullback-Leibler* (KL) divergence to measure the similarity of the distribution of the sampled and original datasets at the variable level, computed over the marginal probability mass functions of the sets (Goncalves et al., 2020). I implement the KL measure as proposed by Maklin (2019). The following mathematical equation denotes the KL divergence measure, where I denote the relative entropy from Q to P for the discrete probability distributions P and Q within similar sample space $X$ (MacKay, 2003):

$$D_{KL}(P||Q) = \sum_{x \epsilon X} P(x)\ ln\ (\frac{P(x)}{Q(x)}) \quad (5)$$

### 4.6 Tree-Structured Parzen Estimator Hyperparameter Tuning

Next, I apply hyperparameter optimization to the models to improve performance. I decide on Bayesian optimization using the Tree-Structured Parzen Estimator (TPE). The rationale for this variant of the Bayesian optimization is that the TPE estimator has shown strong performance in high dimensional search spaces and is in line with previous research on money laundering detection in Bitcoin (Xia et al., 2017; Vassallo et al., 2021). I access the *optuna* library to implement the default TPE algorithm. I use a sample of 5,000 observations to optimize the TPE. I focus on the F1 score as the main variable of interest and maximize this score as the objective function in line with Vassallo et al. (2021) and Weber et al. (2019).

## 5. Results

This section provides the evaluation of the classification and sampling models. I first compare the performance of logistic regression, decision tree, random forest, support vector machine, XGBoost, Graph Neural Network (GNN), and a GBDT+GNN model on three different datasets: the undersampled dataset, the SMOTE sampled dataset, and the GAN sampled dataset. The aim is to evaluate the impact of other sampling methods and machine learning models on the model performance based on the F1 scores regarding under- and overfitting. I also investigate the effect of hyperparameter tuning on the performance of the models. Figure 13 displays a brief overview of a selection of the models of interest. The total overview of the performance of all models on all sampling methods is displayed in figures 14 to 19 in Appendix C.

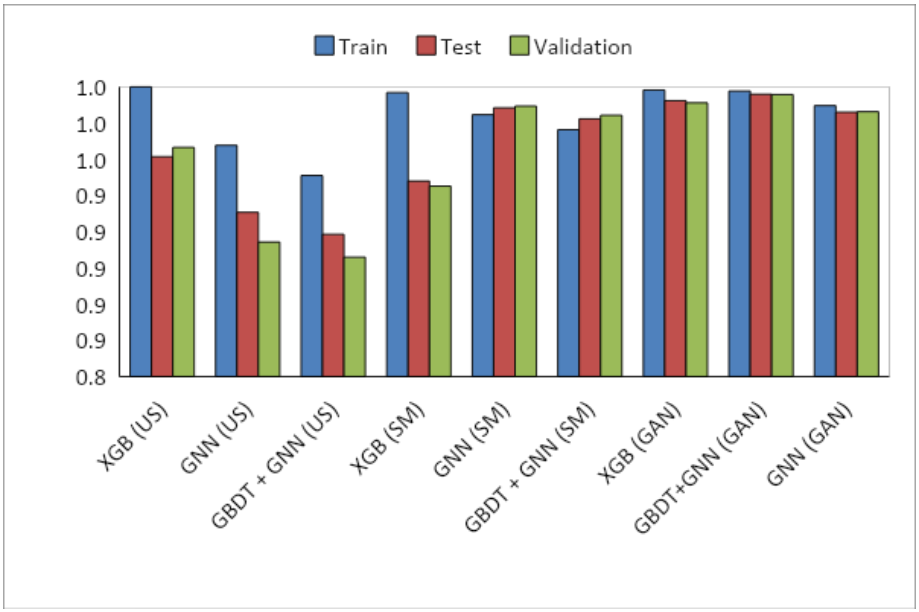


**Figure 13: Brief Overview of F1 Scores for Train, Test, and Validation set of the Tuned XGB, GNN, and GBDT + GNN Models with All Features and Node Embeddings on the Undersampled (US), SMOTE (SM), and GAN Datasets:** I find that models show signs of overfitting for the undersampled datasets. Applying SMOTE sampling significantly improves the signs of overfitting, however, the XGBoost model still shows signs of overfitting. The GAN sampling method shows the best performance and generates the most generalizable results across this selection of models and all models displayed in Appendix C

All models show signs of overfitting for the undersampled dataset in figure 14 in Appendix C, as the training F1 scores are higher than the test and validation scores (Subramanian and Simon, 2013). Tuning the hyperparameters does not significantly affect the overfitting and test and validation of the F1 scores. A potential explanation for the overfitting is the loss of information due to the undersampling process.

For the SMOTE sampled dataset displayed in figure 16 in Appendix C, I observe that the default settings of all models using the Local Features and All Features do not overfit. However, introducing Node Embeddings leads to significant overfitting across all models except for the XGB and GNN models with All Features, which perform well. Hyperparameter optimization slightly improves the performance of most models. However, the models performed on the Node Embedded set still show

overfitting. Hyperparameter tuning does seem to resolve most signs of overfitting in the Node Embedded All Features models for the RF, SVM, and GBDT+GNN.

For the GAN sampled dataset, I find that the models show slight to no overfitting with the default model settings, as displayed in figure 18 in Appendix C. However, the GNN model with All Features shows underfitting. Overall, the GAN sampling method delivers the best performance for the default settings across the sampling methods in default settings and thus generates the most generalizable results. After hyperparameter tuning, the underfitting of the GNN All Features model is resolved, resulting in the tuned GAN dataset showing the most robust and generalizable results across all sampling methods and with default and tuned hyperparameters. These results lead back to the research question, suggesting that the GAN sampling method is the most effective approach for addressing the class imbalance in the context of money laundering in Bitcoin transactions.

Furthermore, these results align with previous research on the GAN method, suggesting GAN produces unique and more realistic samples than conventional techniques (Huang et al., 2021). The results show that traditional techniques (Undersampling and SMOTE) tend to increase noise and result in overfitting. In contrast, the GAN method disregards this noise and recognizes the underlying patterns within the minority class.

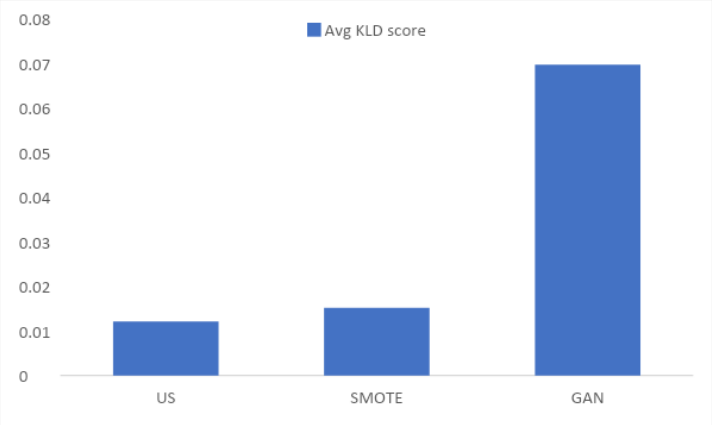


**Figure 20: Kullback-Leibler Divergence Measure Scores:** The results display a higher KL divergence for the GAN sampled than the US and the SMOTE methods. The GAN sampling method generates more 'realistic' samples; these samples resemble the original data but do not precisely replicate it. On the other hand, the US and SMOTE show a closer approximation to the original data

Figure 20 shows the Kullback-Leibler Divergence (KLD) measure results for the GAN, SMOTE, and Undersampled datasets compared to the original dataset. The GAN sampling method has a significantly higher KLD score than the SMOTE and Undersampling method. This indicates that the GAN sampling method generates samples with features that are substantially different from the original data distribution. Firstly, if the samples generated by the GAN deviate too far from the original data distribution, this might result in less stability and poor model performance. Secondly, the samples produced by the GAN may offer more variety to the training data, which might benefit the classification models and lead to better performance. This second suggestion aligns with the previous observations,

as displayed in figure 18 and 19 in Appendix C, where the GAN sampling method results in less overfitting and improved model performance. Combining the earlier observations on comparing the train, test, and validation performance of the sampling methods with the findings of the KLD measure, I find that the KLD measure confirms the suggestion on the solid performance of the GAN sampling method. Thus, I conclude the GAN sampling method produces the most 'realistic' samples, leading to less overfitting and most generalizable results compared to the SMOTE and Undersampling methods.

Next, I take a broader view of the evaluation metrics for the performance of the classification models. I provide a complete overview of the evaluation metrics of all models on the undersampling, SMOTE, and GAN datasets in tables 3, 4 and 5 in Appendix C, respectively. Below, table 6 gives a brief overview of the evaluation metrics of selected models and sampling methods. I include model performance before and after tuning, separated by a hyphen for completeness.

| | Model | Accuracy | F1 | Precision | Recall | ROC AUC | F1 MicroAVG |
|---|---|---|---|---|---|---|---|
| Undesampling | XGBoost (AF) | 0.979 - 0.981 | 0.979 - 0.982 | 0.976 - 0.983 | 0.983 - 0.980 | 0.979 - 0.981 | 0.979 - 0.981 |
| | GNN (NE AF) | 0.906 - 0.917 | 0.900 - 0.914 | 0.953 - 0.939 | 0.853 - 0.891 | 0.905 - 0.917 | 0.906 - 0.917 |
| | GNN + GBDT (NE AF) | 0.893 - 0.910 | 0.887 - 0.906 | 0.942 - 0.947 | 0.837 - 0.869 | 0.893 - 0.910 | 0.893 - 0.910 |
| SMOTE | XGBoost (AF) | 0.992 - 0.989 | 0.996 - 0.994 | 0.993 - 0.992 | 0.999 - 0.996 | 0.966 - 0.960 | 0.992 - 0.989 |
| | GNN (NE AF) | 0.968 - 0.981 | 0.982 - 0.989 | 0.988 - 0.986 | 0.977 - 0.993 | 0.931 - 0.928 | 0.968 - 0.981 |
| | GBDT + GNN (NE AF) | 0.821 - 0.972 | 0.891 - 0.984 | 0.992 - 0.988 | 0.809 - 0.981 | 0.874 - 0.933 | 0.821 - 0.972 |
| GAN | XGBoost (AF) | 0.988 - 0.986 | 0.993 - 0.992 | 0.987 - 0.986 | 0.999 - 0.999 | 0.946 - 0.939 | 0.988 - 0.986 |
| | GNN (NE AF) | 0.976 - 0.976 | 0.987 - 0.987 | 0.978 - 0.978 | 0.996 - 0.996 | 0.889 - 0.888 | 0.976 - 0.976 |
| | GBDT + GNN (NE AF) | 0.974 - 0.975 | 0.986 - 0.986 | 0.975 - 0.976 | 0.998 - 0.997 | 0.873 - 0.879 | 0.974 - 0.975 |

**Table 6: Brief Overview of Evaluation Metrics on a Selection of Models:** I find that the XGB model outperforms the Graph-based models in terms of F1 score across all sampling methods. Other relevant evaluation metrics confirm the XGB is the best classification model

For the undersampled dataset, I find that the XGBoost classifier performed best regarding F1 score across the Local Features, All Features, and with and without Node Embeddings compared to the other models. The GNN and GBDT+GNN models underperform compared to the other models in the undersampled dataset for Local Features and Local Features with Node Embeddings. This underperformance could stem from overfitting or a lack of data in the undersampled dataset. This dataset is relatively small, and these models require large amounts of data to learn meaningful representations.

For the SMOTE sampling method, I observe that the XGBoost model with All Features yields the highest F1 score. Furthermore, the XGBoost model with All Features outperformed all other models with All Features in terms of Accuracy, Precision, and AUC ROC score. Additionally, the other XGBoost models also show robust performance across all metrics.

For the GAN sampling method, I find that the XGBoost model outperformed all other models in terms of F1 score across Local Features, All Features, with and without Node Embeddings. Specifically, the best-performing model in terms of F1 score is the XGBoost with All Features.

Furthermore, the XGBoost outperforms the GBDT+GNN models across all evaluation metrics, even after introducing Node Embedding on the All Features set. This observation is remarkable, as the GBDT+GNN should be able to leverage the Graph network in this set. However, I also find the GNN and GBDT+GNN model outperforms the Logistic Regression, demonstrating the advantages of a graph-based approach over a method that disregards the graph data structure, which is consistent with earlier work by Weber et al. (2019).

Another observation I make is that the LR, DT, SVM, and XGBoost models do not tend to improve upon adding Node Embeddings to the dataset or even perform worse. This observation is in line with my intuition, as these models are not developed to handle graph-structured data precisely and are thus unable to accurately capture the connections and patterns between a graph's nodes. On the other hand, the graph-based models, the GNN and GBDT+GNN, are built to perform on graph-structured data and benefit from the graph's structure and node representations. However, the GNN model does not improve on adding Node Embeddings to the Local Feature set. The slight reduction in model performance could be due to the complexity of the model, data sparsity, or feature engineering. However, I find model improvements upon adding Node Embeddings for the All Features set of the GNN model and the Local and All Features set for the GBDT+GNN model.

To further compare the performance of the XGBoost All Features model versus the GBDT+GNN All Features with Node Embeddings, I display the confusion matrices of both models side by side in figure 21. In the confusion matrices, the X-axis (Actual Values) and Y-axis (Predicted Values) display either a 1 (Positive) or 0 (Negative). This results in TP=4211, FP=56, FN=7, and TN=333 for the XGBoost model and TP=4206, FP=105, FN=12, and TN=333 for the GBDT+GNN model. From these confusion matrices, I find the XGBoost outperformed the GBDT+GNN model in correctly predicting True Positives and True Negatives while also predicting fewer False Positives and False negatives. I conclude the XGBoost model is more effective across the board than the GBDT+GNN model.

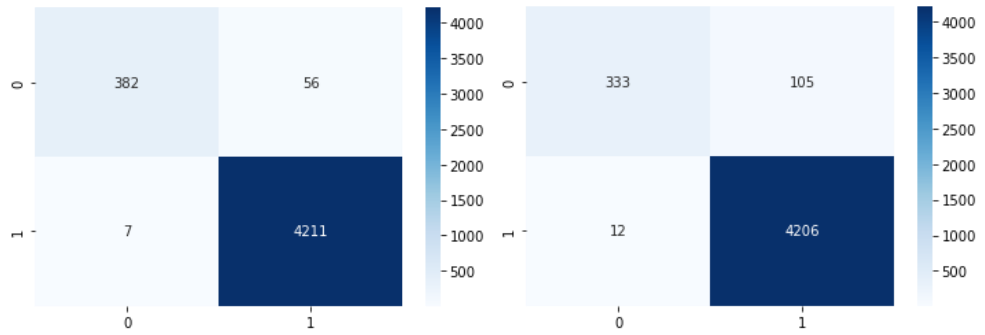


**Figure 21: Confusion Matrices for Tuned XGBoost All Features (Left) and Tuned GBDT + GNN All Features with Node Embeddings (Right) Using GAN Sampling**

Next, I discuss the potential concept drift as indicated by figure 9 in the Data section. To handle the possible concept drift, I take on two approaches. I introduce both "offline" learning with a temporal split and "prequential" or "online" learning models to see if the model performance is affected and potentially deteriorated by concept drift.

I apply offline learning models with a temporal split to the dataset. Through a temporal split, I test the model's ability to generalize new data it has not seen before. The likelihood of potential concept drift increases when the model is significantly less effective on the testing set than the training set. In figure 9 in the Data section, the black market shutdown resulted in a drop in the illicit to licit transaction ratio after timestep 43. Furthermore, I find that the moving average is decreasing after timestep 34. As a result, I decided to split the dataset into training and testing as follows: for the training set, I include all observations based on $t < 34$, and for the testing set, I include all observations for $t <= 34$. I test the models on the imbalance dataset, the SMOTE, and the GAN sampled dataset. I display the results for the basic models and tuned basic models in terms of F1 score in Appendix D, figures 22 and 23. For the default models, I find slight overfitting across all models and datasets. After tuning, I find that this overfitting is reduced, and all models only show minor signs of overfitting. The only model showing significant overfitting for both default and tuned settings is the DT model on the SMOTE dataset. These results, in combination with the effects on the train, test, and validation sets displayed in Appendix C, figures 13 to 18, appear to show that the performance of the models is sound and does not show significant signs of concept drift.

Subsequently, I perform an online learning model to inspect concerns regarding concept drift further. I opt for the XGBoost model for online learning, as this was the best-performing model in the previous experiments.

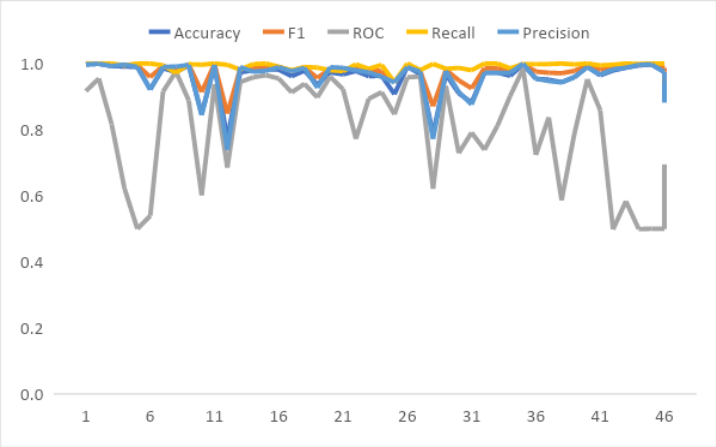


**Figure 24: Prequential Evaluation of Online Learning Model:** The ROC curve shows high variance, with sharp declines through time. This might indicate the model is overfitting or is not robust to potential concept drift in the data.

Here, the XGBoost model is trained continuously as incoming data is added to the model over time. I use prequential evaluation to assess the model sequentially. The results are displayed in figure 24. From the results, I find a high variance in the ROC curve. This high variance in the ROC curve might happen due to various reasons. Firstly, the model may be overfitting on the data. I use the imbalanced dataset for online learning, as it is hard to perform synthetic sampling on time-stamped data. This potentially results in a bias towards the majority class, and the model captures irrelevant patterns in the data. However, it is also possible that the model is not robust to potential concept drift in the data, and the distribution of the data indeed changes over time, leading to poor generalization performance. This leaves the door open for further research on regularization techniques, online learning models, and concept drift detection in the context of money laundering detection in Bitcoin transactions

## 6. Conclusion and Discussion

In summary, I aimed to assess the effects of novel sampling techniques and machine learning models on the effectiveness of classification models for detecting money laundering in Bitcoin transactions. In this paper, I investigated the performance of three sampling methods: Undersampling, SMOTE, and the novel GAN method in concert with the effectiveness of seven classification models, including Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, XGBoost, Graph Neural Network, and the novel Gradient Boosted Decision Tree + Graph Neural Network model. The degree of under- and overfitting and the F1, Accuracy, Precision, and Recall scores of the models were compared. The effect of hyperparameter tuning using the Tree Parzen Estimator (TPE) on the models' performance was also investigated.

The findings demonstrate that the Undersampling method caused overfitting for all classification models. The overfitting improved when applying SMOTE sampling, although the basic models performed on the Node Embedded sets continued to exhibit signs of overfitting. The GAN sampling technique outperformed the other sampling techniques in terms of performance and generalizability. Thus, I discovered that the GAN sampling method shows potential as the best strategy for tackling class imbalance in the context of money laundering in Bitcoin transactions. The Kullback-Leibler Divergence (KLD) measure confirmed that the GAN sampling method produced the most 'realistic' samples, leading to less overfitting and more generalizable results than the SMOTE Undersampling methods. The research findings indicate that where the GAN method detects underlying patterns within the minority class, conventional strategies (Undersampling and SMOTE) tend to enhance noise and lead to overfitting.

Furthermore, I demonstrate that the XGBoost model with All Features is the best-performing model across every sampling technique regarding the F1 score. Additionally, although the GBDT+GNN models were expected to benefit from the graph structure of the data, the results for the GAN sampled data show that the XGBoost model surpasses the GBDT+GNN model in every evaluation criterion. Nonetheless, the results show that the GNN and GBDT+GNN models perform better than the Logistic Regression model, highlighting the potential benefits of a graph-based methodology. All in all, these findings emphasize the power of the XGBoost model in the context of money laundering detection and advance the understanding of how other machine learning models perform for GAN sampling

Additionally, I explored the possibility of concept drift and took on two approaches: offline learning with a temporal split and online learning models. The offline learning experiments suggest that the models' performance is sound and does not show significant signs of concept drift. However, the prequential evaluation for the XGBoost model revealed high variance in the ROC curve, which

highlights the need for further research on regularization techniques, online learning models, and concept drift detection in the context of money laundering detection in Bitcoin transactions.

Overall, I provide insights into how various sampling techniques and machine learning models affect classification model performance for identifying money laundering in Bitcoin transactions. The observations can guide the development of better strategies for dealing with class inequality in this situation.

While I shed light on essential aspects in the current state of the literature on money laundering detection in Bitcoin transactions, there are still ample avenues for future research to be explored in this context. The results show that adding Aggregated Features and Node Embeddings provided diminishing returns in terms of model performance, complexity, and computational times due to the nature of the large dataset with high-dimensional features. As a result, I suggest adding dimensional reduction techniques such as PCA, t-SNE, and LDA to reduce the dimensionality of the data. These dimensional reduction techniques can help reduce overfitting, speed up training, and improve performance, which are essential aspects of money laundering detection. Another approach is to use the reduced embeddings in Graph Networks for tasks such as node classification. However, it's important to note that dimensional reduction techniques can result in a loss of information. Nonetheless, I believe that dimensional reduction techniques provide an exciting avenue for future research in money laundering detection in Bitcoin transactions using traditional and graph-based models.

Furthermore, I suggest further research into combining online learning models with GAN sampling to tackle data imbalance, as the online learning models can adapt to changing patterns in the data over time. As online learning models update predictions in real-time, these models will be less prone to the potential concept drift in time-stamped data. I expect these models to be instrumental in this context, as a rapidly changing environment characterizes cryptocurrencies. Additionally, these online learning models efficiently handle large volumes of data, which is highly relevant in this context. Combining GAN sampling with online learning can improve overall model performance through continuously updated models trained in concert with synthetically balanced samples. This could reduce the potential concept shift that might be present in changes in the transaction data over time.

Moreover, this paper focused on classification and sampling models for money laundering detection on the transaction level. Further research could extend the findings by leveraging fraud detection at the transaction level with fraud detection at the account level and seek to compare and combine detection across these two dimensions.

# 7. Bibliography


Ahmed, A. (2021). Anti-money laundering recognition through the gradient boosting classifier. Academy of Accounting and Financial Studies Journal, 25(5).

Alarab, I., & Prakoonwit, S. (2022). Graph-Based LSTM for Anti-money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data. *Neural Processing Letters*, 1-19.

Alarab, I., Prakoonwit, S., & Nacer, M. I. (2020). Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies* (pp. 23-27).

Barandela, R., Valdovinos, R. M., Sánchez, J. S., & Ferri, F. J. (2004). The imbalanced training sample problem: Under or over sampling?. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004. Proceedings* (pp. 806-814). Springer Berlin Heidelberg.

Bartoletti, M., Pes, B., & Serusi, S. (2018, June). Data mining for detecting bitcoin ponzi schemes. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (pp. 75-84). IEEE.

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3), 1937-1967

Benzik, A. (2019, December 7). Deanonymization of Elliptic dataset transactions. Retrieved from https://habr.com/ru/post/479178/

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, *24*.

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*(2), 197-227.

Bitcoin ATMs in the Netherlands. (2023, February 24). *CoinATMRadar*. https://coinatmradar.com/country/150/bitcoin-atm-netherlands/

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

Buchanan, B. (2004). Money laundering—a global obstacle. *Research in International Business and Finance*, *18*(1), 115-127.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321-357.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Chen, Z., Van Khoa, L. D., Teoh, E. N., Nazir, A., Karuppiah, E. K., & Lam, K. S. (2018). Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems*, 57(2), 245-285.

Chow, A. R. (2022, February 10). Inside the Chess Match That Led the Feds to $3.6 Billion in Stolen Bitcoin. *Time.* Retrieved from https://time.com/6146749/cryptocurrency-laundering-bitfinex-hack/

Christodoulou, E., Ma, J., Collins, G. S., Steyerberg, E. W., Verbakel, J. Y., & Van Calster, B. (2019). A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of clinical epidemiology*, 110, 12-22.

Colladon, A. F., & Remondi, E. (2017). Using social network analysis to prevent money laundering. Expert Systems with Applications, 67, 49-58.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273-297.

De Oña, R., Eboli, L., & Mazzulla, G. (2014). Key factors affecting rail service quality in the Northern Italy: a decision tree approach. *Transport*, 29(1), 75-83.

Deutsch, A., & Meijer, B. H. (2021, April 19). ABN Amro to settle money laundering probe for $574mln. *Reuters*. Retrieved from https://www.reuters.com/business/abn-amro-settle-money-laundering-probe-574-million-2021-04-19/

Department of Justice Office of Public Affairs. (2022, February 8). Two arrested for alleged conspiracy to launder $4.5 billion in stolen cryptocurrency.

Dreżewski, R., Sepielak, J., & Filipkowski, W. (2015). The application of social network analysis algorithms in a system supporting money laundering detection. *Information Sciences*, *295*, 18-32.

Drummond, C., & Holte, R. C. (2003, August). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II* (Vol. 11, pp. 1-8).

Duyne, P. van, Lampe, K. von, & Newell, J. (2003). *Criminal Finances and Organising Crime in Europe*. Enfield Publishing & Distribution Company.

European Central Bank. (2022, April 25). *For a few cryptos more: the Wild West of crypto finance*. European Central Bank. https://www.ecb.europa.eu/press/key/date/2022/html/ecb.sp220425%7E6436006db0.en.html

Financial Action Task Force (2006). Trade-Based Money Laundering. Retrieved from https://www.fatf-gafi.org/en/publications/Methodsandtrends/Trade-basedmoneylaundering.html

Financial Action Task Force (2012). International Standards on Combating Money Laundering and the Financing of Terrorism & Proliferation. Retrieved on February 15 2023, from https://www.fatf-gafi.org/content/dam/recommandations/pdf/FATF+Recommendations+2012.pdf.coredownload.inline.pdf

Financial Action Task Force (2020). Virtual Assets: What? When? How? Retrieved from https://www.fatf-gafi.org/media/fatf/documents/bulletin/fatf-booklet_va.pdf

Financial Action Task Force (2021). Annual Report 2020 - 2021. Retrieved from https://www.fatf-gafi.org/media/fatf/documents/brochuresannualreports/Annual-Report-2020-2021.pdf

Foley, S., Karlsen, J. R., & Putniņš, T. J. (2019). Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies?. *The Review of Financial Studies*, *32*(5), 1798-1853.

Global Cryptocurrency Market Charts. (2022). CoinMarketCap. Retrieved from https://coinmarketcap.com/charts/

Goncalves, A., Ray, P., Soper, B., Stevens, J., Coyle, L., & Sales, A. P. (2020). Generation and evaluation of synthetic patient data. *BMC medical research methodology*, 20(1), 1-40.

Golbeck, J. (2013). *Analyzing the social web*. Newnes.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu B., Warde-Farley, D., Ozair S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*. pp. 2672–2680.

Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864).

Haber, S., & Stornetta, W. S. (1990). How to time-stamp a digital document. *Conference on the Theory and Application of Cryptography* (pp. 437-455). Springer, Berlin, Heidelberg.

Harris, D. A., Pyndiura, K. L., Sturrock, S. L., & Christensen, R. A. (2022). Using real-world transaction data to identify money laundering: Leveraging traditional regression and machine learning techniques. *STEM Fellowship Journal*, 7(1), 21-32.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9), 1263-1284.

Healy, L. M. (2006). Logistic regression: An overview. *Eastern Michigan College of Technology*.

Heidari, N., Hedegaard, L., & Iosifidis, A. (2022). Graph convolutional networks. *Deep Learning for Robot Perception and Cognition* (pp. 71-99). Academic Press.

Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

Huang, Y., Fields, K. G., & Ma, Y. (2022). A tutorial on generative adversarial networks with application to classification of imbalanced data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(5), 543-552.

Hyman, M. (2015). Bitcoin ATM: A Criminal's Laundromat for Cleaning Money. *Thomas L. Rev.*, 27, 296.

Ivanov, S., & Prokhorenkova, L. (2021). Boost then convolve: Gradient boosting meets graph neural networks. *arXiv preprint arXiv:2101.08543*.

Ivanov, S. (2021). Boosted Graph Neural Networks. GitHub. https://github.com/nd7141/bgnn

Jullum, M., Løland, A., Huseby, R. B., Ånonsen, G., & Lorentzen, J. (2020). Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control*, *23*(1), 173-186.

Keriven, N., & Peyré, G. (2019). Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in neural information processing systems*, 30.

Kumar, A., Das, S., Tyagi, V., Shaw, R. N., & Ghosh, A. (2021). Analysis of classifier algorithms to detect anti-money laundering. *Computationally Intelligent Systems and their Applications*, 143-152.

Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., & Koh, E. (2019). Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6), 1-25.

Levi, M. (2002). Money laundering and its regulation. *The Annals of the American Academy of Political and Social Science*, *582*(1), 181-194.

Lin, Y.J., Wu, P.W., Hsu, C.H., Tu, I.P., & Liao, S.W. (2019). An evaluation of bitcoin address classification based on transaction history summarization. *In 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (pp. 302-310). IEEE.

Lv, L. T., Ji, N., & Zhang, J. L. (2008, August). A RBF neural network model for anti-money laundering. In *2008 International conference on wavelet analysis and pattern recognition* (Vol. 1, pp. 209-215). IEEE.

MacKay, D. J.. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Maklin, C. (2019). KL Divergence Python Example - Towards Data Science. *Medium*. https://towardsdatascience.com/kl-divergence-python-example-b87069e4b810

Maron, H., Fetaya, E., Segol, N., & Lipman, Y. (2019, May). On the universality of invariant networks. *In International conference on machine learning* (pp. 4363-4371). PMLR.

Merkle, R. C. (1978). Secure communications over insecure channels. *Communications of the ACM*, *21*(4), 294-299.

Mitchell, T. M. (1997). *Machine learning* (Vol. 1, No. 9). New York: McGraw-hill.

Monamo, P. M., Marivate, V., & Twala, B. (2016, December). A multifaceted approach to bitcoin fraud detection: Global and local outliers. *In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 188-194). IEEE.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.

Nissenbaum, H. (1999). The meaning of anonymity in an information age. *The Information Society*, 15(2), 141-144.

Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, 59(3), 183-187.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81-106.

Raiter, O. (2021). Applying Supervised Machine Learning Algorithms for Fraud Detection in Anti-Money Laundering. Journal of Modern Issues in Business Research, 1(1), 14-26.

Ray, S. (2019, February). A quick review of machine learning algorithms. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)* (pp. 35-39). IEEE.

Richardson, B., Williams, D., & Mikkelsen, D. (n.d.). Network analytics and the fight against money laundering.McKinsey & Company. https://www.mckinsey.com/industries/financial-services/our-insights/banking-matters/network-analytics-and-the-fight-against-money-laundering

Robinson, T. (2019). How to Combat Financial Crime in Cryptocurrencies. Retrieved from https://www.elliptic.co/blog/elliptic-dataset-cryptocurrency-financial-crime

Subramanian, J., & Simon, R. (2013). Overfitting in prediction models–is it a problem only in high dimensions?. *Contemporary clinical trials*, 36(2), 636-641.

Şahin, Y. G., Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines. Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 (Volume 1) (pp. 442-447).

Sanchez-Lengeling, B., Reif, E., Pearce, A., & Wiltschko, A. B. (2021). A gentle introduction to graph neural networks. *Distill*, 6(9), e33.

Seagrave, S. (2010). *Lords of the Rim 2010: The Invisible Empire of the Overseas Chinese*. CreateSpace Independent Publishing Platform.

Senator, T. E., Goldberg, H. G., Wooton, J., Cottini, M. A., Khan, A. U., Klinger, C. D., ... & Wong, R. W. (1995, August). The FinCEN Artificial Intelligence System: Identifying Potential Money Laundering from Reports of Large Cash Transactions. In *IAAI* (pp. 156-170).

Silva Ramalho, D., & Igreja Matos, N. (2021, September). What we do in the (digital) shadows: Anti-money laundering regulation and a bitcoin-mixing criminal problem. In *ERA Forum* (Vol. 22, No. 3, pp. 487-506). Berlin/Heidelberg: Springer Berlin Heidelberg.

Storm, A. (2013). Establishing the link between money laundering and tax evasion. *International Business & Economics Research Journal (IBER)*, *12*(11), 1437-1450.

Sudjianto, A., Nair, S., Yuan, M., Zhang, A., Kern, D., & Cela-Díaz, F. (2010). Statistical methods for fighting financial crimes. *Technometrics*, 52(1), 5-19.

Tang, J., & Yin, J. (2005, August). Developing an intelligent data discriminating system of anti-money laundering based on SVM. In 2005 *International conference on machine learning and cybernetics* (Vol. 6, pp. 3453-3457). IEEE.

Thabtah, F., Hammoud, S., Kamalov, F., & Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513, 429-441.

Vanderford, R. (2022, July 20). Banks Turn to A.I. to Help Dodge Enforcement Spotlight. *Wall Street Journal*. https://www.wsj.com/articles/banks-turn-to-ai-to-help-dodge-enforcement-spotlight-11658309401

Vassallo, D., Vella, V., & Ellul, J. (2021). Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies. *S.N. Computer Science*, 2, 1-15.

Vatra, N. (2009). Public key infrastructure overview. Scientific Studies and Research, 19(2).

Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., & Wang, F. Y. (2017). Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica*, *4*(4), 588-598.

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*.

Wong, E. (1986). Retrieving dispersed data from SDD-1: A system for distributed databases. In *Distributed systems, Vol. II: distributed database systems* (pp. 227-245).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1), 4-24.

Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications*, 78, 225-241.

Yen, S. J., & Lee, Y. S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, *36*(3), 5718-5727.

Zhang, Y., & Trubey, P. (2019). Machine learning and sampling scheme: An empirical study of money laundering detection. Computational Economics, 54(3), 1043-1063.

Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4), 352-375.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. *A.I. Open*, 1, 57-81.

Zuurmond, I. (2022). Banken controleren op witwassen: schending van je privacy? Consumentenbond.nl. Retrieved from https://www.consumentenbond.nl/betaalrekening/banken-controleren-witwassen-privacy

## Appendix A: Machine Learning Methods Explained

This section gives an overview of the standard machine learning methods used in this paper.

### Logistic Regression

Logistic Regression (L.R.) is a supervised machine learning method that is used for classification problems. Logistic regression is a powerful tool for predicting the probability of binary outcome based on independent variables (Healy, 2006). Compared to a linear regression that produces a straight line, the logistic regression produces a curve that represents values between zero and one (Hosmer Jr et al., 2013). The following equation represents the logistic regression in mathematical form:

$$p = \frac{e^{\alpha+\beta_n X}}{1 + e^{\alpha+\beta_n X}}$$

### Decision Trees

The Decision Tree (D.T.) method is a type of supervised learning that is also used to classify data in different categories. The decision tree works by drawing a flowchart-like structure of decisions and their outcomes, based on a set of input data. The data is split in nodes, and the decisions in leaves (Ray, 2019). The tree starts with the root node, which branches out in different paths, matching potential outcomes. Every node defines an attribute, while each branch that subsides from that node corresponds to one of the possible values for that attribute (Mitchell, 1997). In order to classify an instance, one starts at the root node, tests the specified attribute and then moves down the tree branch accordingly. Subsequently, one repeats this process for the subtree rooted at the new node. Below, an example of a decision tree by Quinlan (1986), known for his extensive contributions to the development of decision trees:

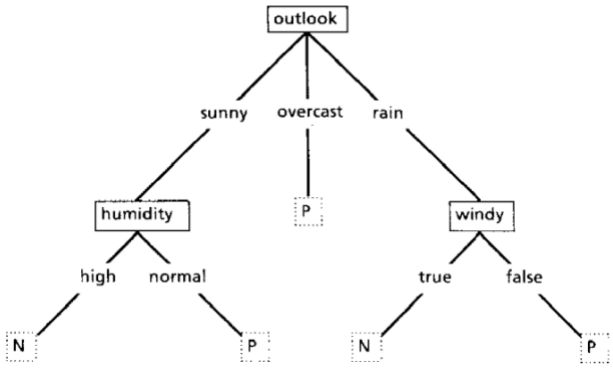


**Figure 3: Decision Tree that Classifies Saturday Mornings if They are Suitable to Play Tennis (Quinlan, 1986)**

Entropy is often used as a splitting criterion in classification problems, an application of the source coding theorem by Shannon (2001). The following formula gives entropy at every internal node of the decision tree:

$$E = -\sum_{i=1}^{c} p_i \times log(p_i)$$

Maximizing this formula attains the most information possible at each node in the decision tree. Here, c is the number of distinct classes, and $p_i$ is the prior probability for each respective class.

**Random Forests**

Random Forests (R.F.) are a type of ensemble method that combines predictions of multiple decision trees to create a more accurate model. Random forests are created by randomly selecting a subset of features from the given dataset and building a decision tree based on these features (Breiman, 2001). This process is repeated multiple times, and the output of each tree is subsequently combined to produce a single prediction. As the random forest is built on multiple trees, each of which is trained on a different subset of features, the model is less prone to overfitting compared to decision trees (Biau and Scornet, 2016). Furthermore, by looking at the relative importance of each feature and identifying which has most significance in predicting output, the model offers feature importance.

**Support Vector Machine**

The Support Vector Machine (SVM) was introduced by Cortes and Vapnik (1995) for two-group classification problems. SVM maps data points from a high-dimensional space into a two–dimensional space. SVM tries to find the optimal two-dimensional space, the hyperplane, to best separate the classes. The goal is to find a decision boundary that maximizes the margin between classes (Chen et al., 2018). The following equation defines the hyperplane that separates the data:

$$w^T x + b = 0$$

Where $w^T$ is the transpose of coefficient matrix w, with $w \in F$ feature space. Furthermore, x is the observed vector and $b \in R$ real numbers (Chen et al., 2018).

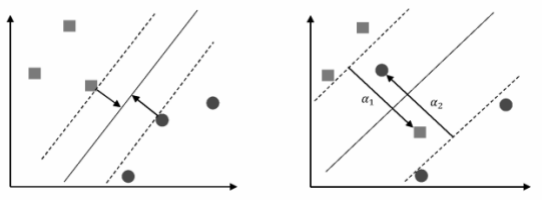


**Figure 4: Example of a Simple 2-class Support Vector Machine (Raiter, 2021)**

# Appendix B: Sample Transaction

BLOCKCHAIR

info@blockchair.com
https://blockchair.com

**TRANSACTION RECEIPT** **BITCOIN**

**TRANSACTION IDENTIFIER:**

74d9bb85c6bbc471c6e18f409d23c3ef1191725bdb90376fdff66fd31da41043

**TRANSACTION TIMESTAMP:**

2016-01-01 04:24 (UTC)

**CONFIRMED TRANSACTION:**

Included in block: #391206 on the Bitcoin blockchain

**SENDERS (INPUTS):**

| # | SENDER | VALUE (BTC) | VALUE (USD) |
|---|---|---|---|
| 0 | 1Goo39FXgugAPVp1pUiy7DucG2EBBe6ESH | 0.24003068 | 102.55 |
| | | TOTAL: 0.24003068 BTC \| 102.55 USD | |

**RECIPIENTS (OUTPUTS):**

| # | RECIPIENT | VALUE (BTC) | VALUE (USD) |
|---|---|---|---|
| 0 | 1Jp3oz2UueZ4aUzq8aTd1kLBETeZsZqhCF | 0.23766510 | 101.54 |
| 1 | 1DgNieAu87xxvXjyGuuVQQSFfU94Bevssd | 0.00226558 | 0.97 |
| / | * * * MINER FEE * * * | 0.00010000 | 0.04 |
| → | | TOTAL: 0.24003068 BTC \| 102.51 USD | |

**NOTE**

BTC-USD RATE AT THE TIME OF TRANSACTION.
OMNI LAYER BALANCES ARE NOT INCLUDED IN THIS REPORT.

**DISCLAIMER**

THIS RECEIPT WAS GENERATED AUTOMATICALLY ON 2023-02-28 09:39 (UTC) AND IS BASED ON PUBLIC DATA FROM THE BITCOIN BLOCKCHAIN. BLOCKCHAIR MAKES NEITHER WARRANTY THAT THIS RECEIPT IS FREE OF ERRORS, NOR WARRANTY THAT ITS CONTENT IS ACCURATE. BLOCKCHAIR WILL NOT BE RESPONSIBLE OR LIABLE TO YOU FOR ANY LOSS OF ANY KIND, FROM ACTION TAKEN, OR TAKEN IN RELIANCE ON INFORMATION CONTAINED IN THIS RECEIPT. BLOCKCHAIR IS NEITHER A BANK, NOR A PAYMENT PROCESSOR FOR THIS PAYMENT. BLOCKCHAIR DOES NOT PROVIDE SUPPORT IN CASE OF PROBLEMS ASSOCIATED WITH THIS RECEIPT.

Scan to check the validity of the receipt at Blockchair.com

**Figure 10: Transaction Receipt Related to Transaction ID 230425980**

# Appendix C: Model Results

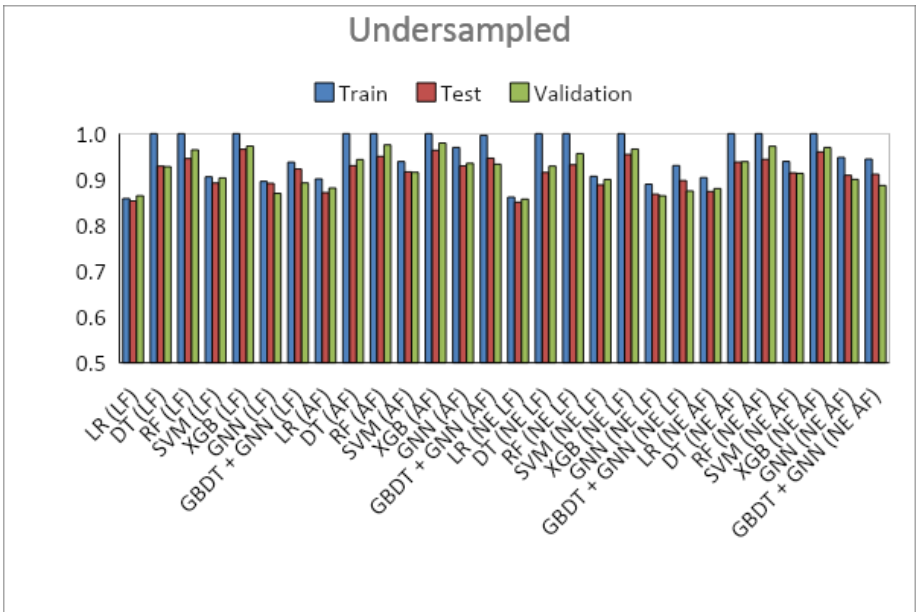


**Figure 14: Train, Test and Validation for the Undersampled Dataset using Default Settings:** Most models slightly overfit, as the training F1 scores were higher than the test and validation F1 scores

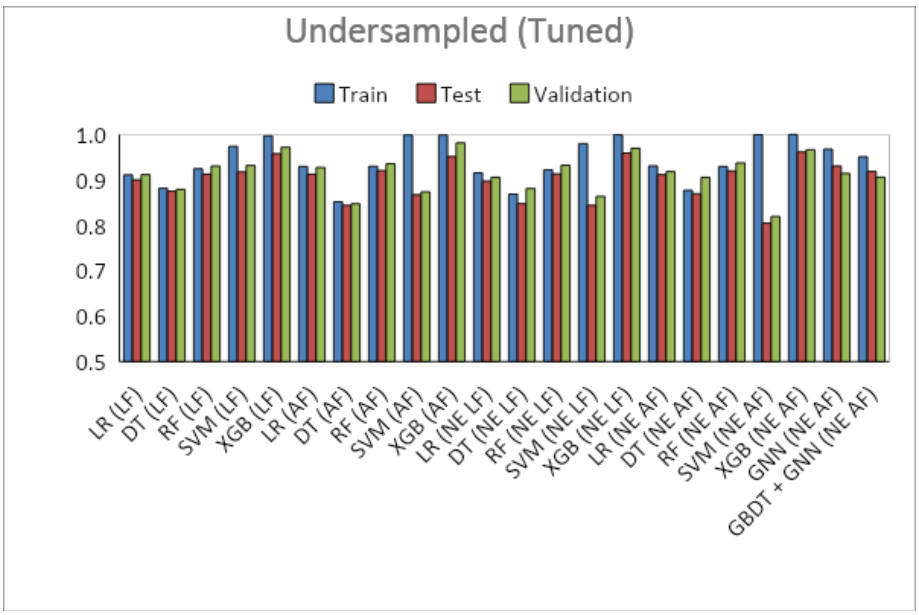


**Figure 15: Train, Test and Validation for the Undersampled Dataset using Tuned Hyperparameters:** Tuning the hyperparameters does not significantly affect the overfitting and test and validation of the F1 scores

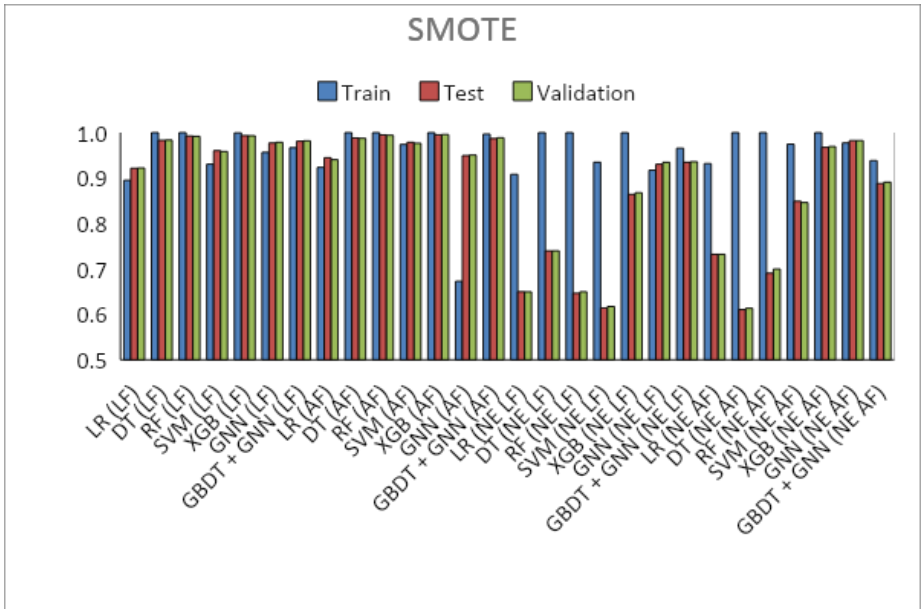


**Figure 16: Train, Test and Validation for the SMOTE Dataset using Default Settings:** The default settings of all models using the local features and all features do not overfit. Introducing node embeddings leads to significant overfitting across all models except for the XGBoost model and the Graph Neural Network model with All Features, which perform well

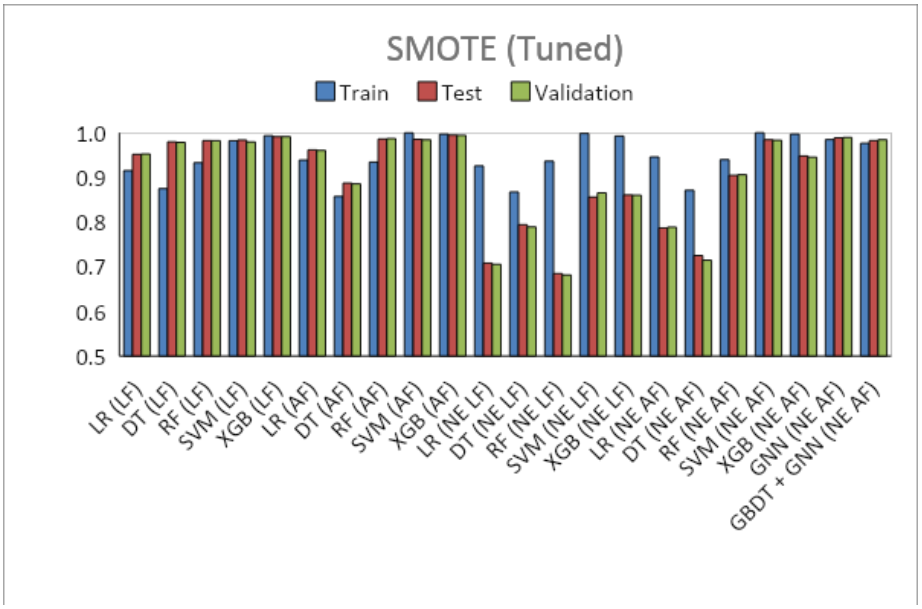


**Figure 17: Train, Test and Validation for the Undersampled Dataset using Tuned Hyperparameters:** The performance of some models improves due to hyperparameter tuning, such as random forest, SVM, XGB, GNN, and GBDT+GNN on the Node-Embedded All Features dataset. Model overfitting is reduced on the named models

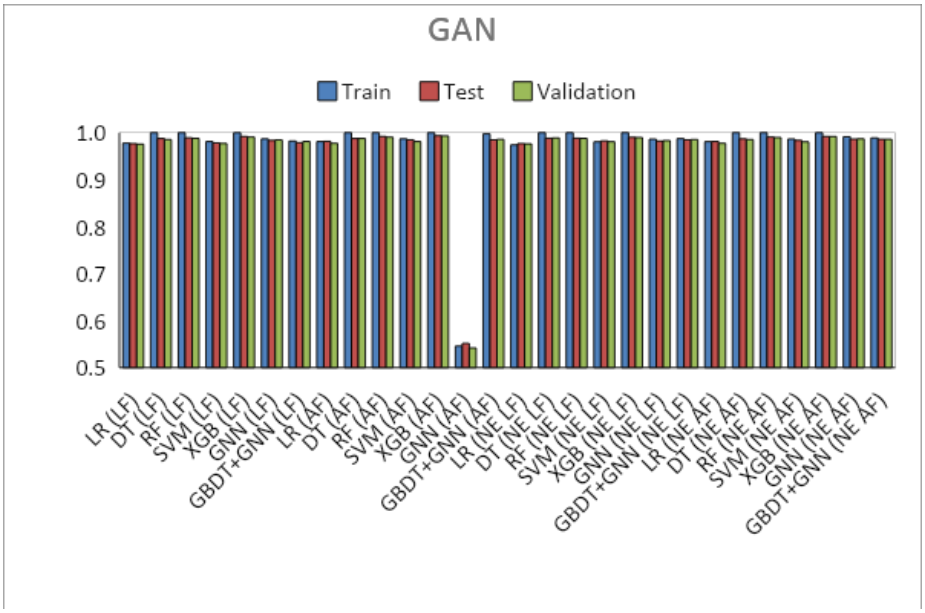


**Figure 18: Train, Test and Validation for the GAN Dataset using Default Settings:** The models do not overfit. However, the GNN model with all features shows signs of underfitting

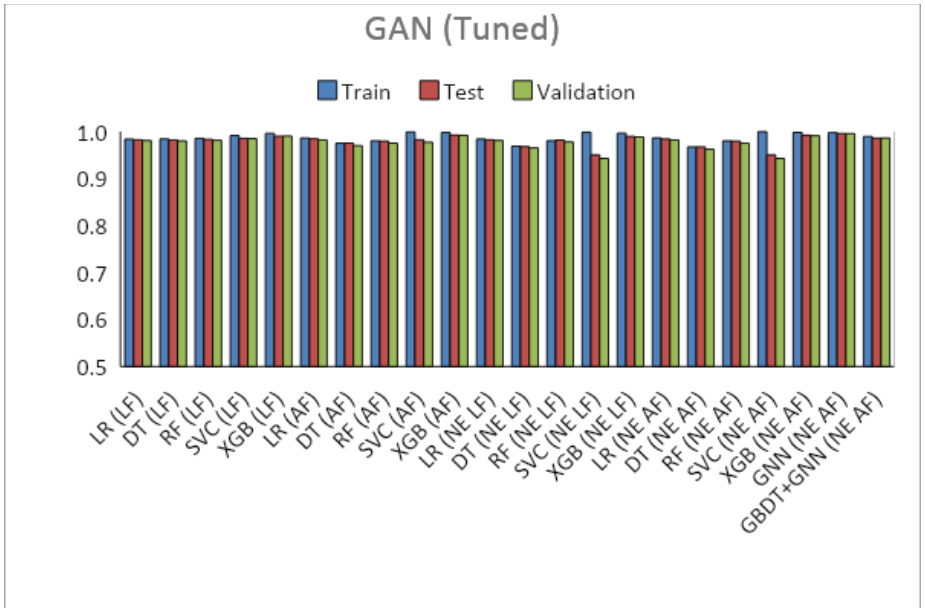


**Figure 19: Train, Test and Validation for the GAN Dataset using Tuned Hyperparameters:** The underfitting of the GNN All Features model is resolved, resulting in the tuned GAN dataset showing the strongest and most generalizable results across all sampling methods and with default and tuned hyperparameters

| Model (Undersampling) | Accuracy | F1 | Precision | Recall | ROC AUC | F1 MicroAVG |
|---|---|---|---|---|---|---|
| Logistic Regression (LF) | 0.870 - 0.909 | 0.864 - 0.912 | 0.928 - 0.906 | 0.809 - 0.918 | 0.872 - 0.909 | 0.870 - 0.909 |
| Logistic Regression (AF) | 0.883 - 0.926 | 0.881 - 0.927 | 0.920 - 0.926 | 0.846 - 0.928 | 0.884 - 0.925 | 0.883 - 0.926 |
| Logistic Regression (NE LF) | 0.862 - 0.903 | 0.857 - 0.906 | 0.914 - 0.905 | 0.807 - 0.907 | 0.864 - 0.903 | 0.862 - 0.903 |
| Logistic Regression (NE AF) | 0.882 - 0.917 | 0.880 - 0.918 | 0.920 - 0.921 | 0.844 - 0.915 | 0.883 - 0.917 | 0.882 - 0.917 |
| Decision Tree (LF) | 0.927 - 0.860 | 0.928 - 0.879 | 0.934 - 0.788 | 0.922 - 0.993 | 0.927 - 0.857 | 0.927 - 0.860 |
| Decision Tree (AF) | 0.942 - 0.854 | 0.943 - 0.847 | 0.947 - 0.915 | 0.939 - 0.790 | 0.942 - 0.856 | 0.942 - 0.854 |
| Decision Tree (NE LF) | 0.928 - 0.882 | 0.929 - 0.881 | 0.934 - 0.912 | 0.924 - 0.852 | 0.928 - 0.883 | 0.928 - 0.882 |
| Decision Tree (NE AF) | 0.938 - 0.904 | 0.939 - 0.906 | 0.945 - 0.916 | 0.933 - 0.896 | 0.938 - 0.905 | 0.938 - 0.904 |
| Random Forest (LF) | 0.963 - 0.926 | 0.965 - 0.931 | 0.953 - 0.891 | 0.976 - 0.974 | 0.963 - 0.924 | 0.963 - 0.926 |
| Random Forest (AF) | 0.974 - 0.931 | 0.975 - 0.936 | 0.960 - 0.898 | 0.991 - 0.976 | 0.974 - 0.930 | 0.974 - 0.931 |
| Random Forest (NE LF) | 0.954 - 0.928 | 0.956 - 0.932 | 0.938 - 0.896 | 0.976 - 0.972 | 0.954 - 0.927 | 0.954 - 0.928 |
| Random Forest (NE AF) | 0.971 - 0.933 | 0.972 - 0.938 | 0.952 - 0.902 | 0.993 - 0.976 | 0.971 - 0.932 | 0.971 - 0.933 |
| Support Vector (LF) | 0.898 - 0.927 | 0.903 - 0.932 | 0.876 - 0.890 | 0.933 - 0.978 | 0.897 - 0.925 | 0.898 - 0.927 |
| Support Vector (AF) | 0.913 - 0.852 | 0.916 - 0.874 | 0.914 - 0.777 | 0.918 - 0.998 | 0.913 - 0.849 | 0.913 - 0.852 |
| Support Vector (NE LF) | 0.894 - 0.842 | 0.900 - 0.964 | 0.875 - 0.775 | 0.926 - 0.976 | 0.894 - 0.839 | 0.894 - 0.842 |
| Support Vector (NE AF) | 0.911 - 0.774 | 0.913 - 0.820 | 0.913 - 0.694 | 0.913 - 1.000 | 0.911 - 0.769 | 0.911 - 0.774 |
| XGBoost (LF) | 0.972 - 0.971 | 0.973 - 0.972 | 0.978 - 0.968 | 0.967 - 0.976 | 0.972 - 0.971 | 0.972 - 0.971 |
| XGBoost (AF) | 0.979 - 0.981 | 0.979 - 0.982 | 0.976 - 0.983 | 0.983 - 0.980 | 0.979 - 0.981 | 0.979 - 0.981 |
| XGBoost (NE LF) | 0.966 - 0.969 | 0.966 - 0.970 | 0.967 - 0.966 | 0.965 - 0.974 | 0.966 - 0.969 | 0.966 - 0.969 |
| XGBoost (NE AF) | 0.969 - 0.966 | 0.970 - 0.967 | 0.966 - 0.961 | 0.974 - 0.972 | 0.969 - 0.965 | 0.969 - 0.966 |
| GNN (LF) | 0.878 | 0.870 | 0.929 | 0.817 | 0.878 | 0.878 |
| GNN (AF) | 0.936 | 0.935 | 0.937 | 0.933 | 0.936 | 0.936 |
| GNN (NE LF) | 0.874 | 0.864 | 0.938 | 0.802 | 0.874 | 0.874 |
| GNN (NE AF) | 0.906 - 0.917 | 0.900 - 0.914 | 0.953 - 0.939 | 0.853 - 0.891 | 0.905 - 0.917 | 0.906 - 0.917 |
| GNN + GBDT (LF) | 0.894 | 0.892 | 0.908 | 0.878 | 0.894 | 0.894 |
| GBDT + GNN (AF) | 0.934 | 0.933 | 0.947 | 0.920 | 0.934 | 0.934 |
| GNN + GBDT (NE LF) | 0.883 | 0.875 | 0.941 | 0.817 | 0.883 | 0.883 |
| GNN + GBDT (NE AF) | 0.893 - 0.910 | 0.887 - 0.906 | 0.942 - 0.947 | 0.837 - 0.869 | 0.893 - 0.910 | 0.893 - 0.910 |

**Table 3: Model Evaluation Metrics for Undersampling. For Completeness the results before tuning are included. The XGBoost model outperforms both the basic models and graph-based models in terms of F1 score**

| Model (SMOTE sampling) | Accuracy | F1 | Precision | Recall | ROC AUC | F1 MicroAVG |
|---|---|---|---|---|---|---|
| Logistic Regression (LF) | 0.868 - 0.917 | 0.922 - 0.952 | 0.990 - 0.989 | 0.863 - 0.919 | 0.892 - 0.908 | 0.868 - 0.917 |
| Logistic Regression (AF) | 0.897 - 0.930 | 0.940 - 0.960 | 0.990 - 0.990 | 0.895 - 0.932 | 0.906 - 0.921 | 0.897 - 0.930 |
| Logistic Regression (NE LF) | 0.530 - 0.588 | 0.649 - 0.705 | 0.998 - 0.998 | 0.481 - 0.545 | 0.736 - 0.768 | 0.530 - 0.588 |
| Logistic Regression (NE AF) | 0.617 - 0.683 | 0.732 - 0.788 | 0.998 - 0.998 | 0.578 - 0.652 | 0.783 - 0.819 | 0.617 - 0.683 |
| Decision Tree (LF) | 0.970 - 0.960 | 0.984 - 0.978 | 0.989 - 0.971 | 0.978 - 0.986 | 0.936 - 0.853 | 0.970 - 0.960 |
| Decision Tree (AF) | 0.977 - 0.812 | 0.988 - 0.885 | 0.990 - 0.987 | 0.985 - 0.803 | 0.945 - 0.851 | 0.977 - 0.812 |
| Decision Tree (NE LF) | 0.625 - 0.683 | 0.740 - 0.789 | 0.994 - 0.993 | 0.589 - 0.655 | 0.777 - 0.805 | 0.625 - 0.683 |
| Decision Tree (NE AF) | 0.494 - 0.595 | 0.613 - 0.715 | 0.994 - 0.987 | 0.444 - 0.560 | 0.708 - 0.745 | 0.494 - 0.595 |
| Random Forest (LF) | 0.985 - 0.968 | 0.992 - 0.982 | 0.989 - 0.986 | 0.995 - 0.978 | 0.945 - 0.922 | 0.985 - 0.968 |
| Random Forest (AF) | 0.989 - 0.976 | 0.994 - 0.987 | 0.991 - 0.986 | 0.998 - 0.988 | 0.954 - 0.925 | 0.989 - 0.976 |
| Random Forest (NE LF) | 0.530 - 0.562 | 0.650 - 0.681 | 1.000 - 0.999 | 0.481 - 0.517 | 0.739 - 0.756 | 0.530 - 0.562 |
| Random Forest (NE AF) | 0.582 - 0.844 | 0.700 - 0.906 | 0.999 - 0.995 | 0.538 - 0.831 | 0.767 - 0.897 | 0.582 - 0.844 |
| Support Vector (LF) | 0.927 - 0.963 | 0.958 - 0.979 | 0.988 - 0.985 | 0.930 - 0.973 | 0.911 - 0.918 | 0.927 - 0.963 |
| Support Vector (AF) | 0.958 - 0.971 | 0.977 - 0.984 | 0.990 - 0.970 | 0.964 - 0.999 | 0.933 - 0.852 | 0.958 - 0.971 |
| Support Vector (NE LF) | 0.498 - 0.783 | 0.617 - 0.865 | 0.996 - 0.986 | 0.447 - 0.771 | 0.716 - 0.834 | 0.498 - 0.783 |
| Support Vector (NE AF) | 0.758 - 0.969 | 0.846 - 0.983 | 0.996 - 0.973 | 0.735 - 0.993 | 0.853 - 0.866 | 0.758 - 0.969 |
| XGBoost (LF) | 0.988 - 0.984 | 0.993 - 0.991 | 0.991 - 0.991 | 0.995 - 0.991 | 0.956 - 0.955 | 0.988 - 0.984 |
| XGBoost (AF) | 0.992 - 0.989 | 0.996 - 0.994 | 0.993 - 0.992 | 0.999 - 0.996 | 0.966 - 0.960 | 0.992 - 0.989 |
| XGBoost (NE LF) | 0.788 - 0.778 | 0.868 - 0.860 | 1.000 - 0.999 | 0.767 - 0.755 | 0.882 - 0.875 | 0.788 - 0.778 |
| XGBoost (NE AF) | 0.946 - 0.906 | 0.969 - 0.945 | 0.998 - 0.998 | 0.942 - 0.898 | 0.964 - 0.941 | 0.946 - 0.906 |
| GNN (LF) | 0.962 | 0.979 | 0.986 | 0.971 | 0.920 | 0.962 |
| GNN (AF) | 0.906 | 0.951 | 0.907 | 0.998 | 0.511 | 0.906 |
| GNN (NE LF) | 0.887 | 0.935 | 0.986 | 0.888 | 0.883 | 0.887 |
| GNN (NE AF) | 0.968 - 0.981 | 0.982 - 0.989 | 0.988 - 0.986 | 0.977 - 0.993 | 0.931 - 0.928 | 0.968 - 0.981 |
| GBDT + GNN (LF) | 0.968 | 0.982 | 0.988 | 0.976 | 0.930 | 0.968 |
| GBDT + GNN (AF) | 0.980 | 0.989 | 0.988 | 0.989 | 0.939 | 0.980 |
| GBDT + GNN (NE LF) | 0.890 | 0.936 | 0.992 | 0.886 | 0.908 | 0.890 |
| GBDT + GNN (NE AF) | 0.821 - 0.972 | 0.891 - 0.984 | 0.992 - 0.988 | 0.809 - 0.981 | 0.874 - 0.933 | 0.821 - 0.972 |

**Table 4: Model evaluation metrics for SMOTE. For Completeness the results before tuning are included. The XGBoost model outperforms both the basic models and graph-based models in terms of F1 score**

| Model (GAN sampling) | Accuracy | F1 | Precision | Recall | ROC AUC | F1 MicroAVG |
|---|---|---|---|---|---|---|
| Logistic Regression (LF) | 0.956 - 0.966 | 0.976 - 0.981 | 0.957 - 0.972 | 0.994 - 0.990 | 0.814 - 0.877 | 0.956 - 0.966 |
| Logistic Regression (AF) | 0.959 - 0.968 | 0.977 - 0.982 | 0.970 - 0.974 | 0.984 - 0.991 | 0.868 - 0.885 | 0.959 - 0.968 |
| Logistic Regression (NE LF) | 0.957 - 0.966 | 0.976 - 0.981 | 0.973 - 0.972 | 0.980 - 0.990 | 0.876 - 0.879 | 0.957 - 0.966 |
| Logistic Regression (NE AF) | 0.959 - 0.968 | 0.977 - 0.982 | 0.972 - 0.972 | 0.982 - 0.992 | 0.876 - 0.880 | 0.959 - 0.968 |
| Decision Tree (LF) | 0.974 - 0.964 | 0.985 - 0.980 | 0.984 - 0.962 | 0.987 - 0.999 | 0.927 - 0.837 | 0.974 - 0.964 |
| Decision Tree (AF) | 0.978 - 0.945 | 0.988 - 0.970 | 0.985 - 0.947 | 0.990 - 0.994 | 0.933 - 0.768 | 0.978 - 0.945 |
| Decision Tree (NE LF) | 0.980 - 0.936 | 0.989 - 0.965 | 0.982 - 0.934 | 0.996 - 0.999 | 0.922 - 0.709 | 0.980 - 0.936 |
| Decision Tree (NE AF) | 0.974 - 0.932 | 0.985 - 0.963 | 0.981 - 0.936 | 0.990 - 0.990 | 0.915 - 0.718 | 0.974 - 0.932 |
| Random Forest (LF) | 0.978 - 0.967 | 0.988 - 0.982 | 0.978 - 0.964 | 0.998 - 1.000 | 0.906 - 0.846 | 0.978 - 0.967 |
| Random Forest (AF) | 0.983 - 0.955 | 0.990 - 0.975 | 0.981 - 0.952 | 1.000 - 1.000 | 0.920 - 0.793 | 0.983 - 0.955 |
| Random Forest (NE LF) | 0.978 - 0.960 | 0.988 - 0.978 | 0.977 - 0.957 | 0.999 - 1.000 | 0.901 - 0.814 | 0.978 - 0.960 |
| Random Forest (NE AF) | 0.981 - 0.955 | 0.990 - 0.975 | 0.980 - 0.952 | 1.000 - 1.000 | 0.914 - 0.792 | 0.981 - 0.955 |
| Support Vector (LF) | 0.959 - 0.973 | 0.977 - 0.985 | 0.968 - 0.972 | 0.986 - 0.998 | 0.860 - 0.881 | 0.959 - 0.973 |
| Support Vector (AF) | 0.966 - 0.959 | 0.981 - 0.977 | 0.971 - 0.957 | 0.991 - 0.999 | 0.875 - 0.814 | 0.966 - 0.959 |
| Support Vector (NE LF) | 0.965 - 0.982 | 0.981 - 0.943 | 0.965 - 0.892 | 0.996 - 1.000 | 0.851 - 0.500 | 0.965 - 0.892 |
| Support Vector (NE AF) | 0.965 - 0.892 | 0.980 - 0.943 | 0.966 - 0.892 | 0.996 - 1.000 | 0.852 - 0.500 | 0.965 - 0.892 |
| XGBoost (LF) | 0.983 - 0.983 | 0.991 - 0.990 | 0.983 - 0.982 | 0.998 - 0.999 | 0.928 - 0.926 | 0.983 - 0.983 |
| XGBoost (AF) | 0.988 - 0.986 | 0.993 - 0.992 | 0.987 - 0.986 | 0.999 - 0.999 | 0.946 - 0.939 | 0.988 - 0.986 |
| XGBoost (NE LF) | 0.981 - 0.979 | 0.990 - 0.989 | 0.980 - 0.980 | 1.000 - 0.998 | 0.915 - 0.913 | 0.981 - 0.979 |
| XGBoost (NE AF) | 0.985 - 0.984 | 0.992 - 0.991 | 0.984 - 0.984 | 1.000 - 0.999 | 0.933 - 0.931 | 0.985 - 0.984 |
| GNN (LF) | 0.971 | 0.984 | 0.973 | 0.996 | 0.865 | 0.971 |
| GNN (AF) | 0.426 | 0.542 | 0.979 | 0.375 | 0.648 | 0.426 |
| GNN (NE LF) | 0.969 | 0.983 | 0.970 | 0.997 | 0.850 | 0.969 |
| GNN (NE AF) | 0.976 - 0.976 | 0.987 - 0.987 | 0.978 - 0.978 | 0.996 - 0.996 | 0.889 - 0.888 | 0.976 - 0.976 |
| GBDT + GNN (LF) | 0.965 | 0.981 | 0.965 | 0.998 | 0.824 | 0.965 |
| GBDT + GNN (AF) | 0.974 | 0.986 | 0.979 | 0.992 | 0.895 | 0.974 |
| GBDT + GNN (NE LF) | 0.973 | 0.985 | 0.974 | 0.997 | 0.871 | 0.973 |
| GBDT + GNN (NE AF) | 0.974 - 0.975 | 0.986 - 0.986 | 0.975 - 0.976 | 0.998 - 0.997 | 0.873 - 0.879 | 0.974 - 0.975 |

**Table 5: Model evaluation metrics for the GAN sampling. For Completeness the results before tuning are included. The XGBoost model outperforms both the basic models and graph-based models in terms of F1 score**

# Appendix D: Models to Inspect Potential Concept Drift

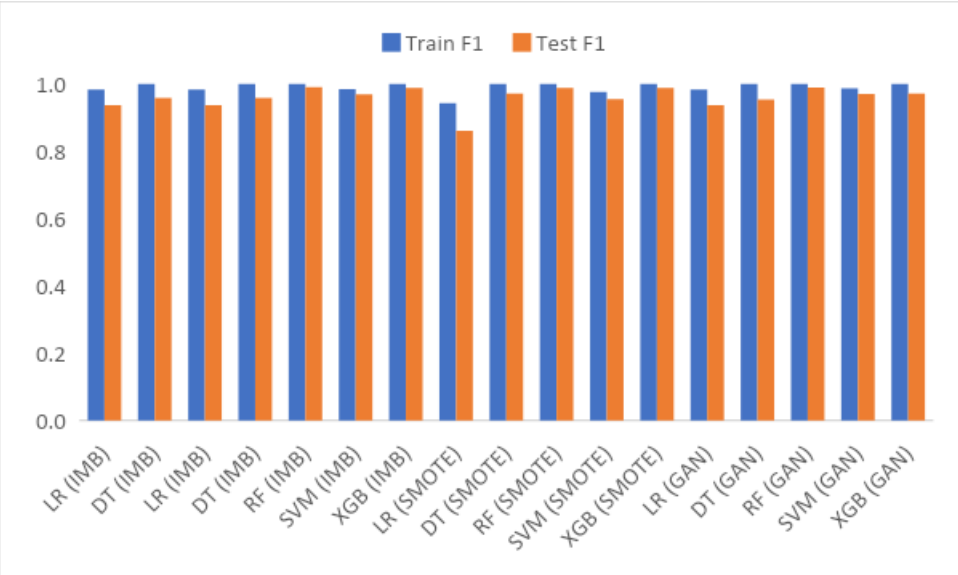


**Figure 22: Offline learning performed on basic models, where the training set includes observations for t < 34 and the test set includes observations for t > 34:** All models show slight overfitting, with most overfitting on the LR model

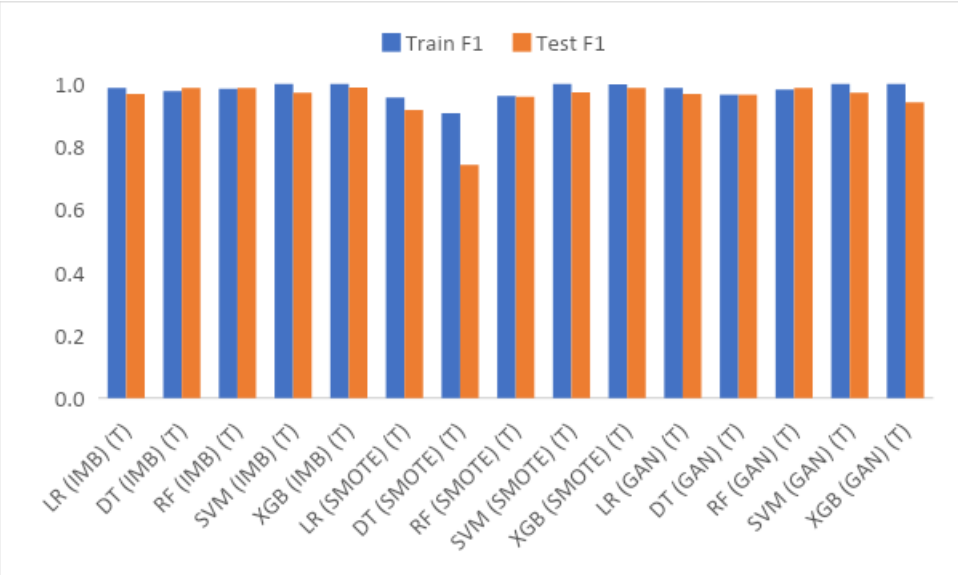


**Figure 23: Offline learning performed on tuned (T) basic models, where the training set includes observations for t < 34 and the test set includes observations for t > 34:** Tuning the models does not significantly improve the slight overfitting compared to the default models