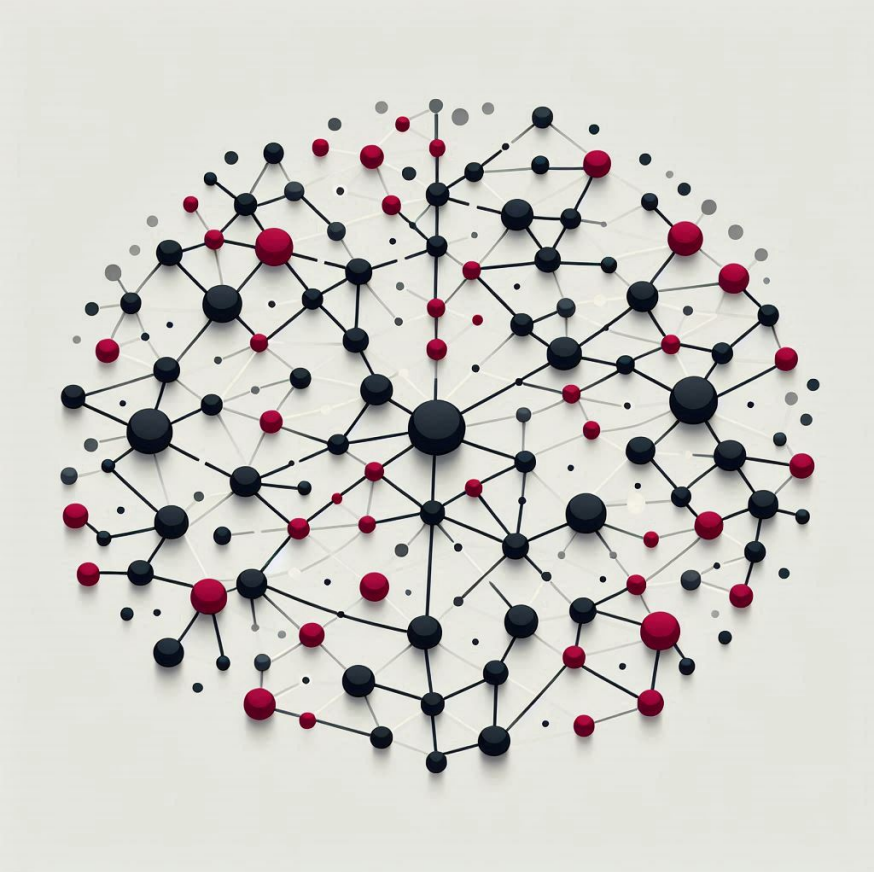

Explainable Anti-Money Laundering

Application of Explainable AI on Graph Neural Networks in Anti-Money Laundering

Master's thesis in Electrical engineering

Tomas Pousette & Agnes Rosendal

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Explainable Anti-Money Laundering

Application of Explainable AI on Graph Neural Networks in Anti-Money Laundering

Tomas Pousette & Agnes Rosendal

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

Explainable Anti-Money Laundering
Application of Explainable AI on Graph Neural Networks in Anti-Money Laundering
Tomas Pousette & Agnes Rosendal

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

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

Cover: **Illustration of a neural network.**

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Explainable Anti-Money Laundering
Application of Explainable AI on Graph Neural Networks in Anti-Money Laundering
Tomas Pousette & Agnes Rosendal
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Financial institutions face significant challenges in detecting suspicious money laundering activities due to high false positive rates generated by traditional models, leading to substantial manual intervention and high operational costs. Deep learning models, particularly graph neural networks (GNNs), have demonstrated potential to improve detection accuracy. However, the complexity and opacity of GNNs pose interpretability issues, limiting their applicability in the anti-money laundering (AML) context. This thesis explores the application of explainable artificial intelligence (XAI) techniques to enhance the interpretability of GNNs in AML. Using synthetic transaction data based on IBM's AMLSim project, a GNN model was explained using three XAI methods: LIME, SHAP, and GraphSVX. The evaluation of the XAI techniques focused on five Co12 metrics: coherence, correctness, completeness, compactness, and confidence. When measuring the coherence between the explainers, we found by comparing the largest 5 to 10 feature importances and the corresponding features about 40% to 50% sign agreement in all pairwise comparisons, except between SHAP and GraphSVX. When comparing the rank, however, the agreement was only about 10% to 20%. The results from measuring the correctness of feature importance were inconclusive, necessitating the establishment of a more suitable method for constructing a feature importance ground truth. This gap presents an opportunity for future research since validation of correctness is critical to build trust in the explanations. The evaluation of the correctness of the node importance obtained from GraphSVX, however, indicate that there is a connection between high node importance and the neighbor also being a money launderer, offering valuable insights for investigators. Furthermore, using 31 features, LIME and SHAP showed, on average, high output completeness for the top 15 feature importances and GraphSVX for the top 20 feature and node importances. Lastly, for the confidence measure, the higher R2 value of the GraphSVX surrogate model over LIME suggests that incorporating comprehensive neighborhood information may be crucial.

Keywords: Graph Neural Network, Anti-Money Laundering, Explainable Artificial Intelligence, Explainable Machine Learning, Anomaly Detection.

Acknowledgements

We would like to express our deepest gratitude to Johan Östman and Edvin Callisen for their exceptional engagement in our work. Their helpfulness and unwavering commitment have been invaluable to us. They have always been available to discuss ideas, offering insightful feedback and continuous support throughout our research. Their dedication has truly made a significant difference in the quality and depth of our thesis.

We also extend our sincere thanks to Jolanta Goldsteine and the data scientist team at Swedbank for giving us a warm welcome and providing us with opportunities to understand and receive feedback on our work from various perspectives within the bank. Their input has been very helpful in shaping our research.

Tomas Pousette & Agnes Rosendal, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AML	Anti-Money Laundering
AI	Artificial Intelligence
LIME	Local Interpretable Model-agnostic Explanations
SAR	Suspicious Activity Report
SHAP	SHapley Additivee exPlanations
SMR	Suspicious Matter Report
XAD	Explainable Anomaly Detection
XAI	Explainable Artificial Intelligence
XML	Explainable Machine Learning
GNN	Graph Neural Network
KYC	Know Your Customer
DDC	Customer Due Dilligence

Contents

List of Figures

List of Tables

1
Introduction

Money laundering is "the conversion or transfer of property, knowing that such property is derived from any offense(s), for the purpose of concealing or disguising the illicit origin of the property or of assisting any person who is involved in such offense(s) to evade the legal consequences of his actions" [3]. Typically, it can be broken down into a three step process: placement, layering, and integration [4]. In the placement phase, the money is moved into the financial system. In the layering phase, the funds are moved around to obscure the trail back to their origin, for example by investing in something legitimate or performing a more complex series of transactions between different accounts or through various companies. Finally, in the integration phase, the funds are re-integrated into a legitimate financial system and reunited with the money launderer, appearing as legitimately earned profits.

The United Nations Office on Drugs and Crime (UNODC) estimated that about 2.3% to 5.5% of the global GDP is laundered money, amounting to about 2.1 trillion US dollars in 2009 [5]. Furthermore, according to a report from the Swedish Economic Crime Authority in 2019 [6], about 130 billion Swedish crowns are laundered each year in the Swedish financial system alone. Since money laundering is commonly associated with criminal organizations aiming at obscuring the origin of cash obtained from for example drug trafficking [7], it is a serious issue with a large societal impact worldwide.

To combat the issue of money laundering, banks and other financial institutions (FIs) are obliged to work intensively with anti-money laundering (AML) efforts. Not adhering to these obligations carries the risk of receiving significant fines. A study from 2022 reveals that financial institutions face compliance costs of approximately $274 billion globally [8]. To this end, FIs in the EU are encouraged to take a risk-based approach, meaning that they assess the risk of money laundering and take appropriate action relative to the level of risk [9, 10]. The risk-based approach involves using various measures to assess and manage the risk, such as Know Your Customer (KYC) for gauging the circumstances and expected behavior of the customer, customer due diligence (CDD) for verifying the validity of the information given by the customer, and transaction monitoring for detecting suspicious behavior, such as large transactions or other behavior that deviates from what is considered normal [4].

Transaction monitoring, which is the primary focus of this work, is, in practice, realized using automated systems that flag accounts for suspicious behavior based on transaction data. An account flagged as suspicious is put under further scrutiny by a team of investigators that manually decide whether a suspicious activity report (SAR) should be filed to the Financial Police, for inspection. Most of the systems

in use today, however, are rule-based models, and suffer from high false-positive rates [11]. This means that many accounts flagged for suspicious behavior turn out not to be actual money launderers when scrutinized by the bank's investigators. This is highly inefficient and leads to large operational costs because of the manual intervention required, and the financial police also receives more reports that are false alarms increasing public costs.

Although a certain percentage of false positives is warranted to guarantee risk coverage, there is a desire to lower it without missing any targets. Thus, there is a clear incentive for employing more sophisticated models to more accurately pinpoint the actual money launderers to reduce the number of false positives, and consequently the operational costs of the bank. The use of deep learning models called graph neural networks (GNNs) has shown promising results in this regard [12]. However, the adoption of these in AML is problematic due to their black-box nature. Since they arrive at conclusions and decisions without comprehensively explaining how they were reached, they may lack the transparency for operating in this domain. This aspect is particularly relevant in light of the recently passed EU AI Act, which put different levels of transparency demands on AI systems according to their risk level [13]. Furthermore, investigators need sufficient information about the decision-making process, to effectively conduct their work. The constraint on transparency therefore means that an additional component is required for incorporating GNNs into the AML pipeline, namely, a mechanism to increase the transparency of black-box models.

This mechanism can be found in the field of explainable artificial intelligence (XAI), which aims at developing methods that provide insights into how black-box models have made their decisions. Furthermore, XAI techniques combined with GNNs could potentially provide investigators with more insights into why an account was flagged. As more sophisticated models such as a GNN utilize more information from the data, an explanation could also be more rich than just the coefficients of a statistical model or the feature importances. For example, an explanation that includes information about the graph structure of the data could be used to pinpoint which other accounts that have significantly impacted the model's prediction. Although there is extensive research on the topic of XAI covering a wide variety of methods [14] applied in various problem domains [15], there is little to no research on XAI being applied in AML [11]. Filling this gap is challenging partially because transaction data from FIs is confidential and generally not available, and partially because there currently is no consensus regarding what constitutes a *good* explanation of a black-box model and how to compare different explanations. In this work, we aim to provide a fundamental study on XAI in AML and challenge the idea of only simple models being explainable.

1.1 Research Questions

In this thesis, we focus on applying existing XAI techniques and analyze their usefulness and shortcomings in the context of AML. Our primary perspective is that of the investigators, who may utilize the information from the XAI techniques to determine if a case should be escalated to the financial police. Our research questions

are thus the following:

Research question 1: Do XAI techniques used on GNNs in AML produce reliable and trustworthy explanations?

Research question 2: Can XAI techniques used on GNNs in AML provide insights into what triggered the flagging of an account, relevant to investigators?

1.2 Approach

Obtaining access to real-world data is challenging due to its confidential nature and the reluctance of banks to disclose information that might reveal typical behavioral patterns. Therefore, we instead use a synthetic dataset based on IBM's AMLSim project [16], which has been further developed at AI Sweden, the Swedish national center for applied AI, in close collaboration with two of Sweden's largest banks: Swedbank and Handelsbanken. The dataset simulates a Swish transaction network [17], with licit and illicit transactions between accounts. We represent the transaction data as a directed graph, where all edges represent a single transaction made between two accounts. This data is then preprocessed and represented as an undirected graph. In the undirected graph, there is an edge between two nodes if there is at least one edge between them in the directed graph, and each node is associated with a node feature vector with information aggregated from the previous edges. The GNNs are then trained on this undirected graph to perform node-level classification to detect suspicious behavior. A variety of XAI techniques is then applied as a postprocessing step to understand the output of the GNN, and the output from these techniques is analyzed. The analysis is focused on investigating i) the level of agreement between the different techniques, ii) the correctness of the explanations, iii) the extent to which the explanations are not missing any relevant information, iv) the size of the explanations, and v) how confident the explanations are.

1.3 Expected contribution

This thesis aims at providing insight into the usefulness of some state-of-the-art XAI techniques in AML, regarding the quality of the explanations and to what degree they meet the requirements of the investigators. It also aims to identify the shortcomings of these techniques to guide future development of XAI techniques tailored more specifically to the domain of AML.

Furthermore, using more sophisticated models could, in addition to lowering the operational costs, potentially help identify more cases of money laundering [12] which will have a positive societal impact [18]. Furthermore, as FIs may feel discouraged from moving from their current models, already having sufficient risk coverage to comply with current laws and regulations, this study aims to accelerate the adoption of more sophisticated models.

1.4 Report Outline

Chapter 2 gives an introduction to AML and XAI. Chapter 3 covers the theory of GNNs and the XAI techniques used. Chapter 4 introduces the synthetic dataset and explains how its parameters have been chosen to reflect real-world conditions, covers the implementation and training of the GNNs, and outlines the metrics used for evaluating the XAI techniques. Chapter 5 summarizes the results. In chapter 6, our results are discussed and directions for future research are proposed. Finally, in Chapter 7, we conclude our findings.

2
Background

The purpose of this section is to provide an overview of the key concepts and literature that form the foundation of this research. Specifically, we delve into the machine learning models used in AML and the growing field of XAI, focusing on their application and relevance in AML.

2.1 Review of Machine Learning Models used in AML

In practice transaction networks are vast, consisting of billions of transactions, and constantly evolving in time. Transaction monitoring on these networks in AML can be approached as an account-level classification problem, meaning that the accounts are flagged as suspicious rather than individual transactions. Monitoring is ongoing but not acted on in real-time, as the alerts undergo manual scrutiny by the team of investigators.

The detection of money laundering is a multi-faceted problem as there are many ways in which an account can behave suspiciously. Typically, several models are employed for detecting different signals, for example, specializing in detecting a particular type of alert pattern, which is a type of money laundering transaction pattern as can be seen in Appendix C. Information that is used in this model can come from transaction data or for example KYC data.

A variety of machine learning models have been applied to detect suspicious behavior in transaction data [11]. Popular models include decision trees, random forests, artificial neural networks (ANNs), and support vector machines (SVMs). In these cases, information about the structure of the transaction network is used by the model only in the form of aggregated features, and accounts are otherwise processed individually and independently by the models. The authors find that these methods have undergone extensive testing and comparisons on large imbalanced datasets and that choosing the right one is problem-dependent. Moreover, these models suffer from large false-positive rates.

In the literature, there is some research on the topic of GNNs, autoencoders, and multi-layer perceptrons in AML [11]. For example [19] uses a GCN to detect suspicious behavior on the AMLSim dataset. Although there is extensive research on machine learning for detecting suspicious behavior in AML according to [11], there is almost no evidence of these models being used in practice. Furthermore, they find that the research on deep learning (DL) in AML is very limited. According to [12] GNNs outperform traditional methods both when trained in a federated setting as

well as in silo and are therefore of interest to study further.

2.2 Explainable AI

According to [2], the definition used of what is explainable is the following: "An explanation is a presentation of (aspects of) the reasoning, functioning and/or behaviour of a machine learning model in human-understandable terms". Here, functioning refers to the internal workings and internal data structure of the machine learning model, while behavior corresponds to how the model globally operates without specifically analyzing the internal workings, leading to a simplified interpretation. As we will rely on [2] for evaluating XAI techniques, the same definition of explainability is used. Although some practitioners distinguish between something being interpretable and explainable [20, 21], many still use the two terms interchangeably [2], which is also what we do in the remainder of this report.

While explainable AI (XAI) has grown as a research topic in recent years, it was first introduced in the literature in the late 70s, where a system's result was explained using applied rules illustrated by a decision tree, which is explainable by design [22]. By studying a decision tree from the top and going down, humans can interpret the system's logic in making decisions.

Today, the concept of XAI has widened, especially with the increasing performance observed in deep learning methods. In the Defense Advanced Research Projects Agency's (DARPA) Explainable Artificial Intelligence Program, they present how the explainability of a machine learning model can be seen as inverse to its prediction accuracy [1].

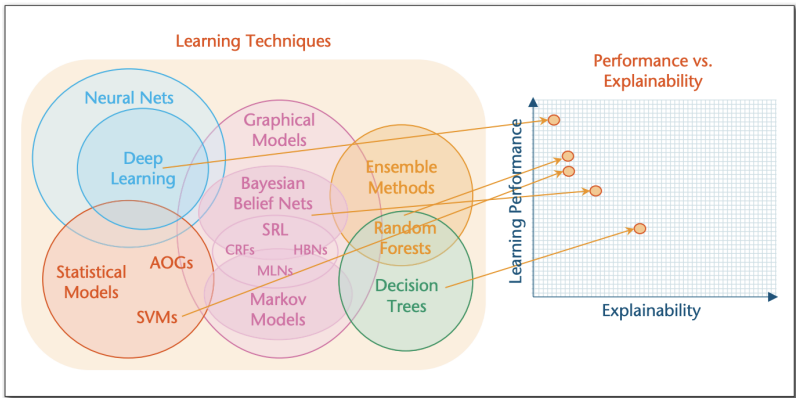

Figure 2.1: Learning Performance Versus Explainability Trade-Off for Several Categories of Learning Techniques from [1].

That is, it is claimed that a deep learning model, such as a GNN, exhibits a higher performance but lower explainability. At the same time, a simple decision tree is highly explainable but sacrifices the prediction accuracy. For this reason, recent research has been devoted to developing XAI techniques for deep learning models, to enable the usage of more sophisticated models in situations where deep learning models are superior to inherently explainable models [22].

2.2.1 Pre- In- and Post-model Techniques

XAI is commonly divided into three categories, pre-model techniques, in-model techniques, and post-model techniques. These differ at which stage in the application of AI they are exerted.

The first category, also called ante-hoc methods, primarily involves feature selection or feature representation methods, focusing on reducing the feature space and thus the complexity of a given model[15]. Reducing the set of features allows for a less complex and more easily read model, which makes the model more explainable.

The second category, in-model techniques, refers to using inherently interpretable models. Other terms for such models are white-box models or transparent models, and the three terms are used interchangeably in the remainder of this report. In [15], a model is defined as transparent if at least one of the following three properties is true, *simulatability*, *decomposability*, and *algorithmic transparency*. Simulatability means that the model can be simulated by a human and that it is possible to reason about its entire decision-making. Decomposability means that the model can be broken into multiple parts, where each part is easy to explain. Finally, algorithmic transparency means that a human can understand the process by which the model generates output from a given input. Examples of such models are decision trees, regression models, and generalized additive models (GAMs).

Finally, the last category corresponds to post-model techniques, also known as post-hoc methods, and is the primary focus of this report. Post-model techniques are methods applied on top of a machine learning model that attempt to explain the decisions made by the system. A common post-model technique is the use of so-called surrogate methods, which adopt another model as an approximation to the opaque model, to explain the prediction models' decision-making. In this report, surrogate methods have been explored to unpack the black-box model studied.

While pre-model techniques can reduce the complexity of a model, they can not on their own explain a black box model. They are more useful for instances where, for example, the large number of features is what hinders the interpretability of the model. Similarly, in-model techniques do not provide explanations for black-box models but are the actual usage of inherently interpretable models. Post-model techniques, however, are specifically designed to produce explanations of complex and opaque models and are therefore most suited to our work.

2.2.2 Risks when using Explainable AI

The application of XAI in AML systems brings about significant benefits but also presents notable challenges. The European Union's 2023 Tech Dispatch on XAI emphasizes how the opacity of AI systems hides possible defects, such as the existence of bias, inaccuracies, or hallucinations [23]. Further, the opaqueness hinders the ability to understand the models logic when used in decision support. The work on XAI attempts to provide a solution to this issue, by having the system provide clear and understandable explanations for their actions and decisions. For deep learning models, it means utilizing post-hoc techniques to uncover the black box model. However, as discussed in [23], XAI does not come without risks. More specifically, the risks considered most relevant to AML are the risk of *misinterpretation* and

over-reliance.

The risk of misinterpretation appears from the potential oversimplification of a system's functioning, done to make the explanation more accessible to the target group, or by explanations that are too complex or technical to the audience [23]. In such scenarios, there's a risk of misinterpretation by individuals. The risk can be reduced by carefully considering the targeted audience and adjusting the level of detail provided in the explanation. In the context of AML, the audience to consider are the investigators at the bank, and the data engineers and data scientists working with developing the system. It is mentioned in [23] that the explanation process can be aided by visual tools such as graphical representation but should be weighted thoughtfully to avoid oversimplification. Using XAI without risking misinterpretation means finding a delicate balance between providing something understandable yet faithful to the actual complexity of the model to explain. It is therefore essential to validate and test XAI methods to ensure the explanations reflect the true behavior of the AI system.

Another risk with using XAI is over-reliance on AI systems by deployers [23]. In [24], it is highlighted that to explain algorithmic decision-making, we need to understand the AI algorithm and its influence on human decisions. The likelihood of automation bias, i.e., that humans blindly accept AI recommendations regardless of their correctness, may increase by the use of XAI [23]. Therefore, it is pointed out that XAI should include mechanisms allowing human intervention to challenge the explanation. It is, therefore, crucial to promote human involvement in AI systems deployed in areas with significant consequences. Additionally, communicating the limitations of an AI system and its explanation is vital to ensure the technology is utilized to make responsible and socially accepted decisions.

2.2.3 Explainable AI and regulatory compliance

On March 13th, 2024, the European Union passed the world's first comprehensive regulation on AI, the so-called AI Act. The AI Act is a legal framework addressing the risks of AI, categorizing AI systems into four levels of risk categories; minimal risk, limited risk, high risk, and unacceptable risk [25]. To enable accountability for decisions made by companies and public authorities and allow citizens to understand the systems' design and use, the AI Act involves transparency obligations[13].

Transparency is listed as a general principle applicable to all AI systems [13]. It means that "AI systems shall be developed and used in a way that allows appropriate traceability and explainability while making humans aware that they communicate or interact with an AI system, as well as duly informing users of the capabilities and limitations of that AI system and affected persons about their rights". In this provision, transparency is related to the technical infrastructure of AI systems. Particularly, the functioning needs to be transparent so that users can understand how decisions are made and the logic behind them. This means that it should explain how an AI system arrived at its decisions, information on the data used to train the system, and the accuracy of the system.

While the above referred to all AI systems, special requirements are applied for high-risk systems: "high-risk AI systems shall be designed and developed in such

a way to ensure that their operation is sufficiently transparent to enable providers and users to reasonably understand the system's functioning." [**?**]. High-risk systems are the systems where AI technology is used in areas including, but not limited to, critical infrastructures, essential public and private services, and law enforcement that may interfere with people's fundamental rights. AI systems used in AML may thus fall under the category of essential private service, and therefore be categorized as high-risk. However, it should be noted that each AI system deployed may require an individual risk assessment, so the exact risk level for such a system remains undetermined.

Lastly, the details of how transparency obligations should be fulfilled for deep learning models are yet to be agreed on. However, XAI, serving as a tool to decrease the opaqueness of an AI system, will likely play a part in achieving these obligations.

2.2.4 XAI techniques used in general

As the interest in XAI has increased in recent years, so has the number of available XAI methods. We can therefore not expect to list all of them here, or even cover all the different types of methods available. Instead, we mention some methods that were considered at the beginning of this thesis. This includes methods applicable to a range of different types of models, as well as models that are specifically tailored for GNNs. While some methods have been more studied than others, the application of XAI methods in the domain of AML has not been thoroughly studied [11].

Two methods that have gained significant attention, in general, are LIME and SHAP [26, 27]. These methods return feature importance, are model agnostic, work on any type of data, and there are specific implementations optimized for different models. There is also an implementation of LIME called C-LIME where the feature importance is an approximation of the gradient at the point that is being explained. Furthermore, the authors of LIME propose a method called Anchors.

Gradient-based attribution methods are a type of explanation methods that aim at using gradient information to produce an explanation. Popular methods include SmoothGrad, Gradient*Input, and DeepLIFT. It can be shown that the output for C-LIME and SmoothGrad converge to the same values in expectation for a large number of perturbed samples [28].

A limitation of the mentioned XAI techniques is that they do not incorporate any information about the graph structure into their explanation. Since GNNs utilize the graph structure of the data it seems reasonable that the explanation should consider this too. As transaction data is naturally represented as a graph, such methods are interesting in the context of AML.

Some methods are more tailored for a specific problem, described in [15]. To include an XAI technique tailored specifically to GNNs, we considered several candidates including GNNExplainer [29], CF-GNNExplainer [30], GraphLIME [31], and GraphSVX [32]. GNNExplainer provides edge importance and a binary node mask, which seemed not to be suitable in this regard. Furthermore, CF-GNNExplainer provides counterfactual explanations that did not seem appropriate: often there is more than one counterfactual explanation, and it is not obvious in which way this information should be used by an investigator.

	Co-12 Property	Description
Content	**Correctness**	Describes how faithful the explanation is w.r.t. the black box.
		Key idea: Nothing but the truth
	Completeness	Describes how much of the black box behavior is described in the explanation.
		Key idea: The whole truth
	Consistency	Describes how deterministic and implementation-invariant the explanation method is.
		Key idea: Identical inputs should have identical explanations
	Continuity	Describes how continuous and generalizable the explanation function is.
		Key idea: Similar inputs should have similar explanations
	Contrastivity	Describes how discriminative the explanation is w.r.t. other events or targets.
		Key idea: Answers "why not?" or "what if?" questions
	Covariate complexity	Describes how complex the (interactions of) features in the explanation are.
		Key idea: Human-understandable concepts in the explanation
Presentation	**Compactness**	Describes the size of the explanation.
		Key idea: Less is more
	Composition	Describes the presentation format and organization of the explanation.
		Key idea: *How* something is explained
	Confidence	Describes the presence and accuracy of probability information in the explanation.
		Key idea: Confidence measure of the explanation or model output
User	**Context**	Describes how relevant the explanation is to the user and their needs.
		Key idea: How much does the explanation matter in practice?
	Coherence	Describes how accordant the explanation is with prior knowledge and beliefs.
		Key idea: Plausibility or reasonableness to users
	Controllability	Describes how interactive or controllable an explanation is for a user.
		Key idea: Can the user influence the explanation?

Figure 2.2: The Co-12 explanation quality properties from [2], grouped by their most prominent dimension: Content, Presentation or User.

2.2.5 Evaluation of Explainability

Although the concept of XAI was coined decades ago, the number of XAI techniques has grown rapidly in recent years. As highlighted in Section 2.2.2, there is a need to validate and test XAI methods properly. However, there is no clear consensus on what a good explanation is, and thus it is unclear how to compare explainers to each other. This is partly because different explainers rely on separate theoretical justifications, and hence have a slightly different perspective on what an explanation is, making the evaluation process complex.

A survey from 2023 summarizes how XAI has been assessed in more than 300 papers published over the previous 7 years [2]. The authors of [2] suggest that explainability shouldn't be seen as a binary characteristic, but instead as a multifaceted property. To this end, the authors propose the co-12 framework, which consists of 12 desirable properties an XAI technique should have, grouped by content, presentation, and user. The twelve properties are *correctness, completeness, consistency, continuity, contrastivity, covariate complexity, compactness, composition, confidence, context, coherence,* and, *controllability.*

The risk of over-reliance and misinterpretation, highlighted in Section 2.2.2, can be mitigated by applying proper evaluation methods. The risk of misinterpretation, which means oversimplifying a system's functioning and leading to potential misinterpretations, is mitigated by evaluating the compactness and output-completeness of an explanation. Ensuring the explanation is concise while at the same time being accurate, would decrease the risk of misinterpretation. Additionally, the risk of over-

reliance decreases through a comprehensive evaluation, as it sheds light on potential limitations an XAI technique has, encouraging human integration and intervention when necessary.

The survey showed how many Co-12 properties were evaluated in the 300 papers studied [2], and revealed that only 58% of the research papers performed a quantitative evaluation, meaning the usage of some type of evaluation metric. In the remaining papers, the evaluation is solely done on "the researcher's intuition of what constitutes a good explanation". Out of the papers including a quantitative evaluation, a clear majority evaluated only one or two of the twelve properties. Notably, only in three papers, five or more properties were evaluated.

2.3 Conclusion

Since transaction data is naturally represented as a graph it seems natural that a model leveraging this structure in addition to aggregated information on each account should perform better than a model using only the aggregated information. Since there is research showing that GNNs trained both in a federated setting as well as in silo perform better, opting to explore XAI on GNNs, therefore, seemed like a reasonable research direction for this thesis.

Given that LIME and SHAP are widely used in other areas, we think these are explainers that are interesting to include in our study. Since these are not specifically designed to explain GNNs, it is interesting to understand the extent to which they are suitable for GNNs, particularly in this use case.

For GraphLIME, the official implementation from the paper, or an implementation from a well-respected library could not be found publicly. The only implementation found seemed to output very sparse explanations with feature importance that had almost no coherence when compared to GraphSVX. For GraphSVX we found the official implementation from the original paper, and we thus decided to use this explainer. GraphSVX also seemed interesting since it is based on Shapley values, and because it attributes importance to the presence of different neighbors around the account that is being explained.

To evaluate the explainers, we used five of the Co-12 properties: coherence, correctness, completeness, compactness, and confidence. These properties were chosen because they are suitable for explainers that output feature importance and are interesting for investigating misinterpretation and over-reliance.

3
Theory

This chapter serves as a cornerstone for understanding the subsequent analyses and applications within the domain of GNNs and XAI techniques. It is divided into three principal sections:

In Section 3.1, we provide a comprehensive examination of graphs, unraveling their fundamental constituents and characteristics. This groundwork is essential for grasping the structural framework underlying graph-based models.

Section 3.2 delves into the basis of GNNs, clarifying their architectural principles, and operational mechanisms. This exploration sheds light on how GNNs leverage the inherent structural complexities of graphs to accomplish learning and inference tasks across various domains.

Lastly, Section 3.3 explores the domain of XAI techniques, focusing on their application within graph-based models. Here, we examine the pivotal role of XAI methodologies in enhancing the interpretability and transparency of black box models, facilitating a deeper understanding and trust in model decisions.

3.1 Definition of a graph

As was mentioned in Section 1, the transaction data can be seen as both a directed graph and an undirected graph. Here we only define an undirected graph, as it is this type of graph the GNNs will be operating on.

An *undirected graph* is defined as a set of *nodes* $\mathcal{V} = \{v_1, \ldots, v_N\}$ and a set of *edges* $\mathcal{E} = \{e_1, \ldots, e_M\}$, where $e_i = \{v, u\}$ for $v, u \in \mathcal{V}$. A visual representation of a graph is shown in Figure 3.1. In this example we have

$$\mathcal{V} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}, \tag{3.1}$$

$$\mathcal{E} = \{\{\mathrm{A}, \mathrm{B}\}, \{\mathrm{A}, \mathrm{C}\}, \{\mathrm{A}, \mathrm{E}\}, \{\mathrm{B}, \mathrm{E}\}, \{\mathrm{C}, \mathrm{D}\}, \{\mathrm{C}, \mathrm{F}\}, \{\mathrm{D}, \mathrm{F}\}\}. \tag{3.2}$$

In the context of this work, nodes represent bank accounts, and an edge between two nodes represents that one or more transactions have been made between the two accounts. Furthermore, we associate each node with a *node feature vector* and a *label*. The node feature vector, $\mathbf{x}_i \in \mathbb{R}^n$, contains *features* such as the amount spent, or the number of transactions made over a given period of time. The label $y_i \in \{0, 1\}$ is 1 if the account is a money launderer and 0 otherwise. More details about exactly what features we used are given in Chapter 4. When referring to *graph structure* we mean the sets $\mathcal{V}$ and $\mathcal{E}$. And when referring to a *graph* we will mean both graph structure and the node feature vectors X. Finally, the neighbors of a node v is the set $\mathcal{N}(v) = \{u \in V : \{u, v\} \in \mathcal{E}\}$, the *k-hop neighborhood* of v

is the set $\mathcal{N}_k(v) = \{u \in V : \text{shortest path between } v \text{ and } u \leq k\}$, and the *k-hop subgraph* around v is the graph $G_k(v)$ defined by the nodes $\mathcal{N}_k(v)$, the edges $\mathcal{E}_k(v) = \{\{u, v\} \in \mathcal{E} : u \in \mathcal{N}_k(v)\}$, and the node feature vectors $\mathcal{X}_k(v) = \{\mathbf{x}_u : u \in \mathcal{N}_k(u)\}$.

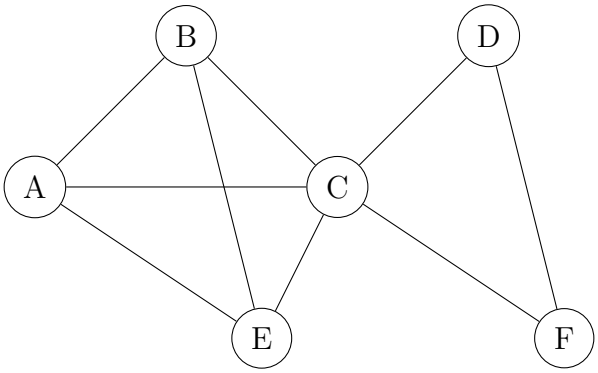

Figure 3.1: An example of a graph. The nodes, denoted by $A, \ldots, F$ represent bank accounts, and the edges between them signify that at least one transaction has occurred between the accounts.

3.2 Graph Neural Networks

A GNN is a type of deep learning model that uses the graph structure of data to translate node feature vectors into embeddings amenable to classification. The fundamental idea of a GNN is to successively compute new node embeddings of each node by combining the information of itself and its neighbors [33]. A GNN consists of layers, where each layer computes a new node embedding for each node. We denote the embedding of node v in the k:th layer by $\mathbf{h}_v^{(k)}$. In the 0:th layer the node embedding is set to $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. Then, the procedure for computing new embeddings is generally broken down into a two-step process: aggregation and updating, as is illustrated for node C from Figure 3.1 in Figure 3.2. The node embeddings of the neighbors of C are aggregated through the function AGG producing a message $\mathbf{m}_{\mathcal{N}(v)}^{(k)}$, and then the new embedding of C is calculated through the update function UPD. Both AGG and UPD are learnable differentiable functions that are different from layer to layer but shared among all nodes in any particular layer.

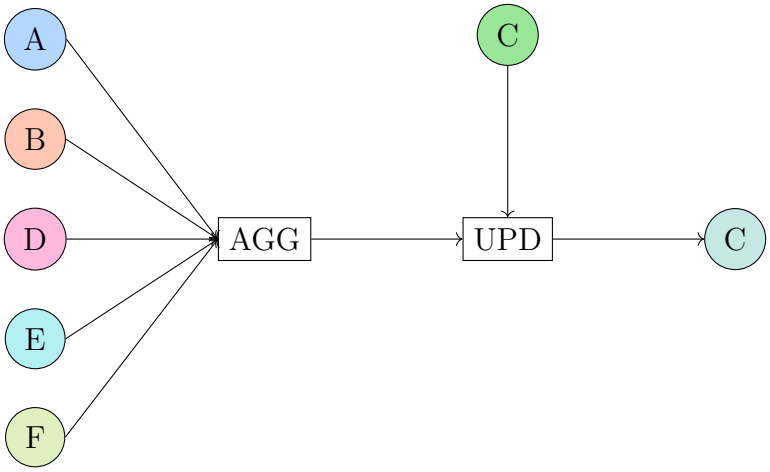

Figure 3.2: A visualization of how the neighborhood information is aggregated and used to update the embedding of the node C.

Mathematically we can express the node embedding $\mathbf{h}_v^{(k)}$ of node v in the k:th layer of the GNN as

$$\mathbf{h}_v^k = \text{UPD}^{(k-1)}(\mathbf{h}_v^{(k-1)}, \mathbf{m}_{\mathcal{N}(v)}^{(k-1)}), \tag{3.3}$$

where

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \text{AGG}^{(k-1)}(\{\mathbf{h}_u^{(k-1)} \ : \ \forall u \in \mathcal{N}(v)\}). \tag{3.4}$$

The function AGG must be permutation invariant so that the ordering of the nodes does not matter. After applying k layers, we obtain a graph with embedded node feature vectors, now containing richer information from their respective neighborhoods, as is shown in figure 3.3. These final node embeddings can then be used for various tasks, such as node level classification.

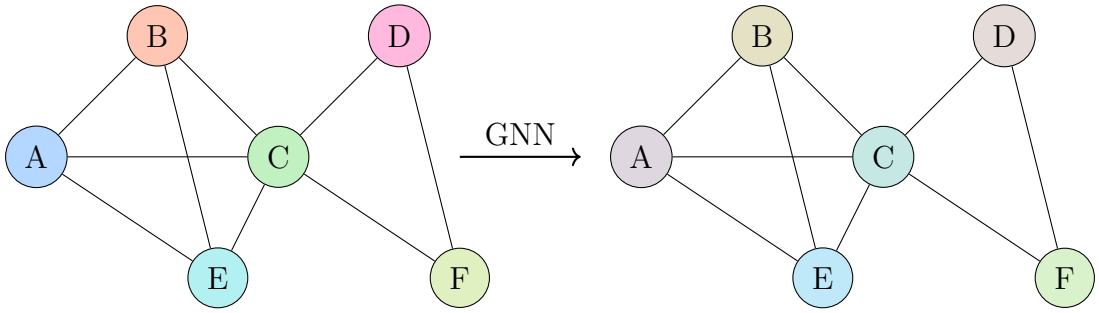

Figure 3.3: Before (left) and after (right) applying a GNN to a graph, where the colors represent the values of the node feature vectors. On the right, the colors are a combination of itself and the neighboring nodes prior to applying the GNN.

Depending on how one chooses the functions AGG and UPD the GNN obtains different properties. In this work we use a type of GNN architecture called Graph Attention Network (GAT) [34]. For GAT, the AGG and UPD functions combine into

$$\mathbf{h}_v^{(k)} = \sigma\Big(\frac{1}{K}\sum_{k=1}^{K}\sum_{u\in\mathcal{N}(v)} \alpha_{vu}^{(k)}\mathbf{W}^{(k)}\mathbf{h}_u^{(k-1)}\Big), \tag{3.5}$$

where K is the number of attention heads, and $\alpha_{vu}^{(k)}$ is the normalized attention coefficients, which essentially determines how much the information from node u that v matters. The formula for $\alpha_{vu}^{(k)}$ is given by

$$\alpha_{vu}^{(k)} = \text{softmax}_u(e_{vu}^{(k)}) = \frac{\exp e_{vu}^{(k)}}{\sum_{\tilde{u}\in\mathcal{N}(v)} \exp e_{v\tilde{u}}^{(k)}}, \tag{3.6}$$

where e_{vu} are called the attention coefficients and are given by

$$e_{vu}^{(k)} = a^{(k)}(\mathbf{W}_a^{(k)} h_v^{(k)}, \mathbf{W}_a^{(k)} h_u^{(k)}), \tag{3.7}$$

where c is defined by $h_v^{(k)} \in \mathbb{R}^c$, $a^{(k)} : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ is called the attention mechanism, and $\mathbf{W}_a^{(k)}$ is a weight matrix. In the implementation used, the attention mechanism is parameterized by a vector $\mathbf{a}^{(k)} \in \mathbb{R}^{2c}$ and expressed as:

$$a^{(k)}(\mathbf{W}_a^{(k)} h_v^{(k)}, \mathbf{W}_a^{(k)} h_u^{(k)}) = \text{LeakyReLU}\Big((\mathbf{a}^{(k)})^\top [\mathbf{W}_a^{(k)} h_v^{(k)} || \mathbf{W}_a^{(k)} h_u^{(k)}]\Big), \tag{3.8}$$

where $||$ denotes concatenation.

3.3 XAI Techniques

In the following section, we delve into the theoretical foundations of three distinct XAI techniques examined in this study: LIME, SHAP, and GraphSVX, as well as the concept of Shapley Values, rooted in cooperative game theory. We begin with an overview of LIME, followed by an exploration of the foundational theory of Shapley Values, which serves as the basis for both SHAP and GraphSVX methods. Finally, we delve into the details of SHAP and GraphSVX.

3.3.1 LIME

LIME stands for Local Interpretable Model-agnostic Explanations, and is a framework for producing explanations by approximating the black box model locally around the prediction with an interpretable model [35]. This interpretable model then serves as an explanation. For example, when fitting a linear model, the coefficients of the model give us an importance related to each feature.

More formally, for any model $f : \mathbb{R}^d \to \mathbb{R}$, the goal of LIME is to find an *explanation* g in some class of *potentially interpretable* functions G, meaning that at least some of the functions in G are simple enough to be interpretable, by solving the following optimization problem:

$$g^* = \arg\min_{g\in G} \Big\{ \mathcal{L}(f, g, \pi_{\boldsymbol{\xi}}) + \Omega(g) \Big\}, \tag{3.9}$$

where $\boldsymbol{\xi}$ is the sample to be explained, $\mathcal{L}$ is a loss function, $\pi_{\boldsymbol{\xi}}$ is a weight function that determines which points are considered to be close to $\boldsymbol{\xi}$, Ω is a regularization function whose purpose is to limit the complexity of g. As is done in [35], we choose G to be the set of affine functions, i.e. explanations g of the form

$$g(\mathbf{z}) = \beta_0 + \sum_{j=1}^{d} \beta_j z_j, \tag{3.10}$$

where $\mathbf{z} \in \{0,1\}^d$, and its coordinates z_j are called *interpretable components* which we will describe in more detail later. Furthermore, for $\mathcal{L}$, we use weighted squared loss averaged over N samples generated through a sampling procedure:

$$\mathcal{L}(f, g, \pi) = \frac{1}{n} \sum_{i=1}^{N} \pi_{\boldsymbol{\xi}}(\mathbf{x}_i)\Big(f(\mathbf{x}_i) - g(\mathbf{z}_i)\Big)^2, \tag{3.11}$$

and we take

$$\Omega(g) = 0. \tag{3.12}$$

By writing $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_d)$, and redefining $\mathbf{z}_i = (1, (\mathbf{z}_i)_1, \ldots, (\mathbf{z}_i)_d)$ we can write $g(\mathbf{z}_i) = \boldsymbol{\beta}^\top \mathbf{z}_i$. In summary, we solve the problem

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta} \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{N} \pi_{\boldsymbol{\xi}}(\mathbf{x}_i)\Big(f(\mathbf{x}_i) - \boldsymbol{\beta}^\top \mathbf{z}_i\Big)^2. \tag{3.13}$$

In the default setting of the python lime package, when using `lime_tabular`, the optimization problem is solved using ridge regression[1].

3.3.1.1 Interpretable components

In the LIME framework, interpretable components are a central concept and might be most easily understood using an example where the input is an image. Given image i, the interpretable components can be constructed by dividing the image into different regions, for example, using semantic segmentation. Then, region j is associated with a binary variable $(\mathbf{z}_i)_j$. The idea is then to attribute importance to the presence $(\mathbf{z}_i)_j = 1$ of this particular region of the image. For example, the head of a dog could carry a lot of importance for classifying the image as a dog, whereas the background carries less importance, making it easy to interpret the meaning of the feature importance score. During the sampling in LIME $(\mathbf{z}_i)_j$ will be either 1 or 0 for different samples, and this will enable us to calculate the importance of this interpretable component, which in turn gives us the importance for this region of the image.

In the case of a continuous feature $\mathbf{x} \in \mathbb{R}^d$ we use the quartiles of the feature distribution to define the interpretable components in the following way: Given a number of bins p, for each feature j, we compute quantiles $\hat{q}_{j,0}, ..., \hat{q}_{j,p}$, where $\hat{q}_{j,0} = -\infty$ and $\hat{q}_{j,p} = \infty$. These quantiles can either be computed empirically, by using the training data directly, or theoretically by knowing the underlying distribution of the training data. Along each dimension j there is a mapping, $\hat{\phi}_j : \mathbb{R} \to \{1, ..., p\}$, associating each real value to the index of the quantile box it belongs to. Any point x can be expressed as a binary vector, where 1 corresponds to belonging to the same quantile box as the explanation point x. In other words, $z_j = \mathbb{1}_{\hat{\phi}_j(\boldsymbol{\xi})}(\hat{\phi}_j(x))$ for all $1 < j < d$.

Note that for $\boldsymbol{\xi}$ we have $z_{\boldsymbol{\xi}} = \mathbb{1}_{\hat{\phi}(\boldsymbol{\xi})}(\hat{\phi}(\boldsymbol{\xi})) = \mathbb{1}$ so in Equation (3.10) the coefficients $\beta_1, \ldots, \beta_d$ says how important it was that the feature values of $\boldsymbol{\xi}$ lied in their respective quantiles.

[1] https://github.com/marcotcr/lime

3.3.1.2 Sampling

The sample procedure is quite non-trivial, and here we intend to clarify the procedure by an example. Given an integer p that is equal to the number of quantiles to use, for example $p = 4$ in the case of using quartiles which is the case in this thesis, we sample N points uniformly $\mathbf{u}_i \sim U(\{1, \dots, p\}^d)$, where $i = 1, \dots, N$. For each point $\mathbf{u}_i$ we construct one continous vector $\mathbf{x}_i \in \mathbb{R}^d$ and one binary vector $\mathbf{z}_i \in \{0, 1\}^d$. The continuous vector is constructed by sampling from a truncated normal distribution

$$(\mathbf{x}_i)_j \sim \mathcal{N}_T(\mu_j, \sigma_j, q_l, q_r), \tag{3.14}$$

where $(\mathbf{x}_i)_j$ is the j:th coordinate of $\mathbf{x}_i$, and q_l, q_r are the quantiles such that $l + 1 = r = (\mathbf{u}_i)_j$. The binary vector is constructed by featurewise comparison with the discretization of $\boldsymbol{\xi}$ so that

$$(\mathbf{z}_i)_j = \mathbb{1}_{\hat{\phi}_j(\boldsymbol{\xi})}((\mathbf{u}_i)_j). \tag{3.15}$$

By generating several samples in this way we obtain several pairs of values (x_i, z_i) that we optimize the β-coefficients for in Equation (3.13). For an example of how a sample is generated, see Appendix B.

3.3.1.3 Weight function $\pi_{\boldsymbol{\xi}}$

The weight function $\pi_{\boldsymbol{\xi}}$ can be defined in different ways. In this work, we use a Gaussian kernel

$$\pi_{\boldsymbol{\xi}}(\mathbf{x}_i) := \exp\left(\frac{-||\mathbf{x}_i - \boldsymbol{\xi}||^2}{2\nu^2}\right), \tag{3.16}$$

where $\nu > 0$ is the called the *kernel width*. In the implementation of LIME, the default value is set to $\nu = 0.75\sqrt{d}$, where d is the number of features, but no motivation for this value is given in the article.

The bandwidth size has a clear impact on the result. A too large ν will give nonzero weights to all examples, and thus the effect will be that g gets fitted globally, which is unsuitable when f is too complicated compared to g. For too small ν values, on the contrary, only examples sharing the same bin categories will receive a non-zero weight, and the linear model will be a constant fit that misses the information we aim to capture [36]. Since there is no standard approach to choosing ν we evaluate LIME using its default value.

3.3.1.4 Theoretical implications

While the use of interpretable components make the interpretation of the results easier, it disguises to some degree what is actually being optimized and consequently what the β-coefficients measure. As is done in [37], a different approach is to not use interpretable components, writing (3.10) as a sum over the continuous features x_j, and only sample continuous points around $\boldsymbol{\xi}$, the optimization quite clearly make the β-coefficients equal to the gradient of a plane fitted locally to the black-box.

Using interpretable components, however, the interpretation is not as easy. As is shown in [38]: under the assumption that f is bounded, using

$$\pi_{\boldsymbol{\xi}}(\mathbf{x}_i) := \pi_i = \exp\left(-\frac{||\mathbb{1} - \mathbf{z}_i||^2}{2\nu^2}\right), \tag{3.17}$$

and $\Omega = 0$, the coefficients obtained from (3.13) converge to

$$\beta_j^f := c^{-d}\Big\{\frac{-pc}{pc-1}\mathbb{E}[\pi_i f(\mathbf{x})] + \frac{p^2c^2}{pc-1}\mathbb{E}[\pi_i z_j f(\mathbf{x})]\Big\}, \tag{3.18}$$

and

$$\beta_0^f := c^{-d}\Big\{\Big(1 + \frac{d}{pc-1}\Big)\mathbb{E}[\pi_i f(\mathbf{x})] + \frac{pc}{pc-1}\sum_{j=1}^{d}\mathbb{E}[\pi_i z_j f(\mathbf{x})]\Big\}, \tag{3.19}$$

where c is a constant depending on p and ν. From Equation (3.18) we see that the importance of feature j is captured in a non-trivial way through the second term involving $\mathbb{E}[\pi_i z_j f(\mathbf{x})]$.

3.3.2 Shapley Values

Shapley value is a concept stemming from cooperative game theory, coined by Shapley [39]. The general idea is to calculate the contribution of each player collaborating in a joint game, and the values are seen as a fair way of distributing profits (or losses) among the participants forming the coalition.

3.3.2.1 Shapley Values in game theory

A coalitional game $\langle N, w\rangle$ is defined by a set of players $N = \{1, \ldots, n\}$ and a function $w : 2^N \to \mathbb{R}$, s.t. $w(\varnothing) = 0$, where 2^N is the power set of N. The set N is called the *grand coalition*, and subsets $S \subseteq N$ are called *coalitions*. The function w is called the *worth*. The problem is to split the worth $w(N)$ among the players in a "fair" way, so the solution to the problem is an operator ϕ mapping the coalitional game to a vector of payoffs $(\phi_1, \ldots, \phi_n)$, corresponding to each player's fair share.

The definition of "fair" is based on four axioms - efficiency, symmetry, dummy, and additivity, which, in contrast to other explanation methods, makes a formal definition of what constitutes a fair distribution [40].

Efficiency axiom: Summing the Shapley value for all players i in the set N should be equal to the output value of all players forming a coalition.

$$\sum_{i \in N}\Phi(w) = w(N) \tag{3.20}$$

Symmetry axiom: The symmetry axiom ensures two players that are interchangeable relative to w receive the same share. If for two players i and j,

$$w(S \cup \{i\}) = w(S \cup \{j\}), \forall S \tag{3.21}$$

when $S \subset N, i, j \notin S$, then

$$\Phi_i(w) = \Phi_j(w) \tag{3.22}$$

Dummy axiom: The dummy axiom ensures a non-contributing player won't receive any share. Thus, if

$$w(S \cup \{i\}) = w(S), \forall S \tag{3.23}$$

when $S \subset N, i \notin S$, then

$$\Phi_i(w) = 0 \tag{3.24}$$

Additivity axiom: Finally, the additivity axiom states that Shapley values can be added together to obtain an overall contribution. For any pair of games w, w':

$$\Phi(w + w') = \Phi(w) + \Phi(w') \tag{3.25}$$

where $(w + w')(S) = w(S) + w'(S)$, $\forall S$.

Shapley showed that using this definition of fairness, the solution ϕ is unique, and is given by the formula

$$\phi_i(w) = \sum_{S \subseteq N \setminus i,\ s=|S|} \frac{s!(n-s-1)!}{n!} \Big(w(S \cup \{i\}) - w(S) \Big). \tag{3.26}$$

So the payoff $\phi_i(w)$ is a weighted average of the difference between the worth when player i is first included and then excluded, taken over all possible coalitions S that do not include player i. In this way, the Shapley Value takes into account how valuable player i is depending on his or her teammates.

3.3.2.2 Shapley Values in Machine Learning

The concept of Shapley Values can be adopted for calculating feature importance in machine learning. The idea is to use Shapley values for measuring the contribution of each feature to an outcome from our model f. In [41] the concept of additive feature attribution methods is introduced, which are XAI methods defined by having an explaining function g that is of the form

$$g(\mathbf{z}') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{3.27}$$

where $\mathbf{z}' \in \{0,1\}^M$, $\phi_i \in \mathbb{R}$, and M is the number of *simplified* input features that map to the domain of f through a function $h_{\mathbf{x}}(\cdot)$ where $\mathbf{x}$ is the point at which f is being explained. The authors then introduce the following properties that are desirable for an explanation:

Property 1. (Local accuracy)

$$f(\mathbf{x}) = g(\mathbf{x}'), \text{ when } \mathbf{x} = h_{\mathbf{x}}(\mathbf{x}') \tag{3.28}$$

Property 2. (Missingness)

$$x'_i = 0 \implies \phi_i = 0 \tag{3.29}$$

Property 3. (Consistency) Let $f_{\mathbf{x}}(\mathbf{z}') = f(h_{\mathbf{x}}(\mathbf{z}'))$, and let $\mathbf{z}' \setminus i$ denote setting $z'_i = 0$. If f and f' are two different models then

$$f'_{\mathbf{x}}(\mathbf{z}') - f'_{\mathbf{x}}(\mathbf{z}' \setminus i) \geq f_{\mathbf{x}}(\mathbf{z}') - f_{\mathbf{x}}(\mathbf{z}' \setminus i) \quad \forall \mathbf{z}' \in \{0,1\}^M \implies \phi_i(f', \mathbf{x}) \geq \phi_i(f, \mathbf{x}), \tag{3.30}$$

i.e. if the contribution of a feature i increases in a second model f', independent of the value of the other features, then the contribution ϕ_i of that feature should not be lower than in the first model f.

Using Shapleys result [39] they then show that for a given model $f : \mathbb{R}^d \to \mathbb{R}$, an instance $\mathbf{x} \in \mathbb{R}^d$, and an input mapping $h_{\mathbf{x}}(\cdot)$, that a function g defined by Equation (3.27) satisfying properties 1-3 is unique, having Shapley values as coefficients:

$$\phi_i(f, \mathbf{x}) = \sum_{\mathbf{z}' \subseteq \mathbf{x}'} \frac{|\mathbf{z}'|!(M - |\mathbf{z}'| - 1)!}{M!} \Big(f_{\mathbf{x}}(\mathbf{z}') - f_{\mathbf{x}}(\mathbf{z}' \backslash i) \Big). \tag{3.31}$$

In this way, the coefficients that explain f's prediction at a given instance are defined in a way that aligns well with our intuition of what a "fair" contribution is.

3.3.3 SHAP

Since a machine learning model generally cannot accept input of varying sizes, the notion of removing parts of the input is problematic. One way to overcome this issue is to retrain the model for every possible collision of features, but this quickly turns computationally expensive, as the number of models that need to be retrained grows exponentially as 2^n.

An option is to approximate the Shapley values in a different way. SHAP is a framework for doing this. In [41] the authors show that the Shapley values in Equation (3.30) can be approximated through the optimization problem in Equation (3.13), using the following weight function:

$$\pi_{\boldsymbol{\xi}}(\mathbf{z}') = \frac{(M-1)}{\binom{M}{|\mathbf{z}'|}|\mathbf{z}'|(M - \mathbf{z}')}, \tag{3.32}$$

where M is the number of features, and $|\mathbf{z}'|$ is the number of nonzero coordinates in the interpretable component vector $\mathbf{z}'$. They call this method KernelSHAP. To show that the Shapley values can be approximated in this way the authors assume that the features are independent and that the model f is linear. These assumptions are worth keeping in mind when engineering features since highly correlated features could potentially worsen the approximations.

3.3.3.1 Sampling

The sampling procedure in SHAP is somewhat simplified compared to that of LIME. In SHAP, the interpretation of $\mathbf{z}'$ is that $z'_j = 0$ means that the feature value has been replaced by the corresponding mean value of this feature, and $z'_j = 1$ means that the original value is still being used. If many features are used, it is not feasible to include all possible coalitions. In the implementation used, the coalitions are sampled starting with masking only 1 feature and all but 1 feature. Then masking 2 features and all but 2 features, and so on.

3.3.4 GraphSVX

GraphSVX [32] is a decomposition technique that considers both node features and graph structure in its explanations, tailoring it to models employing GNNs. Due to the promising performance of GNN in Anti-Money Laundering [1], GraphSVX makes an interesting explanation candidate within the domain.

GraphSVX computes Shapley values extended to graphs and by doing so, captures the "fair" contribution for each feature and node in the local neighborhood of the instance being explained [32]. The method applies to node classification as well as graph classification. For the task of detecting accounts involved in money laundering, GraphSVX will be explained in terms of node classification. The instance to explain is thus a node that represents an account, and the local neighborhood to this node is limited to the computation graph, i.e. the subgraph declared by the k-hop neighborhood used in the GNN.

The process of GraphSVX can be explained in three steps [32]. 1. Constructing a perturbed dataset via a mask generator algorithm, 2. Computing the marginal contribution of the generated masks (or perturbed dataset) towards the prediction using a graph generator, and 3. Learning a carefully defined explanation model on the dataset, and providing it as an explanation.

Constructing perturbed datasets uses a sampling algorithm favoring coalitions that should have high weight, using smart space allocation [32]. Smart space allocation means that 10% of the space is reserved for random coalitions, whilst the rest is allocated for coalitions that should have high weight. Coalitions with high weight are defined as samples with features and nodes with either few or many elements shared with the instance to explain, as it is easier to capture individual effects from the combined effect in these cases. The sampling algorithm creates binary masks M_F and M_N, with 1 indicating a feature, or node, being present, and 0 indicating that the node is canceled out. In the algorithm, P' samples are made with perturbations in the instance to explain feature values, and $P' - P$ samples have perturbations in the connecting neighbors. The algorithm thus reserves 10% to random coalitions and lets the space of 90% favor coalitions with the number of elements close to 0 or the total number of variables. The masks then get stored in a random variable $\mathbf{z} = (M_F||M_N)$.

Next, the joint effect of the selected group of variables on the original prediction is estimated [32]. This is done by passing the masks to a graph generator. Masked-out nodes are dealt with by isolating them completely from the graph. If, due to isolated nodes, the node to explain v is no longer connected to an unmasked node w, the shortest path to it is found by Djikstra's algorithm. The nodes making up the shortest path are reintroduced in the graph but with their feature vector values set to mean values obtained by Monte Carlo sampling, to keep the influence from the nodes low. As for masked-out feature values, they're replaced with the dataset's expected value. The graph generator outputs a perturbed feature matrix and a perturbed adjacency matrix.

The perturbed matrices stored in $\mathbf{z}' = X', A'$ get further passed to the GNN, and the pairs $\mathbf{z}, f(\mathbf{z}')$ are stored into a dataset $\mathcal{D}$ [32]. Finally, the dataset $\mathcal{D}$ fits a surrogate model for which learned parameters ϕ are used as explanations. Ω is called the interpretable domain, and is limited to the set of Weighted Linear

Regression (WLR). The weights are determined so that $\mathbf{z}$ samples with high or low dimensions are given high weight, making it easier to capture individual effects. The loss function is defined accordingly.

$$\mathcal{L}_{f,\pi}(g) = \sum_{\mathbf{z}} (g(\mathbf{z}) - f(\mathbf{z}'))^2 \pi_z \tag{3.33}$$

where,

$$\pi_{\mathbf{z}} = \frac{F + N - 1}{(F + N) \cdot |\mathbf{z}|} \cdot \binom{F + N - 1}{|\mathbf{z}|}^{-1} \tag{3.34}$$

The explanation provided by GraphSVX is thus the solution to the following optimization problem.

$$\phi = \underset{g \in \Omega}{\operatorname{argmin}}\ \mathcal{L}_{f,\pi}(g) \tag{3.35}$$

4
Methods

In Chapter 4, we delve into the methodologies employed in our study to enhance the interpretability and trustworthiness of Graph Neural Networks (GNNs) using eXplainable Artificial Intelligence (XAI) techniques. This chapter serves as a practical guide to the implementation and evaluation of our approach, structured into several key sections.

We begin with Section 4.1, where we explain the data generation process, including its configurations, and the preprocessing steps undertaken to prepare the dataset for training. In Section 4.2 insights into the architecture and training procedures adopted for our GNN model are provided.

Section 4.3 delves into the adaptation of LIME and SHAP techniques to GNNs, outlining the strategies employed to ensure their compatibility and efficacy in interpreting model predictions within the graph context.

The subsequent section, 4.4, is dedicated to the evaluation of our approach. Here, we describe how we systematically assess the coherence, correctness, completeness, compactness, and confidence of our XAI techniques. From measuring the alignment between explanations to evaluating the faithfulness of explanations to the underlying model, this section provides a comprehensive analysis of the interpretability and reliability of our model.

Through the methodologies outlined in this chapter, we aim to provide a robust framework for understanding and enhancing the interpretability of GNNs using XAI techniques, facilitating deeper insights into model decisions and fostering trust in AI systems.

4.1 Data

Since transaction data and customer profiles are privacy-sensitive, using real-world data to create and explore new AML models is challenging. In sensitive domains, synthetic data generation has become a popular approach to creating realistic data, that can be used to develop new systems [12]. One publically available tool to create realistic transaction networks is AMLSim [16], designed specifically for the AML problem. The study is thus performed using a synthetic graph, that includes 100,000 accounts, and approximately 1.7 million transactions, generated by an extended version of AMLSim. The version of AMLSim used has been refined by a team at AI Sweden, as part of a collaborative project named “Federated Learning in Banking” with Swedbank and Handelsbanken [42].

4.1.1 Data generation process

The data is generated in a two-fold process. First, it builds a transaction network structure. The network structure is defined using a specified number of normal-transaction patterns, and correspondingly a specified number of alert patterns. The normal patterns mimic normal transaction behavior described in patterns such as *single*, *fan-out*, *fan-in*, *forward*, *mutual* and *periodical*. The alert patterns injected into the transaction network are the following *gather-scatter*, *cycle*, *fan-out*, *fan-in*, *scatter-gather*, *bipartite*, and *stacked-bipartite*, and their structure can be reviewed in Appendix C. Note that fan-in and fan-outs are both used for normal and alert patterns. During the initialization of the transaction network structure, properties such as timestamps, describing when the transaction should be performed, are also generated.

In the next step, the transaction simulation is performed. That is, each scheduled transaction in the network is realized. In the simulation process, the number of steps is 365, where one step represents one day, yielding a total time frame of one year. In the beginning, all accounts were given an initial balance. Additionally, "days until bank change" and "days until phone change" were sampled. During the simulation, the transactions within the network were carried out, but also transactions from a *source* outside of the bank, and to a *sink* outside of the bank. The transactions from the source represent two types of income. One is a monthly salary, and the other is a smaller value performed randomly. Transactions to the sink are also of two categories, monthly expenses (housing costs, etc.) and general consumption. The sink transactions that mimic general consumption are executed based on the probability of spending money, which is determined by considering the current account balance, and the mean balance of the past 28 days. In each time step, the "days until bank change" and "days until phone change" were updated and a change was recorded when the day occurred, and the "days until change" was resampled. Finally, a transaction log was finalized, including transactions performed inside the network, from the source, and to the sink.

4.1.2 Data generation configurations

Considering the nature of money laundering, where the parties involved do not want to be discovered, the ratio of money launderers in practice is difficult to determine. In this study, we have chosen to work with three different datasets, with different ratios and thus different difficulty levels. One easy dataset, where the ratio of money launderers is 25%, one medium dataset where the ratio is set to 5%, and one difficult dataset, with the ratio of 1%. This is done due to the uncertainty of the real-world ratio, and to gauge the sensitivity of the suggested approach.

In all datasets, the number of accounts was 100,000. The number of accounts was selected to be large enough to mimic a realistic transaction network, and small enough to be computationally efficient for our study. The behavioral properties of each account, such as the amount of money spent or received, were configured in a reverse engineering approach. In [43], it is stated that today's, commonly used, rule-based AML systems exhibit a false-positive rate of 95-98%. For that reason, we tuned the behavioral parameters by training a rule-based model (random forest)

iteratively, until the false-positive rate reached the desired level. For the tuning, we used the medium dataset, where alert patterns were injected so that the percentage of money laundering accounts was 5%. Given this, the behavioral properties were set so that the discrepancy between normal and money-laundering accounts was low to create a realistic scenario. The monthly income distribution was chosen the same for money launderers and normal accounts, while the remaining behaviors were sampled from distributions that differed slightly. Additionally, the money launders were given a cash balance, which was used instead of their account balance with a probability of 70%, when they made transactions to the sink.

4.1.3 Preprocessing

The purpose of the preprocessing step was to utilize the transaction log to generate and construct training and test sets, resulting in two outputs: a file containing each node with its corresponding feature vector $\mathcal{X}$ and a file with information on all edge pairs $\mathcal{E}$.

The initial task was to determine how to split the training and test sets. Unlike statistical models that handle data records i.e. the accounts independently, our model processes entire networks, making a traditional 80-20 split of the accounts unsuitable. Therefore, we adopted a different approach by splitting the data along the time dimension. This method ensures that the training and test sets cover equally many days, making the size of the features in the two sets comparable, thus allowing for normalization using the same mean and standard deviation. Further on, we chose to have a slight overlap in the transaction data used for the training versus the test set. Although this overlap introduces some information leakage between the training and test sets, this choice is motivated by two key aspects.

The feature vectors for all nodes will differ between the training and test sets since the information is aggregated over time, provided there are transactions for all accounts in the non-shared time period. Additionally, this approach aligns with how banks would realistically manage such data. Banks aim to use as much data as possible for training (e.g., one year's worth) without waiting another year to predict newly recorded data. To maintain sets that are equal in size, it is necessary to include some of the transaction data from the training set in the test set. An overlap of 80% between the training and test sets would result in approximately 30 days of unseen data in the test set. It is reasonable to assume that banks would prefer to retrain their models continually to stay updated with recent money laundering behaviors. Thus, retraining the model monthly appears to be a feasible and practical approach.

This preprocessing strategy helps mitigate the risk of information leakage while ensuring that the model is trained on the most recent and comprehensive data available, closely mirroring real-world practices in financial institutions.

Continuing, features were extracted from the transaction logs. The features chosen can be categorized into three types, *network*, *spending*, and *other account specific* features. The network features were obtained using the transactions in the network, excluding the ones from the source and to the sink. As for the spending features, those were extracted using only the sink transactions, excluding any cash transactions as they're not typically visible to the bank. The other account-specific

features were two, the number of phone changes, and the days in bank. As a result, 31 features were constructed, all of which can be found in Table 4.1.

Table 4.1: Feature description

Feature	Description
sum	The sum of all network transactions in and out from an account
in_sum	The sum of all network transactions in to an account
out_sum	The sum of all network transactions out from an account
mean	The mean of all network transactions in and out from an account
in_mean	The mean of all network transactions in to an account
out_mean	The mean of all network transactions out from an account
median	The median of all network transactions in and out from an account
in_median	The median of all network transactions in to an account
out_median	The median of all network transactions out from an account
std	The standard deviation of all network transactions in and out from an account
in_std	The standard deviation of all network transactions in to an account
out_std	The standard devation of all network transactions out from an account
max	The max value transaction of all network transactions in and out from an account
in_max	The max value transaction of all network transactions in to an account
out_max	The max value transaction of all network transactions out from an account
min	The min value transaction of all network transactions in and out from an account
in_min	The min value transaction of all network transactions in to an account
out_min	The min value transaction of all network transactions out from an account
count_in	The count of all network transactions in to an account
count_out	The count of all network transactions out from an account
n_unique_in	The number of unique accounts making transactions in to an account
n_unique_out	The number of unique accounts an account makes transacions to
count_days_in_bank	The number of days an account has been in the bank
count_phone_changes	The number of phone changes an account has made
sum_spending	The sum of all sink transactions an account has made
mean_spending	The mean of all sink transactions an account has made
median_spending	The median of all sink transactions an account has made
std_spending	The standard deviation of all sink transcations an account has made
max_spending	The max value sink transaction an account has made
min_spending	The min value sink transaction an account has made
count_spending	The number of sink transaction an account has made

4.2 Model architecture and training

In this thesis, we opt for graph attention networks (GATs) among several graph neural network (GNN) architectures. The specific architecture utilized is depicted in Figure 4.1, and takes the node feature matrix X and the edge indices $\mathcal{E}$ as input, and outputs logits for the two classes *not money launderer* and *money launderer*.

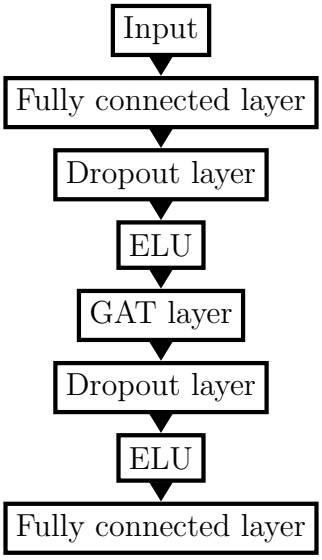

Figure 4.1: Schematic drawing of GAT architecture used. The first and second fully connected layers work as preprocessing and postprocessing layers, respectively.

When selecting this architecture, we initially experimented with a single GAT layer and later explored augmenting the model with fully connected (FC) layers while varying the number of GAT layers. We noted a decrease in performance, measuring the F_1 score and the average precision, when including additional GAT layers, leading us to opt for a single layer. This decline could possibly be caused by oversmoothing [44]. Regarding FC layers, we observed enhancement by incorporating one layer before (as a preprocessing step) and one after (as a post-processing step). However, further additions of FC layers did not improve performance. To mitigate overfitting and introduce non-linearities, we integrated dropout layers and ELU activation functions, respectively, into the model.

For training, we employed the Adam optimizer and class-balanced loss. Despite the slightly superior average precision achieved with cross-entropy loss in some instances, we favored class-balanced loss. This choice yielded a model with enhanced precision at high recall, deemed crucial for minimizing false positives. The final model training utilized the Optuna library [45]. Details of hyperparameter search intervals, resulting hyperparameters, and performance metrics are presented in Appendix A.

4.3 Adaption of LIME and SHAP to GNNs

There are various implementations of LIME and SHAP that are specifically designed for different types of models[1][2]. Since there is no implementation specifically designed for GNNs, we used the *LIME tabular* and *Kernel SHAP* implementations in this study and adapted them to these models.

Since both LIME tabular and Kernel SHAP call the model's forward function

[1]https://github.com/marcotcr/lime
[2]https://github.com/shap/shap

with a single node feature vector, and the GAT's forward function takes a 1-hop subgraph $G_1(i)$ around the node i being explained as input, we implemented a modified forward function, `forward_NFV_input`, where NFV stands for *node feature vector*. This forward function takes as input a node feature vector $\mathbf{x}_u$ instead of $G_1(i)$. The way it works is that $\mathbf{x}_i$ is overwritten by $\mathbf{x}_u$ on all features j where $(\mathbf{x}_i)_j \neq (\mathbf{x}_u)_j$. This is also done for the nodes in the subgraph as well. Then the GNN is evaluated on $G_1(i)$ with the modified node feature vectors $\mathcal{X}_1(i)$. The reason we overwrite the neighboring nodes as well is because this ensures that the GNN outputs the same value on all nodes i when all features are replaced by their respective mean values, which is important for SHAP to ensure that the average prediction on all nodes is the same. Since GNNs are quite dependent on the neighboring nodes, this would not be the case if we only overwrite i's node feature vector. As a consequence, the feature importance obtained from LIME and SHAP reflects the significance of the features in the node's neighborhood, not just the features of the node being explained.

4.4 Evaluation

With the growth of possible XAI methods, the demand for XAI evaluation metrics is increasing [46, 20, 14]. As for today, there's no clear consensus on how explainability should be evaluated in a structured way[2]. Many studies provide theoretical justifications, or user studies, for their methodology but lack in performing any quantitative analysis. This is mainly because many XAI techniques are built on entirely different theoretical backgrounds, making it difficult to share a common evaluation approach.

In [2], more than 300 research papers on XAI are surveyed, and the different evaluation approaches in the field are summarized. They found that 1 out of 3 papers evaluate exclusively with anecdotal evidence, and 1 in 5 papers conduct a user study to evaluate. Out of all the models to be explained, 75% of them were deep learning models, and the most common explanation type was feature importance. Further on, the authors created a framework for evaluating explainability techniques regarding twelve desirable properties an XAI technique should possess, starting with the prefix "co". They showed that the most common quantitative evaluations according to their 12 properties were coherence and output-completeness. Based on the Co-12 properties, we appointed five properties most relevant to our study. The exact approach can be defined under each category.

4.4.1 Coherence

Coherence measures how well the explanations align with prior or other beliefs than the explanation itself. Two approaches are presented in [2], alignment with domain expertise and alignment with other XAI techniques. Exploring domain expertise alignment would be interesting, but it wasn't feasible for the time frame given. Our option was thus to study the alignment between different XAI techniques. Alignment between two explainers that rely on separate theoretical backgrounds serves as a positive indicator. However, the lack of alignment shouldn't necessarily dismiss any

of the compared explainer's performance, one of them may be poor and thus there wouldn't be any agreement. The coherence is measured in four different metrics, *feature agreement*, *rank agreement*, *sign agreement* and *signed rank agreement*.

The feature agreement is given by

$$\text{FeatureAgreement}(E_a, E_b, k) = \frac{|T_f(E_a, k) \cup (T_f(E_b, k)|}{k} \tag{4.1}$$

where E_a and E_b are the explanations provided by explainers A and B, and T_f returns the top k features of the explanation E. Note that the top k features are extracted using the absolute value of the feature importance, sorting from largest to smallest. It is implied from the explainers that the largest importance in absolute terms is what's most important for a certain prediction, which makes it reasonable to compare the top k features using the absolute value. The feature agreement measures how many k features are listed as the top k in both explainers.

The next metric is the rank agreement, defined as

$$\begin{aligned}&\text{RankAgreement}(E_a, E_b, k)\\ &= \frac{|\bigcup_{s \in S} \{s | s \in T_f(E_a, k) \wedge s \in T_f(E_b, k) \wedge \text{rank}(E_a, s) = \text{rank}(E_b, s)|}{k}\end{aligned} \tag{4.2}$$

and shows how well the ordering of the top-k features from different explainers aligns. Here, T_f is defined as before, and $\text{rank}(E, s)$ returns the position of the feature s according to the explainer E, and S is the complete set of features in the data. The rank agreement is more restrictive than the feature agreement since it considers the presence of features in the top k features and their ordering.

Next, the sign agreement is given by

$$\begin{aligned}&\text{SignAgreement}(E_a, E_b, k)\\ &= \frac{\left|\bigcup_{s \in S} \{s | s \in T_f(E_a, k) \wedge s \in T_f(E_b, k) \wedge \text{sign}(E_a, s) = \text{sign}(E_b, s)\}\right|}{k}\end{aligned} \tag{4.3}$$

and measures the fraction of top k features that share the same sign in both explainers. Here, $\text{sign}(E, s)$ returns thee sign of the feature s according to the explanation E. Having a feature listed as highly important in two explainers, but with opposite signs is a clear contradictory indication, which is why this measure is relevant.

Lastly, the final measure used in the coherence evaluation is signed rank agreement, which is defined according to

$$\begin{aligned}\text{SignedRankAgreement}(E_a, E_b, k) = &\frac{\left|\bigcup_{s \in S} \{s | s \in T_f(E_a, k) \wedge s \in T_f(E_b, k)\right.}{k}\\ &\frac{\left.\wedge\ \text{sign}(E_a, s) = \text{sign}(E_b, s) \wedge \text{rank}(E_a, s) = \text{rank}(E_b, s)\}\right|}{k}\end{aligned} \tag{4.4}$$

The signed rank agreement is the strictest measurement since it considers both the positioning and order of the features and their sign.

4.4.2 Correctness

The correctness metric intends to measure how faithful or truthful the explanation is, with respect to the model being explained [2]. It addresses whether or not the explanations adhere to the "true" black box behavior, in a global or local context. The correctness principle emphasizes how an explanation should be "nothing but the truth". In the context of AML, the principle of correctness appears highly desired. To ensure the AI system is not discriminatory, and reason in a proper way, it is vital that the explanations provided show the true functioning of the model and nothing else. With the chosen evaluation technique, we utilize working with a simulation tool and attempt to extract the "ground truth" information making an account a money laundering account.

4.4.2.1 Feature importance

The idea to evaluate feature importance in terms of correctness is based on creating a ground truth feature importance for each node. This was done by reproducing the data generation process and extracting duplicate feature vectors for each node. We define the duplicate outputs as the $\mathcal{D}_o$ for the original dataset, and $\mathcal{D}_w$ for the dataset without money laundering. $\mathcal{D}_w$ differs from $\mathcal{D}_o$ in the sense that all behaviors related to money laundering are excluded. The alert patterns are removed, and the accounts with money laundering use the normal distribution when sampling transactions, instead of the distribution for money launderers. In that way, we obtain a dataset that shows a scenario where the money launderers are now engaging only in normal behaviors.

One additional adjustment was made to tailor the ground truth importance according to the graph structured data. Since the GNN makes use of the graph structure to embed each node, the datasets were first modified, by utilizing the attention mechanism in a GAT to make a new feature vector representation for each node. That is, all feature vectors were replaced with a new feature vector which was a weighted sum of all feature values in the computation graph, weighted by their attention:

$$\tilde{\mathbf{x}}_v = \sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{x}_u, \tag{4.5}$$

where α_{vu} are the attention weights from the single GAT-layer of the model used. This was done to ensure the graph structure was considered also in the ground truth importance.

Once two complete datasets were created, a distance metric between the two versions was introduced, to get comparable importance for each feature. Two approaches were defined and explored, involving different theories. The two explored strategies are explained in the subsections below.

In the approach called *standard deviation*, the without money laundering dataset is subtracted from the original dataset, and a matrix containing the difference for each node is obtained. The difference vector is divided by the standard deviation from the training set to allow for comparison between different features. The final importance is given by the absolute value of the differences divided by the standard deviation. The absolute value is used to represent that all features

with differences should have a positive contribution to predicting money laundering. This is motivated by knowing that the feature vector would look the same in the dataset without money laundering as the original one if an account weren't involved in laundering money. All feature changes occur only if an account has been laundering money and can thus be seen as having a positive contribution. The feature importance is calculated according to:

$$FI^G_{STD} = \frac{abs(\mathcal{D}_o - \mathcal{D}_w)}{\sigma} \tag{4.6}$$

where FI^G_{STD} is the feature importance, and σ is the standard deviation.

The second approach, built on the log-likelihood ratio, is slightly different. In this approach, each feature value in the datasets is fitted to the LLR of belonging to the normal distribution vs. the money laundering distribution. That is, first the LLR ratio is found using the original dataset $\mathcal{D}_o$, and then the dataset without money laundering behavior $\mathcal{D}_w$. The final importance is given by the subtraction of the two LLRs. That is, no difference in features between the two datasets will give the importance of 0, while all changes indicate a positive or negative impact.

Formally, for a node with node feature vector $\mathbf{x}$ we define the feature-wise log-likelihood ratio as

$$\mathrm{LLR}(\mathbf{x}) = \Big(\log \frac{f_{\mathcal{N}}(x_1; \mu_{n,1}, \sigma_{n,1})}{f_{\mathcal{N}}(x_1; \mu_{s,1}, \sigma_{s,1})}, \ldots, \log \frac{f_{\mathcal{N}}(x_d; \mu_{n,d}, \sigma_{n,d})}{f_{\mathcal{N}}(x_d; \mu_{s,d}, \sigma_{s,d})} \Big), \tag{4.7}$$

where μ_n, σ_n and μ_s, σ_s are the mean and standard deviation of the normal and money laundering distributions, respectively, and $f_{\mathcal{N}}(\,\cdot\,; \mu, \sigma)$ is the probability density function of $\mathcal{N}(\mu, \sigma)$. We note that a positive LLR ratio indicates that it is more likely to belong to the normal distribution, and a negative LLR ratio implies the feature value appears more likely to belong to the money laundering distribution. The LLRs using $\mathcal{D}_o$ and $\mathcal{D}_w$ are then

$$\mathrm{LLR_o} = \mathrm{LLR}(x), \quad x \in \mathcal{D}_\mathrm{o}, \tag{4.8}$$

$$\mathrm{LLR_w} = \mathrm{LLR}(x), \quad x \in \mathcal{D}_\mathrm{w}. \tag{4.9}$$

The ground truth feature importance for this node is then computed as

$$FI^G_{LLR} = \mathrm{LLR_o} - \mathrm{LLR_w}. \tag{4.10}$$

Finally, once having a ground truth importance, the task is to compare it to the importance given by the explainers. This is done by looking at the feature agreement, defined similarly as in (4.1). Feature agreement between explainers and the ground truth is given by

$$\text{Feature Agreement}(E_a, E_g, k) = \frac{|T_f(E_a, k) \cup (T_f(E_g, k)|}{k} \tag{4.11}$$

where E_g, is the ground truth explanation, and E_a is the explanation given from an explainer. Thus, the feature agreement measure measures what percentage of the top k importance agrees with the ground truth. When we compare the importance of the LLR approach with the explainers, we use their absolute value. This is because

both the LLR approach and the explainers give positive and negative values and the same reasoning as for the coherence metric. However, the comparison with the STD approach needs a slight adjustment. Since this method only provides positive ground truth importance, we won't use the absolute value when sorting in the comparison, only the actual value.

4.4.2.2 Detailed analysis on feature importance

Note that the correctness measure explored through this method has some limitations. It relies heavily on the ground truth being based on accurate assumptions. Otherwise, a potential poor agreement may be caused by the ground truth being based on inaccurate assumptions or because the explainer studied is inaccurate. To get insight into whether the ground truth appears correct or not, complementary data can be extracted to analyze the result further. Below, we outline this procedure for the LLR-based approach.

To provide more detail into the measure of feature agreement between the ground truth and the explainers, we define seven different types of *characteristics* a feature can have. This will help identify which characteristics are prioritized by different explainers in comparison to the ground truth. The characteristics are extracted by calculating the LLR using the original dataset, and the dataset without money laundering respectively. A feature may have a positive LLR, meaning it is more likely to belong to the normal distribution or a negative LLR, suggesting it more likely belongs to the money laundering distribution. The different characteristics are thus based on calculating the LLR for a feature in the original dataset and the dataset without money laundering. The description of the characteristics, and their sign using the LLR approach, are presented in Table 4.2. In the table, n denotes a feature that is more likely to belong to the normal distribution, while s denotes a feature that is more likely to belong to the money laundering distribution, which makes it suspicious. Additionally, a feature is said to be typical of suspicious behavior, if the LLR is negative and typical of normal behavior if the LLR is positive.

To explore why a potential disagreement between the explainers and the ground truth occurs, a check is done using the top 10 important features. The top 10 features are picked in two ways, sorting by absolute value and sorting by positive to negative. These two ways of sorting provide different information. First, the absolute value is used when reviewing feature agreements. Hence, this sorting provides direct detail relevant to how the feature importance is measured. Additionally, sorting from the largest to the smallest provides further detail on where our explainers and the ground truth differ. In one way, this can be seen as more restrictive, by sorting from positive to negative, we get information on the characteristics of the differences when limiting our attention to the most positive feature importance by the constructed ground truth and the explainers respectively.

For each explainer, and each dataset, 500 predicted money laundering nodes are reviewed at random, and their top 10 features are compared against the ground truth's top 10. For each of the reviewed nodes, we use the top-10 features to compute a feature intersection set $\mathcal{F}_{\bigcap} = \mathcal{F}_G \bigcap \mathcal{F}_E$, where $\mathcal{F}_G$ and $\mathcal{F}_E$ are the top 10 features of the ground truth and the explainer, respectively. Features only important according

Table 4.2: Characteristics, their descriptions, and their sign using the LLR approach

Characteristics	Description	Sign
$n \uparrow n$	In the original dataset, the feature appears typical of normal behavior. In the dataset without money laundering, the feature remains typical of normal behavior but slightly more so.	+
$n \downarrow n$	In the original dataset, the feature appears typical of normal behavior. In the dataset without money laundering, the feature remains typical of normal behavior but slightly less so.	-
$n \rightarrow s$	In the original dataset, the feature appears typical of normal behavior. In the dataset without money laundering, the feature appears typical of suspicious behavior.	-
$s \rightarrow n$	In the original dataset, the feature appears typical of suspicious behavior. In the dataset without money laundering, the feature appears typical of normal behavior.	+
$s \downarrow s$	In the original dataset, the feature appears typical of suspicious behavior. In the dataset without money laundering, the feature remains typical of suspicious behavior but slightly less so.	+
$s \uparrow s$	In the original dataset the feature appears typical of suspicious behavior. In the dataset without money laundering the feature remains typical of suspicious behavior but slightly more so.	-
no change	There is no change in the feature value between the original dataset and the dataset without money laundering.	0

to the ground truth are stored in a feature set $\mathcal{F}_{G\setminus E} = \mathcal{F}_G \setminus \mathcal{F}_E$. Features only important according to the explainer are stored in a feature set $\mathcal{F}_{E\setminus G} = \mathcal{F}_E \setminus \mathcal{F}_G$. For each feature set, the features' characteristics are counted. The count is finally summarized over all the 500 suspicious nodes. This detail will provide insight into what characteristics are more important for different explainers versus the ground truth.

4.4.2.3 Node importance - GraphSVX

From GraphSVX we get node importance in addition to feature importance. The intuition of node importance is that a node will contribute positively towards predicting that the node is a money launderer if it is also part of the money laundering scheme. To verify whether this is the case or not we can compare nodes with positive importance with their ground truth labels.

To do this in a systematic way we use the following procedure for each node i that we explain. We introduce a threshold $\tau \in \{0, 0.01, \ldots, 0.99, 1\}$. For each τ, we compare the nodes in the explanation with Shapley values that are greater than this threshold with their ground truth labels. By dividing the number of neighboring nodes that are labeled as money laundering, $m_i^{(\tau)}$, by the number of nodes that have a Shapley value above this threshold, $c_i^{(\tau)}$, we compute the value

$$a_i^{(\tau)} = \begin{cases} m_i^{(\tau)}/c_i^{(\tau)}, & \text{if } c_i^{(\tau)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

which measures the accuracy of how well the Shapley values predict if neighboring nodes are also money launderers. Note that for some nodes c_i may be zero above some threshold, and we set a_i to zero. In practice, the value $a_i^{(\tau)}$ is set to in this case does not matter and is consequently only a matter of formality.

Having calculated $a_i^{(\tau)}$ for all nodes i that were predicted as money launderers we compute the average value at each threshold:

$$\bar{a}^{(\tau)} = \frac{\sum\limits_{i \in \mathcal{C}_{\neq 0}^{(\tau)}} m_i^{(\tau)}}{\sum\limits_{i \in \mathcal{C}_{\neq 0}^{(\tau)}} 1}, \tag{4.13}$$

where $\mathcal{C}_{\neq 0}^{(\tau)} = \{i : c_i^{(\tau)} \neq 0\}$. Note that we only compute this average based on $a_i^{(\tau)}$ for nodes i where there is at least one node importance that is above the threshold. The only negative effect of this is that the results at larger thresholds are based on a smaller number of samples. We compute the percentage of nodes i that have at least one node importance above τ in the following way

$$\bar{c}^{\tau} = \frac{\sum\limits_{i \in \mathcal{C}_{\neq 0}^{(\tau)}} 1}{n}, \tag{4.14}$$

where n is the number of accounts predicted as money laundering.

To summarize the results we finally plot $\bar{m}^{(\tau)}$ and $\bar{c}^{(\tau)}$ against the threshold values τ. Large values of $m^{(\tau)}$ will indicate that the interpretation of the node importance aligns well with our intuition.

4.4.3 Completeness & Compactness

Completeness addresses the extent to which an explanation explains the black box model. In [2] two types of completeness are mentioned, reasoning-completeness and

output-completeness. In this thesis, we focused exclusively on output-completeness, which aims to measure how well the explanation does not miss any information that explains the decision. Since it may be of interest to the banks to not report all features to the investigator, but rather for example the top 10 features, it is of interest to investigate how complete the explanation is when the other features are excluded, i.e. how much excluding the remaining features would impact the model's prediction. Ideally, only a few features would carry high importance, and excluding the remaining features should not have a significant impact on the model's decision.

In light of this, it is also interesting to investigate the compactness of the explanation, i.e. how many features, on average, are necessary to explain the black-box model. To reduce the complexity of the explanation, it is desirable to report only a small number of features. Compactness measures how many features, on average, must be included in the explanation to not significantly alter the output prediction.

We aim to measure both completeness and compactness through a method called incremental deletion check. This method aims to measure the impact of the features by incrementally excluding them from the input and measuring how much it affects the output. There is no obvious way to exclude a feature from the input since the model cannot handle arbitrarily missing inputs. We approximate the effect of excluding a feature by replacing the feature value with the mean value of that feature in the training set. The argument for this is that the mean value should be primarily determined by the normal accounts, because of the class imbalance, and, should make it more uninformative for predicting that the account is a money launderer. Note also that the order in which features are removed can affect the result since the features are more or less correlated.

Furthermore, we also need to take the interpretation of the feature importance into account: Because of our adaption of LIME and SHAP to GNNs, the feature importance reflects the importance of that feature in the whole subgraph around the node v that is being explained. When removing a feature during the incremental deletion check, for LIME and SHAP, the feature is therefore removed for all nodes in the neighborhood of v as well. For GraphSVX, on the other hand, the feature importance only says something about the features on v, since all other nodes are masked out during the sampling. Therefore, the feature is only removed on v during the incremental deletion check. Finally, it should be noted that during the incremental deletion check we make no distinction between feature and node importance, i.e. we order all importances irrespective of their type. To remove a neighboring node u, we copy the node feature vector of v after having possibly removed some features. There may be other possible ways to remove nodes. This approach at least excludes the information on that node, making it less informative, which seems reasonable. Also it mimics the way that GraphSVX removes nodes during sampling.

The algorithm for the incremental deletion check follows the following procedure:

Definitions. Let $\hat{y}_i$ be the original class probability for node i, before removing any features, and let FI_i denote the vector of feature importances for node i for an explainer, where FI_{ij} is the feature importance of the j:th feature.

Furthermore, let $I(FS_i, \tau) = \{j : FI_{ij} < \tau\}$, i.e. the set of indices j corresponding to the features that have a feature importance less than the threshold τ. Finally $K_i(\tau) = |I(FS_i, \tau)|$.

1. For every node to explain i do the following: Starting from the lowest τ, successively determine the features that have an importance lower than the threshold value, $I(FS_i, \tau)$, and compute class probability of the GNN $\hat{y}_i^{(\tau)}$ after replacing the values of the features in $I(FS_i, \tau)$ by their respective mean value, as calculated over all samples in the training set. Then compute the difference $(\Delta\hat{y})_i^{(\tau)} = \hat{y}_i - \hat{y}_i^{(\tau)}$. Also compute $K_i(\tau)$. In this way, we will successively remove features starting with the most negative importance and ending with the most positive importance, recording the difference in output probability and the number of features removed.

2. For each threshold τ, compute the average $\overline{(\Delta\hat{y})^{(\tau)}}$ over all samples i. Also, compute the average number of features removed $\overline{K(\tau)}$ at each threshold.

3. Plot the average difference in class probability $\overline{(\Delta\hat{y})^{(\tau)}}$ against the average number of features removed $\overline{K(\tau)}$.

Intuitively $\overline{(\Delta\hat{y})^{(\tau)}}$ should change the most at the edges of the plotted interval, as it is the features corresponding to the most negative feature importance (the left side of the interval) and the most positive feature importance (the right side of the interval) that should have the greatest impact on the result. Features with low feature importance should not alter the class probability significantly, and we, therefore, expect to see a plateau around the value of $\overline{K(\tau)}$ corresponding to $\tau = 0$. The size of this plateau, i.e. the length along the x-axis in the plot, would indicate how many features on average need to be considered in the explanation.

To perform the incremental deletion check we considered two approaches: i) starting by deleting features from the smallest feature importances based on the absolute value, or starting from the largest. ii) starting by deleting features from the most negative feature importance to the most positive feature importance, or the other way around. Using approach i), it could happen that we first remove a feature with negative importance, making Δy decrease, and then remove a feature with positive importance, making Δy increase. So after having removed, say 5 features, the difference Δy could still be around 0, making it seem like no features have had any impact, when in reality they have had both positive and negative impact, only net impact is zero. We therefore opted for using approach ii) instead should therefore give a more fair result. However, one should still note Δy still depends on the order that features are removed. That is why we chose to remove them both from negative to positive feature importance, as well as from positive to negative feature importance.

4.4.4 Confidence

In both LIME and GraphSVX we solve similar optimization problems given in Equation (3.13) and Equation (3.35) from which an R2 value can be obtained. The R2 value, also called the coefficient of determination, measures how good a fit the surrogate model has based on the ratio of the residual sum of squares and the total sum of squares. Each time an explanation is calculated, we extract the R2 value from the respective methods. We then summarize these values in a box plot, to compare which method is generally more confident as well as gauging the spread of confidence over the samples explained.

Since SHAP solves the same optimization problem as LIME, it should be possible to extract the R2 value from SHAP as well. Unfortunately, the solver chosen in the default implementation did not return the R2 value. Due to time limitations, we choose not to make any adjustments to the source code, and we leave this experimentation for future work.

5
Results

5.1 Alert patterns recognized by model

In the tables 5.1, 5.2, and 5.3, the frequency of different alert patterns recognized by the model are listed, one table for each dataset. The different types of alert patterns can be found in the C. It is noticeable that overall, the model employed identifies almost all accounts participating in the alert patterns bipartite and stack. The model shows the worst performance for the cycle pattern across all datasets but with a decreasing percentage as the difficulty level is higher. For scatter-gather and fan-in, the model is performing consistently across the different datasets, placing them as the third and fourth most recognized pattern consistently. For fan-out and gather-scatter, fan-out is more easily recognized in the MID and HARD datasets, whilst for the EASY dataset the opposite is true.

Table 5.1: Number of accounts per alert pattern in the dataset EASY25, sorted by the percentage

Model type	Percentage	Accounts found	Total number of accounts
bipartite	0.980908	2723	2776
stack	0.954841	2643	2768
scatter_gather	0.954757	2258	2365
fan_in	0.777504	3152	4054
gather_scatter	0.675746	1563	2313
fan_out	0.668250	2814	4211
cycle	0.667134	2858	4284

Table 5.2: Number of accounts per alert pattern in the dataset MID5, sorted by the percentage

Model type	Percentage	Accounts found	Total number of accounts
bipartite	0.973070	542	557
stack	0.951175	526	553
scatter_gather	0.889362	418	470
fan_in	0.777778	637	819
fan_out	0.728155	600	824
gather_scatter	0.559471	254	454
cycle	0.333720	288	863

Table 5.3: Number of accounts per alert pattern in the dataset HARD1, sorted by the percentage

Model type	Percentage	Accounts found	Total number of accounts
bipartite	0.928571	104	112
stack	0.927928	103	111
scatter_gather	0.791667	76	96
fan_in	0.698795	116	166
fan_out	0.471698	75	159
gather_scatter	0.452632	43	95
cycle	0.140244	23	164

5.2 Coherence

In this section, we review the result from comparing the output between different explainers. Each subsection covers a certain type of agreement and contains two figures. The first figure consists of three plots, where each plot combines all pairwise comparisons for a specific difficulty on the dataset. The second figure, on the other hand, consists of three plots where each plot combines the pairwise comparisons of two explainers on the three different datasets to investigate the impact of varying the difficulty of the dataset.

For larger k:s the feature agreement tends towards one, which makes sense if one considers what happens at $k = 31$: in this case, both top-k sets will contain all features and the feature agreement will consequently be 1. Since the agreement in our results is generally quite low, the area of interest will mostly be between $k = 1$ and $k = 15$. For larger k:s the agreement seems to increase linearly along a straight line, which is the same as what is obtained when comparing two explainers that output random feature importance.

In summary, the feature agreement in all pairwise comparisons lies between 40% and 60% for $5 \leq k \leq 10$. Furthermore, the signed agreement is between 40% to 50% at $k = 10$. The rank and signed rank agreement was fairly low with values roughly between 10% and 20% when $k = 5$. For feature agreement, rank agreement, and sign agreement, the only clear pattern of varying the difficulty of the dataset was observed for SHAP and GraphSVX, in which case the agreement decreased when the dataset became more difficult. In all other cases, no clear pattern was observed when varying the difficulty of the dataset.

5.2.1 Feature agreement

In feature agreement, we look at the percentage of the top-k features that are shared between two explainers. On the EASY dataset, we see from Figure 5.1 that SHAP and GraphSVX have better feature agreement for $k \leq 10$. For MID it is LIME and SHAP that have slightly better agreement for $k \leq 6$. Finally, for the HARD dataset, LIME and SHAP seem to have better agreement for $k \leq 5$. From these results, it seems like LIME and GraphSVX have the worst agreement, with the other pairs having slightly higher agreement depending on which dataset is being used.

Overall, the agreement hovers around 40% to 60% for all pairwise comparisons for $4 \leq k \leq 15$, with higher agreement closer to $k = 15$.

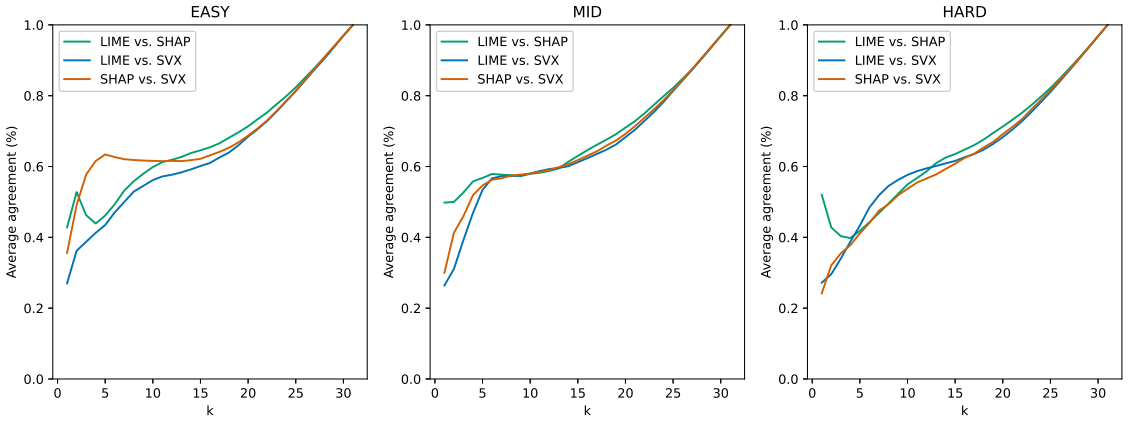

Figure 5.1: Pairwise feature agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

In Figure 5.2 we see that there is no clear consistent effect of using a dataset of different difficulty, however, agreement using the HARD dataset seems to be low in all cases. For LIME and SHAP there seems to be best agreement on the MID dataset. For LIME and GraphSVX the agreement is only slightly better on the MID dataset. Comparing SHAP and GraphSVX there is best agreement on the EASY dataset, followed by the MID dataset.

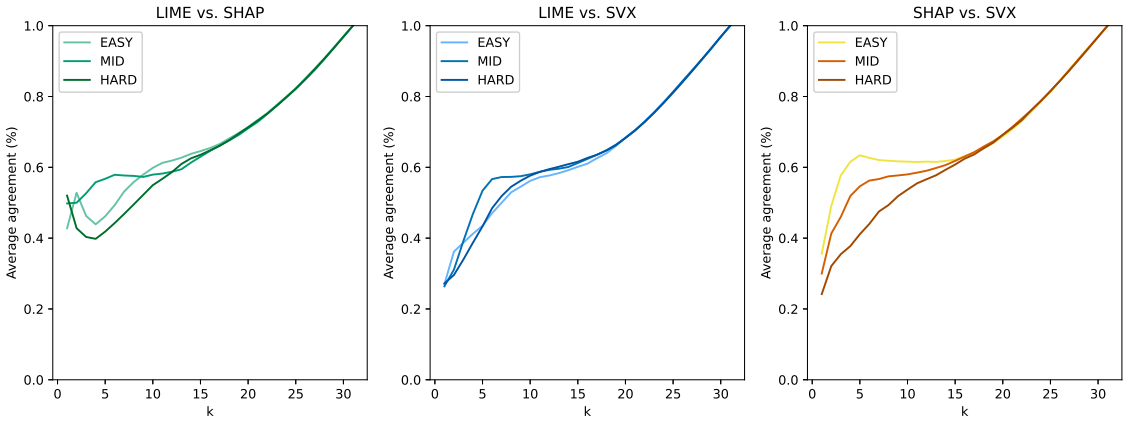

Figure 5.2: Pairwise feature agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

5.2.2 Rank agreement

In feature agreement, we look at the percentage of the top-k features between two explainers that are the same under the added condition that the two features also have the same rank. In Figure 5.3 we see that the rank agreement across all comparisons and datasets is generally quite low.

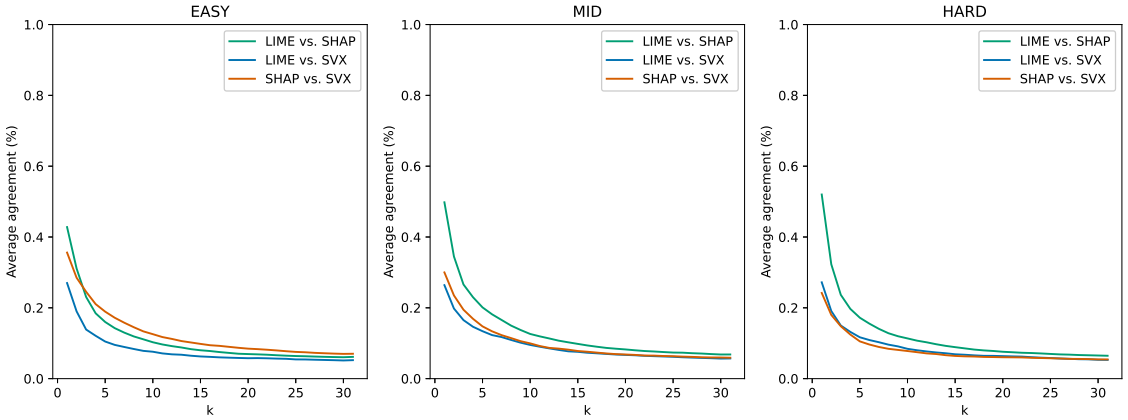

Figure 5.3: Pairwise rank agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

In Figure 5.4 we see that there is no significant impact of using different difficulties on the dataset, except in the case of SHAP and GraphSVX, which indicated slightly that increasing the difficulty makes the rank agreement worse. Having 10% rank agreement at $k = 10$ means that there is on average 1 feature that is ranked as equally important between the two explainers.

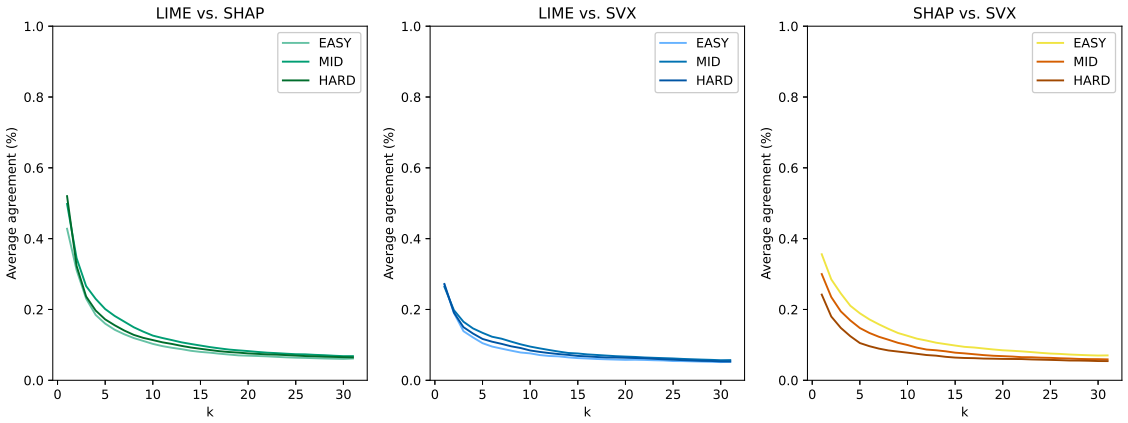

Figure 5.4: Pairwise rank agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

5.2.3 Sign agreement

In sign agreement we look at the percentage of the top-k features that are shared between two explainers under the added condition that the two features also have the same sign. In Figure 5.5 we see that the sign agreement between SHAP and GraphSVX is the lowest on all datasets, except between $k = 3$ and $k = 5$ for the EASY dataset. The other pairwise comparisons have between 10 to 20 percent higher agreement. On the MID and HARD datasets, LIME and SHAP superior for $k \leq 5$, while LIME and SVX are superior for $k > 5$. On EASY they are essentially equal for $k > 5$.

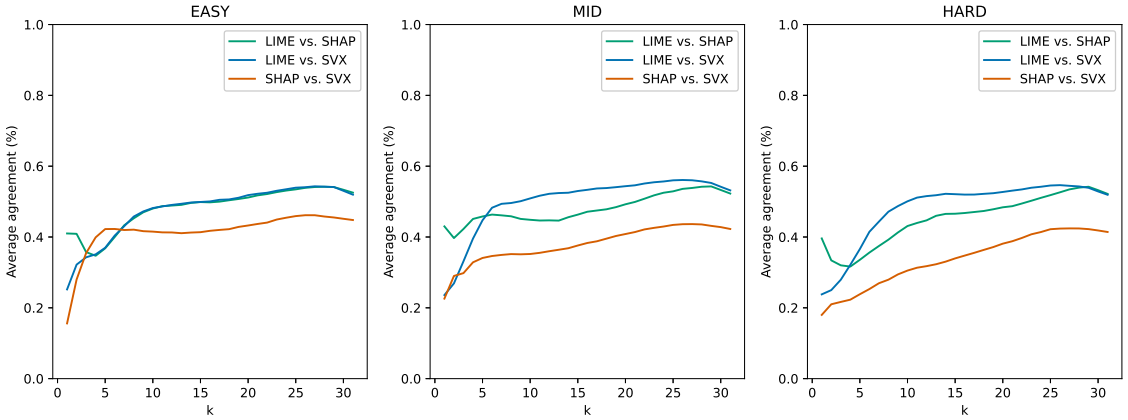

Figure 5.5: Pairwise sign agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

The effect of varying the dataset is shown in Figure 5.6. Similar to rank agreement, there is no indication that the difficulty of the dataset has a significant impact on the results, except in the case of SHAP and GraphSVX, where the agreement decreases when the difficulty increases.

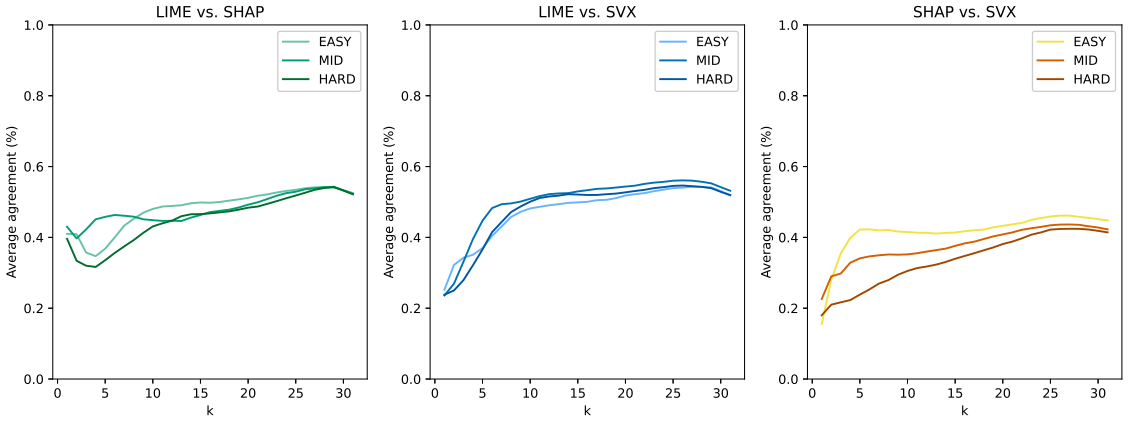

Figure 5.6: Pairwise sign agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

5.2.4 Signed rank agreement

In the signed rank agreement we look at the percentage of the top-k features that are shared between two explainers under the added condition that the two features also have the same sign and rank. For small $k \leq 5$ there is between 10% and 20% agreement as shown in Figure 5.7. The highest agreement is consistently found between LIME and SHAP.

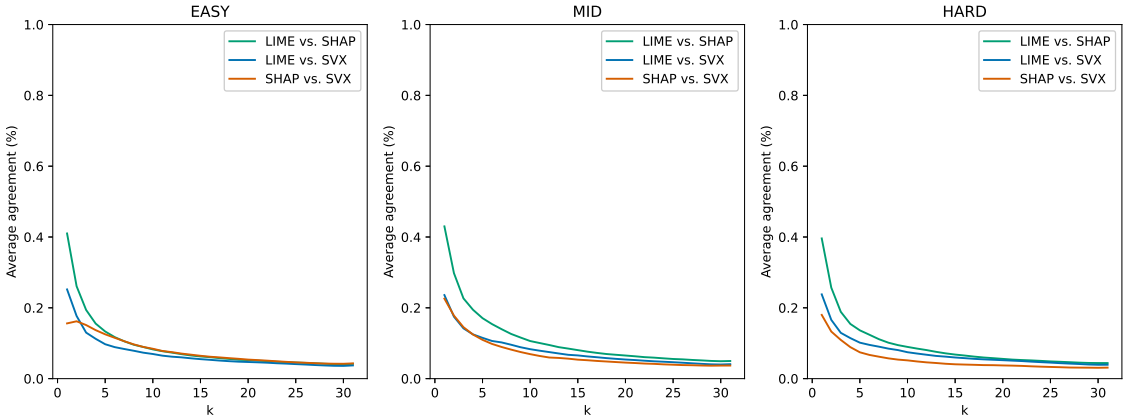

Figure 5.7: Pairwise signed rank agreement among the top-k features between different explainers. EASY, MID, and HARD indicates indicate which dataset was used.

Finally, in Figure 5.8 we see that the effect of varying the difficulty of the dataset has negligible impact.

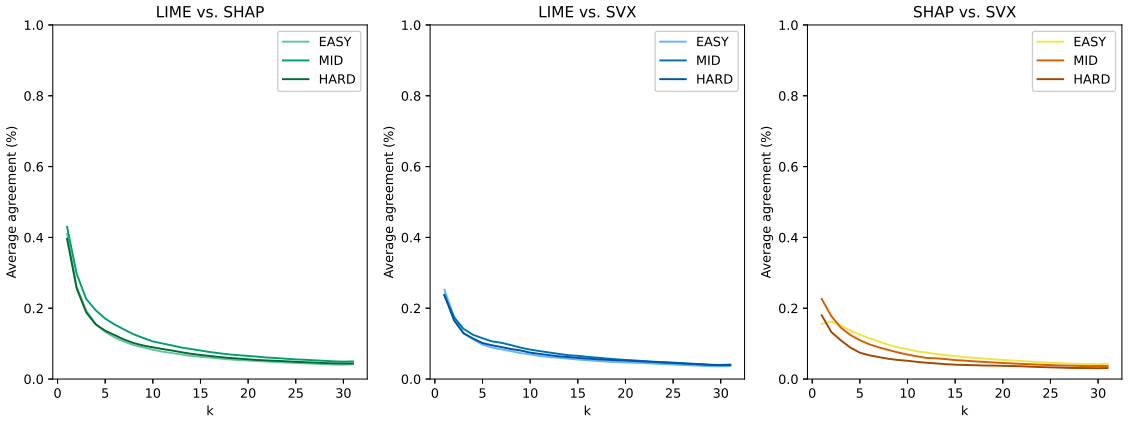

Figure 5.8: Pairwise signed rank agreement among the top-k features between different explainers. EASY, MID, and HARD indicate which dataset was used.

5.3 Correctness

Evaluating the correctness of feature and node importance measures is essential for validating the reliability and interpretability of machine learning models. Correctness, in this context, refers to how well the importance scores assigned by different explainers align with the ground truth importance. This section delves into the correctness property of feature and node importance by comparing the outputs of several explainers against ground truths. For feature importance, two approaches to constructing a ground truth are used. For node importance, we use the actual label of the nodes as ground truths. In the upcoming sections, we review the results of evaluating the correctness property.

5.3.1 Feature importance

The results from the correctness measure on feature importance are found below. The result is divided into sections 5.3.1.1 and 5.3.1.2, based on the approach used to construct the ground truth importance.

5.3.1.1 Standard deviation approach

In the upcoming section, the results presented are based on the ground truth importance, calculated using the STD approach. In Figure 5.9, the agreement between the ground truth and the explainers is shown in three different subplots. In each subplot, the feature agreement is plotted using the various difficulties of datasets. One can observe how the easy dataset exhibits better performance when $k <= 5$ for SHAP, while SVX and LIME do not show any clear pattern.

In Figure 5.10, the plots are categorized according to the datasets, where each subplot contains the agreement comparing the different explainers in each dataset. We can notice that SHAP exhibits a higher feature agreement for $k <= 10$ for the EASY dataset and $k <= 15$ for the other two datasets. It appears that the ground truth constructed using the STD approach agrees better with the feature importance given by SHAP overall. Other than that, SVX shows a slightly lower agreement than LIME for the MID and HARD dataset.

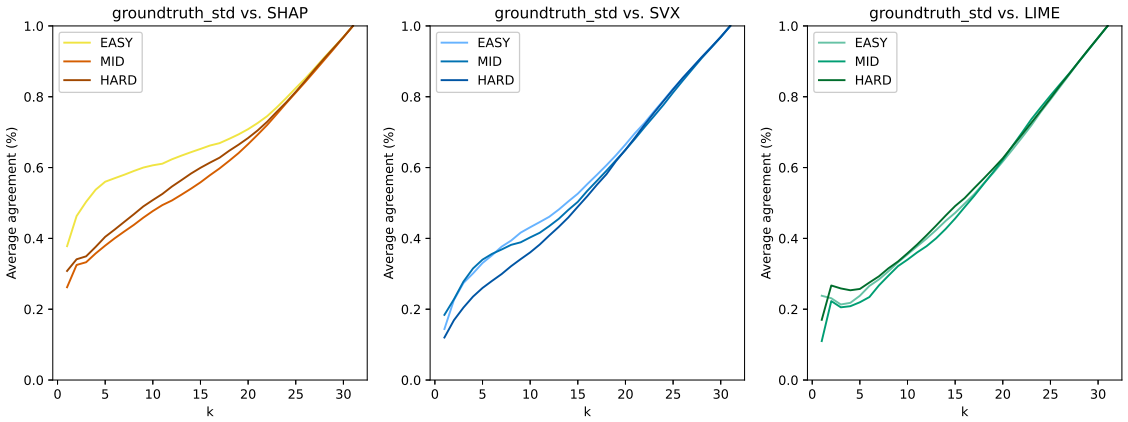

Figure 5.9: Feature agreement data comparison - STD approach.

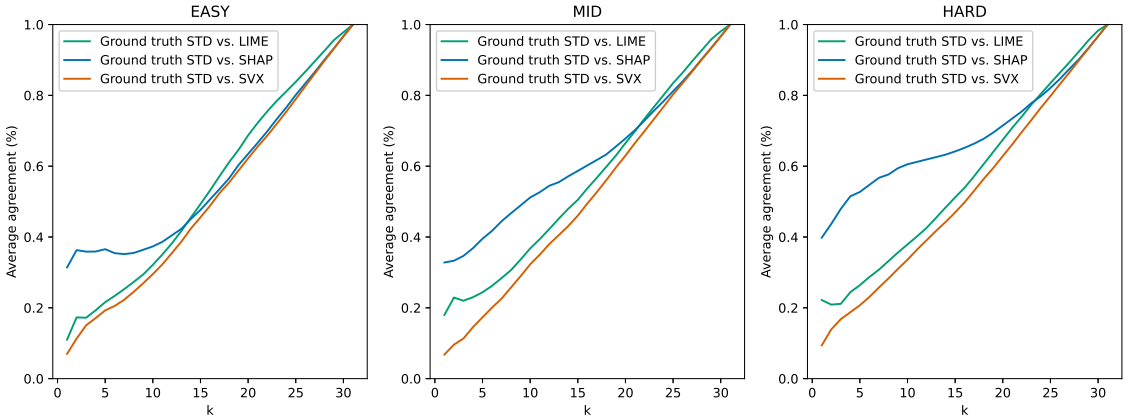

Figure 5.10: Feature agreement explanation comparison - STD approach.

5.3.1.2 Log-likelihood ratio approach

In the following section, the ground truth is no longer calculated using the STD approach, instead the LLR is used. In Figure 5.11, the feature agreements between the ground truth and the explainers are shown in three subplots, each using all datasets on that specific explainer. The feature agreement is measured for top k features, sorted on their absolute value. An observation from the plots is that the agreement does not seem to depend on the difficulty level of the dataset used in this approach. That is, the ratio of money launderers in the dataset has little to no impact on whether the feature importance given by an explainer is adhering to the constructed ground truth.

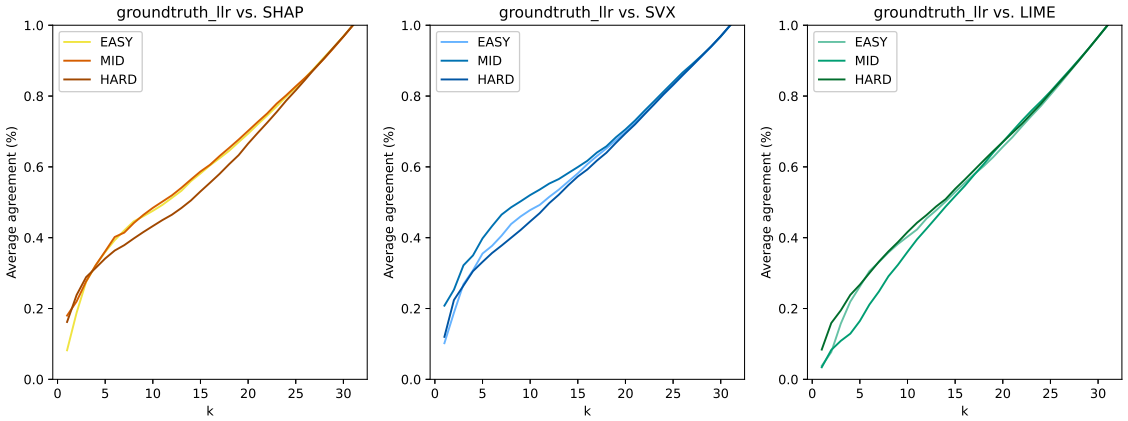

Figure 5.11: Feature agreement data comparison - LLR approach.

In Figure 5.12, the feature agreements are divided into three plots using the difficulty of datasets, allowing for a comparison between each explainer measured against the ground truth. It appears that the performance is worse for LIME than for SHAP and SVX. Overall, the agreement between the explainers and ground truth seems relatively random. Note that the feature agreement between an explainer and a randomly sampled feature importance would show a straight line through the origin and end at the value 1 when $k = 31$. The highest agreement is shown for

SVX on the EASY dataset, where the average agreement is approximately 50% at $k = 10$, which appears slightly higher than random.

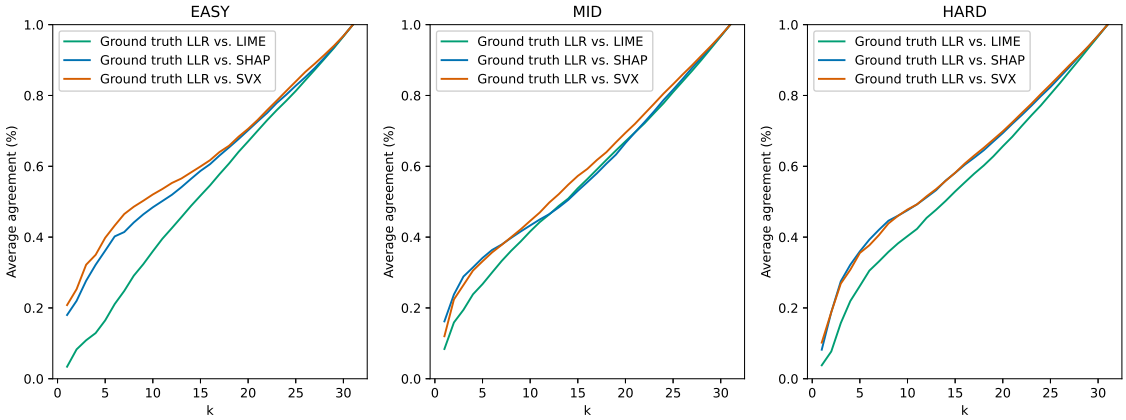

Figure 5.12: Feature agreement explanation comparison - LLR approach

5.3.1.3 Detailed analysis on feature importance

The results from the detailed analysis on feature importance for the explainers LIME, SHAP, and GraphSVX, comparing them to the ground truth using LLR, are visible in Table 5.4. In the tables, the E in $\mathcal{F}_{G\setminus E}$ is replaced by the explainer name. For example $\mathcal{F}_{G\setminus \text{LIME}}$, if the explainer E is LIME. Across all datasets, the explainers and the ground truth seem to pay high attention to the features with the characteristics $s \uparrow s$ and $s \downarrow s$, where, as a reminder, $s \uparrow s$ and $s \downarrow s$ referred to the cases when the feature was typical of suspicious behavior in the original dataset, and less typical and more typical, respectively, of suspicious behavior in the dataset without money laundering. Both characteristics are where we see the highest number of features gathered by both the explainers and the ground truth.

However, looking at the $\mathcal{F}_{E\setminus G}$ column across all datasets, there are consistently more $s \downarrow s$ than there are $s \uparrow s$, except for the MID dataset when comparing the ground truth against LIME. That is, the explainers pay attention to many instances with the characteristic where it goes from being typical of suspicious behavior to being slightly less so, while the constructed ground truth appears to miss more of those instances. The only other category the ground truth seems to pay somewhat more attention to is $s \uparrow s$. This is the largest missed category for all explainers, except for the HARD dataset using LIME, and the MID and HARD dataset using SVX, where $s \downarrow s$ also is the largest.

In table 5.4 when comparing the ground truth against LIME, it is noted that several features show no change in the original vs. the dataset without money launderers, but quite frequently occur among the top 10 by LIME. However, for SHAP and GraphSVX, this is not the case. It is not clear why this occurs for LIME, but it might be connected to the fact that the LIME surrogate model is poorly fitted, as can be seen in Section 5.5.

Table 5.4: $\mathcal{F}_{\bigcap}$ and missed features when comparing ground truth with LIME, SHAP, and SVX respectively, for EASY, MID, and HARD Datasets, using the top 10 features ordered using absolute value.

	EASY			MID			HARD		
	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{LIME}}$	$\mathcal{F}_{\text{LIME}\setminus G}$	Intercection	$\mathcal{F}_{G\setminus\text{LIME}}$	$\mathcal{F}_{\text{LIME}\setminus G}$	Intercection	$\mathcal{F}_{G\setminus\text{LIME}}$	$\mathcal{F}_{\text{LIME}\setminus G}$
$n \uparrow n$	0	0	3	0	0	201	0	0	6
$n \downarrow n$	0	0	0	0	0	224	0	0	3
$n \rightarrow s$	33	43	11	30	42	183	25	46	20
$s \rightarrow n$	126	219	195	123	193	208	232	205	326
$s \downarrow s$	658	1386	1495	797	1500	828	695	1465	1124
$s \uparrow s$	983	1552	1077	1130	1185	833	1062	1270	1071
no change	0	0	419	0	0	443	0	0	436
	EASY			MID			HARD		
	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SHAP}}$	$\mathcal{F}_{\text{SHAP}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SHAP}}$	$\mathcal{F}_{\text{SHAP}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SHAP}}$	$\mathcal{F}_{\text{SHAP}\setminus G}$
$n \uparrow n$	0	0	9	0	0	144	0	0	8
$n \downarrow n$	0	0	0	0	0	278	0	0	6
$n \rightarrow s$	38	38	12	18	54	177	6	65	12
$s \rightarrow n$	130	215	202	124	192	242	321	116	252
$s \downarrow s$	1005	1039	1482	1133	1164	1110	1257	903	1392
$s \uparrow s$	1248	1287	873	889	1426	883	795	1537	949
no change	0	0	1	0	0	2	0	0	2
	EASY			MID			HARD		
	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SVX}}$	$\mathcal{F}_{\text{SVX}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SVX}}$	$\mathcal{F}_{\text{SVX}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SVX}}$	$\mathcal{F}_{\text{SVX}\setminus G}$
$n \uparrow n$	0	0	3	0	0	150	0	0	7
$n \downarrow n$	0	0	0	0	0	214	0	0	0
$n \rightarrow s$	49	27	24	38	34	236	22	49	15
$s \rightarrow n$	164	181	276	171	145	332	294	143	387
$s \downarrow s$	976	1068	1247	889	1408	983	941	1219	1257
$s \uparrow s$	1413	1122	841	1131	1184	855	1133	1199	940
no change	0	0	7	0	0	1	0	0	4

The same analysis was made by sorting from positive to negative instead of using absolute value. The results from the detailed analysis on feature importance for the explainers LIME, SHAP, and GraphSVX, comparing them to the ground truth using LLR, are visible in Table 5.5. The results show one remarkable aspect, the most common characteristic by far among the $\mathcal{F}_{E\setminus G}$, is the $s \uparrow s$, for all datasets when compared to all explainers. Additionally, we see some of the characteristics of $n \rightarrow s$ never being picked up by the GT, although it is now and then seen among the top 10 important features given by explainers.

Table 5.5: $\mathcal{F}_{\bigcap}$ and missed features when comparing ground truth with LIME, SHAP, and SVX respectively for EASY, MID, and HARD Datasets, using the top 10 features ordered from positive to negative.

	EASY			MID			HARD		
	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{LIME}}$	$\mathcal{F}_{\text{LIME}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{LIME}}$	$\mathcal{F}_{\text{LIME}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{LIME}}$	$\mathcal{F}_{\text{LIME}\setminus G}$
$n \uparrow n$	0	3	0	22	50	148	1	2	3
$n \downarrow n$	0	0	0	6	14	198	0	0	1
$n \rightarrow s$	2	0	53	1	2	205	0	2	45
$s \rightarrow n$	247	370	124	247	341	169	273	466	164
$s \downarrow s$	1524	2808	927	1468	2798	622	1331	2854	857
$s \uparrow s$	4	22	2113	7	12	1887	7	29	2242
no change	0	20	6	0	32	20	8	27	68
	EASY			MID			HARD		
	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SHAP}}$	$\mathcal{F}_{\text{SHAP}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SHAP}}$	$\mathcal{F}_{\text{SHAP}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SHAP}}$	$\mathcal{F}_{\text{SHAP}\setminus G}$
$n \uparrow n$	0	3	1	24	48	167	0	3	2
$n \downarrow n$	0	0	0	14	6	309	0	0	0
$n \rightarrow s$	0	2	41	0	3	220	0	2	10
$s \rightarrow n$	239	378	128	241	347	133	373	366	114
$s \downarrow s$	1817	2515	958	1751	2515	623	1964	2221	761
$s \uparrow s$	9	17	1797	4	15	1510	13	23	1759
no change	0	20	10	1	31	3	0	35	4
	EASY			MID			HARD		
	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SVX}}$	$\mathcal{F}_{\text{SVX}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SVX}}$	$\mathcal{F}_{\text{SVX}\setminus G}$	$\mathcal{F}_{\bigcap}$	$\mathcal{F}_{G\setminus\text{SVX}}$	$\mathcal{F}_{\text{SVX}\setminus G}$
$n \uparrow n$	2	1	2	18	54	157	0	3	5
$n \downarrow n$	0	0	0	9	11	199	0	0	1
$n \rightarrow s$	1	1	57	1	2	224	0	2	60
$s \rightarrow n$	258	359	144	260	328	166	290	449	155
$s \downarrow s$	1413	2919	854	1293	2973	591	1219	2966	807
$s \uparrow s$	8	18	2162	3	16	1921	6	30	2321
no change	6	14	93	7	25	151	7	28	129

5.3.2 Node importance

In Figure 5.13 the result for comparing node importance with the corresponding ground truth labels is shown. The solid line shows the accuracy that a positive Shapley value, above a given threshold, predicts that the ground truth label is money launderer. The dashed line shows the percentage of nodes that have at least one node with a Shapley value above the corresponding threshold. As a general trend for all datasets, the accuracy increases as a function of the Shapley value, meaning that the larger the Shapley value seen on a node, the more probable that the node is also part of the money laundering scheme. Furthermore, the percentage of nodes having a Shapley value above a given threshold is quite similar between all datasets, although it is slightly lower for the EASY dataset.

The accuracy for the EASY dataset is higher than 80% across almost the whole interval. However, around $\tau = 0.9$, we see a small dip in the accuracy, but then it increases to 1 at the end of the interval. The final decrease to 0 is probably because

there is no Shapley value above this threshold, in which case the accuracy would be 0 by default.

For the MID dataset, the accuracy is lower than for the EASY one. However, it is still fairly high, starting from 60% at $\tau = 0$ and increasing quite steadily to 85% at $\tau = 0.9$. Then it increases rapidly to 1 in the last part of the interval.

Finally, for the HARD dataset, the accuracy is lower than MID. It starts at 40% at $\tau = 0$ then rises steadily to about 50% at $\tau = 0.6$. After this point, the accuracy increases exponentially along the rest of the interval up to 1.

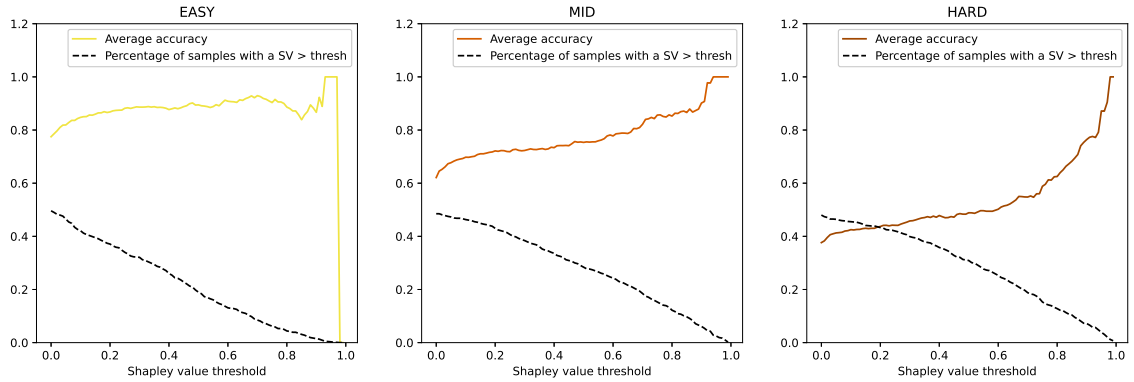

Figure 5.13: Average accuracy as a function of node importance.

5.4 Completeness & Compactness

In Figure 5.14 the plots obtained from the incremental deletion check are shown, with the removal of features from the most negative to the most positive importance in the top row, and the removal of features from the most positive to the most negative importance in the bottom row. In each plot, the result is shown for the three different datasets, and the dashed black line marks the average number of features removed where the threshold τ for the feature importance is equal to 0. It is important to note that for LIME and SHAP only features are excluded, whereas for GraphSVX both features and nodes are excluded, which is why the plotted intervals differ.

Looking at LIME and SHAP in the top row we see that most change in $\overline{\Delta\hat{y}}$ occurs at the edges of the interval involving about 15 features in the intervals 0-6 and 22-31. Between these intervals the curve is fairly flat, indicating that the features removed here are less influential on the model prediction. For GraphSVX in the top row, we see a similar behavior although the middle part is less flat. This suggests that features with small negative importance still impact the model prediction.

Changing our attention to the plots for LIME and SHAP in the second row, we see that most change again occurs at the edges of the interval in roughly the same intervals as previously. However, the EASY curve for SHAP has taken a slightly less flat form. Despite this deviation, the result indicates that about 15 features corresponding to the features with the smallest importance could (in absolute value), on average, be excluded from the explanation without leaving out a significant amount of information. Moreover, for GraphSVX in the bottom row, we see curves with a more flat middle section compared to the corresponding plot in the top row. This

suggests that GraphSVX may have about 20 features or nodes that could be excluded from the explanation in the interval 10-30, although this result seems less certain compared to LIME and SHAP, for which the plots were more consistent in the top and bottom rows.

For GraphSVX, the zero marker is not as centered as for LIME and SHAP. Its position indicates that the features or nodes that can be removed mostly have small negative importance. For LIME and SHAP however, it is roughly equally many features with positive versus negative importance that can be removed.

Finally, the difficulty level of the dataset shows a consistent impact on the result across all plots. The easier the dataset, the more sensitive the model seems to be to remove features in this way as $\overline{\Delta\hat{y}}$ varies more. Also, the flat section generally seems slightly flatter for the models trained on harder datasets.

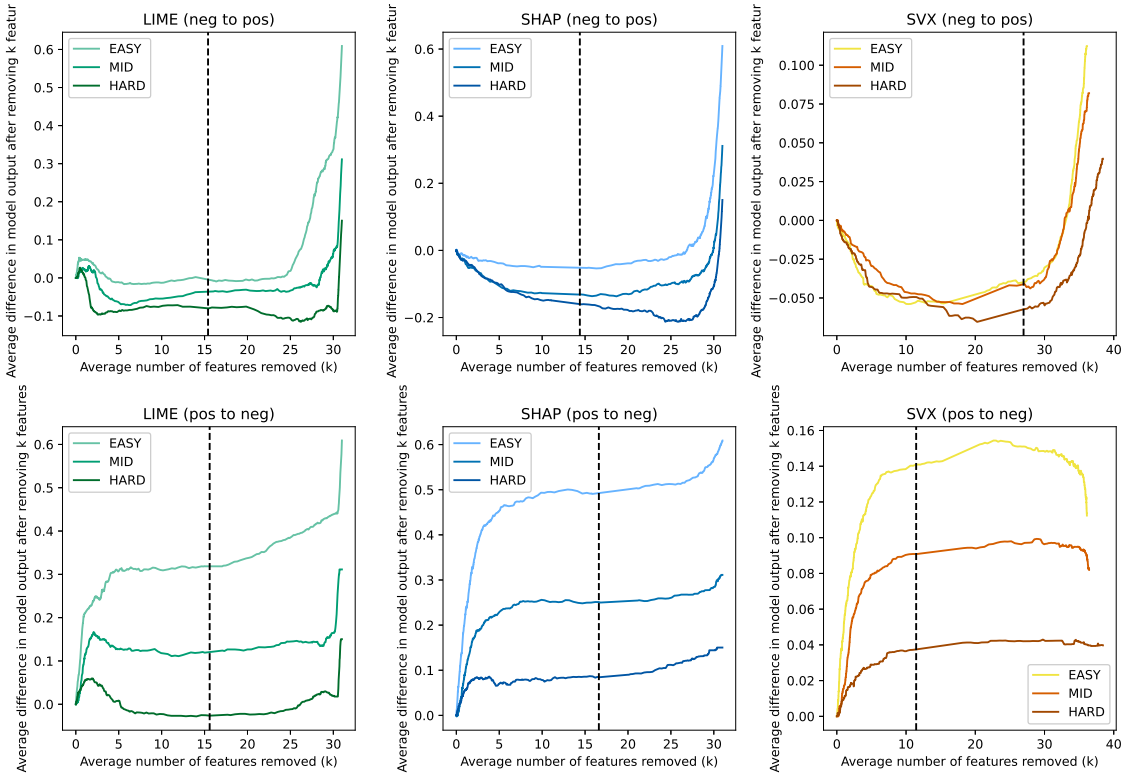

Figure 5.14: Average difference in prediction during an incremental deletion check going from negative to positive feature importances in the top row, and from positive to negative feature importances in the bottom row.

5.5 Confidence

In Figure 5.15 we compare the R2 values from fitting the surrogate models in LIME and GraphSVX. For EASY we see that most R2 values are higher for GraphSVX than for LIME. Furthermore, we see that the spread of the R2 values is greater for GraphSVX and that some outliers are stretching toward the negative numbers indicating a terrible fit. MID and HARD have a similar result, although GraphSVX is not getting any negative values. The results show that the fit of the surrogate

model in GraphSVX is significantly better than in LIME, in most cases.

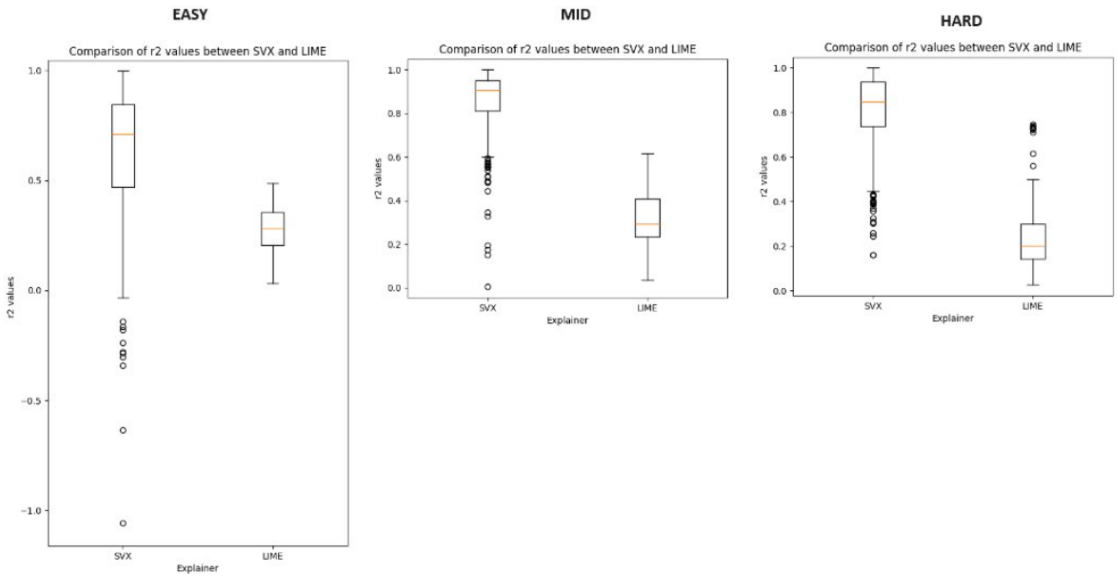

Figure 5.15: R2 values from fitting a surrogate model in the explanation methods.

6
Discussion

In this chapter we discuss our results, highlighting the shortcomings and suggesting directions for future research on the subject.

All three XAI techniques LIME, SHAP and GraphSVX make explicit assumptions about the features being independent. This is a potential source of error, as some of the features used, for example, sum and median, are highly correlated. Using less correlated features is therefore something that would be interesting to pursue in further studies. This gives rise to a trade-off, however: depending on what features are used, the information we get from an explanation will be different. Having a large feature importance on the feature sum indicates that a lot of money has been flowing through this account, without indicating the direction of this flow. Having large feature importance on the feature in_sum indicates, on the other hand, that it entails that ongoing transactions are of interest. The feature engineering will thus balance model performance, investigator usefulness when highly important, and low correlation.

6.1 Coherence

To build trust in one or more explainers it is desirable that they agree on what features are important. If two explainers based on slightly different theories frequently arrive at the same conclusion, it would indicate that they both measure what they set out to measure. Our results show that the agreement among the explainers considered in this study is surprisingly low compared to what we hoped for. One reason may be due to our adaption of SHAP and LIME to graph-structured data. In this extension, feature importance is viewed slightly differently from GraphSVX, potentially giving rise to disagreements. However, it is worth noting that low coherence does not imply that all explainers are performing poorly. If, hypothetically, two explainers were performing poorly then all coherence plots would be affected by this. Therefore it is important to consider coherence in combination with other metrics, such as correctness. Having about 40% to 50% sign agreement for the top 10 features indicates that the explainers are not just providing random explanations. In light of this, agreeing on 5 of 10 features conditioned on sign indicates that these 5 features are important and contribute towards predicting a specific class, even if there is disagreement on the remaining features. Also, the magnitude of the feature importance on the disagreeing features should be taken into consideration. If the disagreeing features are close to zero, the disagreement could arise from randomness.

When computing the pairwise agreement between the top-k features we only

considered the ordering of these features, and did not analyze which the features were. Overall, we did not make any systematic analysis of which features were most frequently considered important by for example computing each feature's average importance. Such an analysis could serve as a sanity check if we know a priori which features should be most important according to the model, but this aspect was instead addressed when measuring correctness. Analyzing the average feature importance could also serve as an attempt at creating global feature importance, which might be interesting for getting a rough overview of how sensitive the model is to different features. For each individual explanation, however, the feature importance of the top-k features does not need to agree with their respective average importance, and each case must consequently be evaluated separately nonetheless.

Furthermore, an alternative way to measure coherence could be to compare the given explanation against the reasoning of an inherently transparent model, such as a tree-based model. However, it is important to note that this is not a foolproof check, since that would mean comparing different models, and thus their reasoning will not be 100% aligned. Moreover, it is expected that a GNN should be able to capture patterns that a tree-based model can not, since we expect the GNN to outperform traditional rule-based models. Possibly, as a fictive example, we could generate data that only contains patterns that both a GNN and a tree-based model can capture. Then, one could train a tree-based and a GNN separately on that data, so that they approximately make the same predictions, and consequently have similar decision boundaries. Once that is ensured, the coherence between a tree-based model explanation, and the feature importance given by the studied XAI techniques could have been compared.

6.2 Correctness

6.2.1 Feature importance

To ensure the explainers are faithful to the black box model reasoning, the measure of correctness was introduced in Section 4.2.2. Evaluating the agreement between ground truth and the explainer itself would be an ideal way to assess the correctness of an explainer. Having access to a synthetic dataset, appeared to simplify the process of doing so, although the question of how to properly construct ground truth importance remained. In the results gathered, the two approaches of constructing a ground truth were compared against the individual explainers. A relatively low importance was displayed, which can imply two things. Either, the ground truth created is based on inaccurate assumptions, or the explainer is not capturing the true black-box reasoning.

The detailed analysis of the different characteristics showed how the explainers and the ground truth often captured a large part of the same features, with the characteristics $s \uparrow s$ and $s \downarrow s$. That is, in the original dataset, the feature appears typical of suspicious behavior for both characteristics. In the dataset without money laundering, the feature remains typical of suspicious behavior but slightly more ($\uparrow$) or less ($\downarrow$). Nonetheless, those were also the characteristics frequently "missed" by the explainer and the ground truth when compared against each other. This could

indicate a potential issue with the ground truth constructed, which we'll elaborate on. The ground truth measures the importance by looking at the difference in LLR for a feature in the original dataset versus the dataset without money laundering. This means it does not consider the magnitude of the individual LLR for a feature in the original dataset, or the dataset without money laundering respectively. As an example, a feature may have a high log-likelihood of belonging to the money laundering behavior, both in the original dataset and the dataset without money laundering. But, as long as the differences between those are small, the importance will be relatively small when constructing the ground truth using the suggested approach. This may be an inaccurate assumption, as the feature may still have strong importance for predicting the class money launderer. Since the magnitude of the importance made using the STD approach is based using the difference divided by the standard deviation, this would probably be shown if a deeper investigation was made into the STD approach as well. This problem is something that indicates that the ground truth used in this report may not be ideal.

Continuing, when performing the analysis of counting the characteristics by sorting the top 10 features from positive to negative, another insight can be made. Since the $s \uparrow s$ consistently shows a high count of $\mathcal{F}_{E \setminus G}$, this may be because of another faultiness in the construction of the ground truth. The features with the characteristic $s \uparrow s$ will have a negative importance in the constructed ground truth. However, all explainers appear to still point to those types of features as positively predicting money laundering. Additionally, we see some instances of $n \rightarrow s$ also being considered as most important by the explainers, but never for the ground truth. This may be caused by the fact that we are looking at each feature and its likelihood of belonging to the normal distribution, versus the money laundering distribution, in isolation. That is, a node with a feature that seems to belong to the normal distribution when reviewed in isolation, may have a composition of other features, that make the value of the feature studied seem more suspicious than it would otherwise. This is something that a proper ground truth explanation also may benefit from incorporating.

Note that there were other available methods to measure correctness, such as constructing a synthetic dataset and fixating the parameters so that only a handful of features differ, and then reviewing if this was captured by the explainer. Although this would've been feasible, the approach we chose would test the explainer in a real-life situation, which appeared more interesting.

To summarize the correctness measured on the feature importance, we cannot conclude whether the explainers studied are correctly adhering to the black box reasoning or not. However, our investigation of the results may serve as a starting point to create a more appropriate ground truth importance.

6.2.2 Node importance

Node importance seems like a promising indicator for determining which nodes are in the vicinity of an alerted account that is also part of the money laundering scheme. In our understanding, investigators today are not provided with this kind of information, but rather just the feature importance. This makes node importance

a potential source of new information that can guide an investigator to the accounts where suspicious transactions most probably have been made.

Moreover, it may be possible to use the node importance in combination with a GNN as a detection model to more accurately detect money launderers. For example, if the GNN has predicted a node u to not be a money launderer, while the node importance when explaining a neighboring node v indicates that u is a money launderer, it may be worth putting u under scrutiny nonetheless. In this thesis, we have not explored the negative node importance, but if the results show something similar, i.e., that negative node importance indicates that a node is not a money launderer, it could be used to reduce the number of false positives.

6.3 Completeness & Compactness

The results show that about 15 features can be removed from the explanation for LIME and SHAP, and about 20 features or nodes for GraphSVX, without affecting the output of the model significantly. Consequently, the explanations show high completeness using the top 15 features for LIME and SHAP, and the top 20 features or nodes for GraphSVX have the highest impact on the model. It is not clear, though, to what degree this result stems from feature engineering. Having too many correlated features could potentially make some of them less important because the information they carry remains in another feature, for example, if we are using both sum and median. Being able to remove many features in this case may thus be a consequence of having redundant features.

6.4 Confidence

The confidence results show that the surrogate model of GraphSVX is a better fit than that of LIME. This suggests that the explanation from GraphSVX is more trustworthy. To some extent, this may be due to the choice of the kernel width ν in LIME. Sampling from a too large neighborhood could cause a poor fit, as it tries to minimize the squared distance to all points simultaneously.

Another argument as to why we might want to trust the result of LIME less compared to GraphSVX is that it neglects some of the information that the GNN utilizes in its computations. The GNN takes as input the 1-hop subgraph of the node v that is being explained. LIME, however, only optimizes its surrogate model over $\mathbf{x}_v$ and not over $\mathcal{N}(v)$ (and consequently not over its neighbor's node feature vectors). In the adoptions of SHAP and LIME to GNNs, however, the sampled points affect both v:s feature vector and the feature vectors of the nodes in $\mathcal{N}(v)$. However, for LIME, the features will be masked depending only on whether the sampled point is in the same quantile as x_j, disregarding the quantiles of the neighboring nodes. Because all feature values in the new sample will be different from x_j, all nodes in $\mathcal{N}(v)$ will have the same value, thus making the GNN completely ignore the graph structure. In hindsight, we could have adopted LIME and SHAP in two different ways. For SHAP we could have kept the current adoption, but for LIME we should not have overwritten all neighboring features. In contrast, GraphSVX optimizes

over both $\mathbf{x}_v$ and $\mathcal{N}(v)$ and has more interpretable components, and the fit might consequently become better because of this.

6.5 Explainers

In our study, we employed three explainability techniques—LIME, SHAP, and GraphSVX—to interpret the results of our model. Each method comes with its own set of strengths and weaknesses, which we have observed during our research. Additionally, we made specific extensions to adapt these methods to our graph-structured data, which further introduces unique advantages and challenges.

LIME creates a surrogate linear model that approximates the behavior of the black-box model locally, and it can be applied to any machine learning model, making it a versatile choice. However, LIME's results can be sensitive to the choice of neighborhood for the surrogate model, which can lead to variability in the explanations as suggested by our results. Moreover, when adapted for graph data, LIME might neglect the structural information of the graph, as it optimizes only over the node's features and not its neighbors. This can lead to less reliable explanations. Implementing LIME for graph data proved to be more complex than anticipated, requiring significant adjustments to accommodate the graph structure.

SHAP provides a solid theoretical grounding based on cooperative game theory, ensuring fair attribution of feature importance. SHAP values offer consistency, guaranteeing that the feature importances sum up to the model prediction. However, calculating SHAP values can be computationally expensive, especially for large datasets. Similar to LIME, adapting SHAP to graph data might introduce inconsistencies due to the oversimplification of neighborhood interactions.

GraphSVX, designed specifically to handle graph-structured data, takes into account both node features and graph topology, leading to potentially more accurate explanations. Our results showed that the surrogate model of GraphSVX fits better compared to LIME, suggesting higher trustworthiness. However, GraphSVX's complexity can make it harder to implement and understand compared to more generic methods like LIME and SHAP. In addition, the method may struggle with scalability issues for very large graphs due to the computational demands of optimizing the graph structure.

The adaptations we made to LIME and SHAP for graph data involved modifying how these methods sample and interpret the neighborhood of nodes. While these extensions aimed to better capture the dependencies within the graph structure, they introduced specific challenges. By incorporating node neighborhoods, these extensions aim to provide more relevant and context-aware explanations and could uncover insights that are missed by purely feature-based approaches. However, modifying LIME and SHAP to handle graph data required significant changes, complicating the implementation. Considering the limitations observed, exploring other graph-tailored techniques might provide a better comparison for validating the results obtained by GraphSVX.

7
Conclusion

In this thesis, we have explored the application of three XAI techniques to GNNs in the context of AML. The techniques were applied to GNNs trained to detect money launderers in synthetic transaction data represented as a graph. The explanation methods used were LIME, SHAP, and GraphSVX. While LIME and SHAP have been thoroughly studied in the literature, their adaption to produce explanations for GNNs was more challenging than anticipated. GraphSVX, on the other hand, is an XAI technique specifically tailored to explain GNNs. To evaluate the performance of the XAI techniques we used 5 metrics introduced in the Co12 framework, coherence, correctness, completeness, compactness, and confidence. In conclusion, our analysis highlights several important directions for future research and improvements in applying explainability methods to GNNs in AML. The low coherence among explainers, potentially due to the adaptations made for LIME and SHAP, indicates a need for further investigation to determine their reliability and trustworthiness, although partial agreement on important features suggests some validity. The measure of correctness for feature importance remains inconclusive, highlighting the necessity of constructing a more appropriate ground truth. Since there is currently no systematic approach to construct a ground truth feature importance for this type of data, this gap is an opportunity for future research. As systematically validating the correctness is important to build trust in an explainer, and ultimately deploying it in practice, future research on this topic is warranted. Node importance given by GraphSVX emerges as a promising indicator for identifying suspicious nodes in money laundering schemes, potentially providing better insights and useful information for investigators. Moreover, the results suggest that fewer features can maintain model performance, indicating high completeness. Lastly, the better fit of the GraphSVX surrogate model over LIME suggests that incorporating comprehensive neighborhood information may be crucial. These findings emphasize the complexity and importance of refining and enhancing XAI evaluation methods to properly assess the accuracy and reliability of the explanations given.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY

Bibliography

[1] D. Gunning and D. Aha, "Darpa's explainable artificial intelligence (xai) program," *AI magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[2] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van Keulen, and C. Seifert, "From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–42, 2023.

[3] United Nations Office on Drugs and Crime, "Money laundering," https://www.unodc.org/unodc/en/money-laundering/overview.html, accessed: 2024-05-31.

[4] D. Cox, *Handbook of Anti-Money Laundering.* John Wiley amp; Sons, Incorporated, 2014.

[5] R. F. D. TRAFFICKING, "Estimating illicit financial flows resulting from drug trafficking and other transnational organized crimes," 2011.

[6] *Nationell riskbedömning av penningtvätt och finansiering av terrorism i Sverige 2019*, ser. A303.013/2019. Polismyndigheten, 2019.

[7] European Comission, "Money laundering," https://home-affairs.ec.europa.eu/policies/internal-security/organised-crime-and-human-trafficking/money-laundering_en, accessed: 2024-05-31.

[8] LexisNexis, "True Cost of Financial Crime Global Summary," https://risk.lexisnexis.com/global/-/media/files/financial%20services/infographics/lnrs-global-tcoc-infographic2022_v2-nxr15749-00-1122-en-us.pdf, 2022, [Accessed: 2024-06-17].

[9] Piotr Bąkowski, "Fourth anti-money laundering directive," https://www.europarl.europa.eu/legislative-train/theme-civil-liberties-justice-and-home-affairs-libe/file-fourth-anti-money-laundering-directive, 2019, accessed: 2024-05-31.

[10] FATF, "Guidance for a risk-based approach. the banking sector," https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Risk-based-approach-bankingsector.html#:~:text=A%20risk%2Dbased%20approach%20means,with%20the%20level%20of%20risk, 2014, accessed: 2024-05-31.

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

[12] B. for International Settlements, “Project aurora,” Bank for International Settlements, BIS Report, 2023. [Online]. Available: https://www.bis.org/publ/othp66.pdf

[13] European Union. (Year of publication) Key issue: [title of the key issue]. [Online]. Available: https://www.euaiact.com/key-issue/5

[14] A. Adadi and M. Berrada, “Peeking inside the black-box: a survey on explainable artificial intelligence (xai),” *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[15] Z. Li, Y. Zhu, and M. Van Leeuwen, “A survey on explainable anomaly detection,” *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 1, pp. 1–54, 2023.

[16] T. Suzumura and H. Kanezashi, “Anti-Money Laundering Datasets: InPlus-Lab anti-money laundering datadatasets,” http://github.com/IBM/AMLSim/, 2021.

[17] Swish.nu, “Swish.nu,” https://www.swish.nu/, [Accessed: 2024-06-17].

[18] Finanspolisen, “Banker och finansiella institut som brottsverktyg.”

[19] M. Weber, J. Chen, T. Suzumura, A. Pareja, T. Ma, H. Kanezashi, T. Kaler, C. E. Leiserson, and T. B. Schardl, “Scalable graph learning for anti-money laundering: A first look,” *arXiv preprint arXiv:1812.00076*, pp. 1–7, 2018.

[20] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai,” *Information fusion*, vol. 58, pp. 82–115, 2020.

[21] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, “Explaining explanations: An overview of interpretability of machine learning,” in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[22] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, “Explainable ai: A brief survey on history, research areas, approaches and challenges,” in *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8*. Springer, 2019, pp. 563–574.

[23] European Data Protection Supervisor. (2023) TECH DISPATCH. [Online]. Available: https://www.edps.europa.eu/data-protection/our-work/publications/techdispatch/2023-11-16-techdispatch-22023-explainable-artificial-intelligence_en

[24] M. Busuioc, “Accountable artificial intelligence: Holding algorithms to account,” *Public administration review*, vol. 81, no. 5, pp. 825–836, 2021.

[25] European Commission. (Year of publication) Regulatory framework for ai. [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

[26] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan *et al.*, “Explainable ai (xai): Core ideas, techniques, and solutions,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–33, 2023.

[27] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, “Explainable ai: A review of machine learning interpretability methods,” *Entropy*, vol. 23, no. 1, p. 18, 2020.

[28] S. Agarwal, S. Jabbari, C. Agarwal, S. Upadhyay, S. Wu, and H. Lakkaraju, “Towards the unification and robustness of perturbation and gradient based explanations,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 110–119.

[29] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, “Gnnexplainer: Generating explanations for graph neural networks,” *Advances in neural information processing systems*, vol. 32, 2019.

[30] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, “Cf-gnnexplainer: Counterfactual explanations for graph neural networks,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 4499–4511.

[31] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, “Graphlime: Local interpretable model explanations for graph neural networks,” *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[32] A. Duval and F. D. Malliaros, “Graphsvx: Shapley value explanations for graph neural networks,” in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer, 2021, pp. 302–318.

[33] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[34] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, “Graph attention networks,” *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[35] M. T. Ribeiro, S. Singh, and C. Guestrin, “" why should i trust you?" explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[36] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[37] T. Han, S. Srinivas, and H. Lakkaraju, “Which explanation should i choose? a function approximation perspective to characterizing post hoc explanations,” *Advances in neural information processing systems*, vol. 35, pp. 5256–5268, 2022.

[38] D. Garreau and U. Luxburg, “Explaining the explainer: A first theoretical analysis of lime,” in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 1287–1296.

[39] L. S. Shapley *et al.*, “A value for n-person games,” 1953.

[40] E. Strumbelj and I. Kononenko, “An efficient explanation of individual classifications using game theory,” *The Journal of Machine Learning Research*, vol. 11, pp. 1–18, 2010.

[41] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in neural information processing systems*, vol. 30, 2017.

[42] AI Sweden, “Federated learning in banking,” https://www.ai.se/en/project/federated-learning-banking, 2023, accessed: 2024-05-07.

[43] J. Gerlings and I. Constantiou, “Machine learning in transaction monitoring: The prospect of xai,” *arXiv preprint arXiv:2210.07648*, 2022.
[44] T. K. Rusch, M. M. Bronstein, and S. Mishra, “A survey on oversmoothing in graph neural networks,” *arXiv preprint arXiv:2303.10993*, 2023.
[45] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
[46] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, “A survey of methods for explaining black box models,” *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

A

Hyperparameters

Table A.1: Row Names and Parameter Ranges

	Parameter Ranges
Hidden channels	10,11,12,13,14,15
Dropout	(0.1, 0.3)
Learning rate	(0.001, 0.005)
β	(0.9999, 0.99992)

B

LIME Sampling

Example:
Let $q_{j,1} = -1, q_{j,2} = 0, q_{j,3} = 1$ for all j, and $\xi = (0.9, -1.1, -1.2, -0.1)$. We then have that $\hat{\phi}(\xi) = (3, 1, 1, 2)$. Further, let $u_1 = (2, 1, 3, 2)$ be a sampled point. First, we construct the continuous vector by

$$\begin{aligned}
(x_1)_1 &\sim \mathcal{N}_T(\mu, \sigma, q_1, q_2) \quad \text{(for example } (x_1)_1 = -0.4\text{)}, \\
(x_1)_2 &\sim \mathcal{N}_T(\mu, \sigma, q_0, q_1) \quad \text{(for example } (x_1)_2 = -1.7\text{)}, \\
(x_1)_3 &\sim \mathcal{N}_T(\mu, \sigma, q_2, q_3) \quad \text{(for example } (x_1)_3 = 0.35\text{)}, \\
(x_1)_4 &\sim \mathcal{N}_T(\mu, \sigma, q_1, q_2) \quad \text{(for example } (x_1)_4 = -0.2\text{)},
\end{aligned}$$

giving us for example $x_1 = (-0.4, -1.7, 0.35, -0.2)$. Second, we construct the binary vector by

$$\begin{aligned}
(z_1)_1 &= \mathbb{1}_{\hat{\phi}_1(\xi)}((u_1)_1) = \mathbb{1}_3(2) = 0, \\
(z_1)_2 &= \mathbb{1}_{\hat{\phi}_2(\xi)}((u_1)_2) = \mathbb{1}_1(1) = 1, \\
(z_1)_3 &= \mathbb{1}_{\hat{\phi}_3(\xi)}((u_1)_3) = \mathbb{1}_1(3) = 0, \\
(z_1)_4 &= \mathbb{1}_{\hat{\phi}_4(\xi)}((u_1)_4) = \mathbb{1}_2(2) = 1,
\end{aligned}$$

giving us $z_1 = (0, 1, 0, 1)$.

C

Alert Patterns

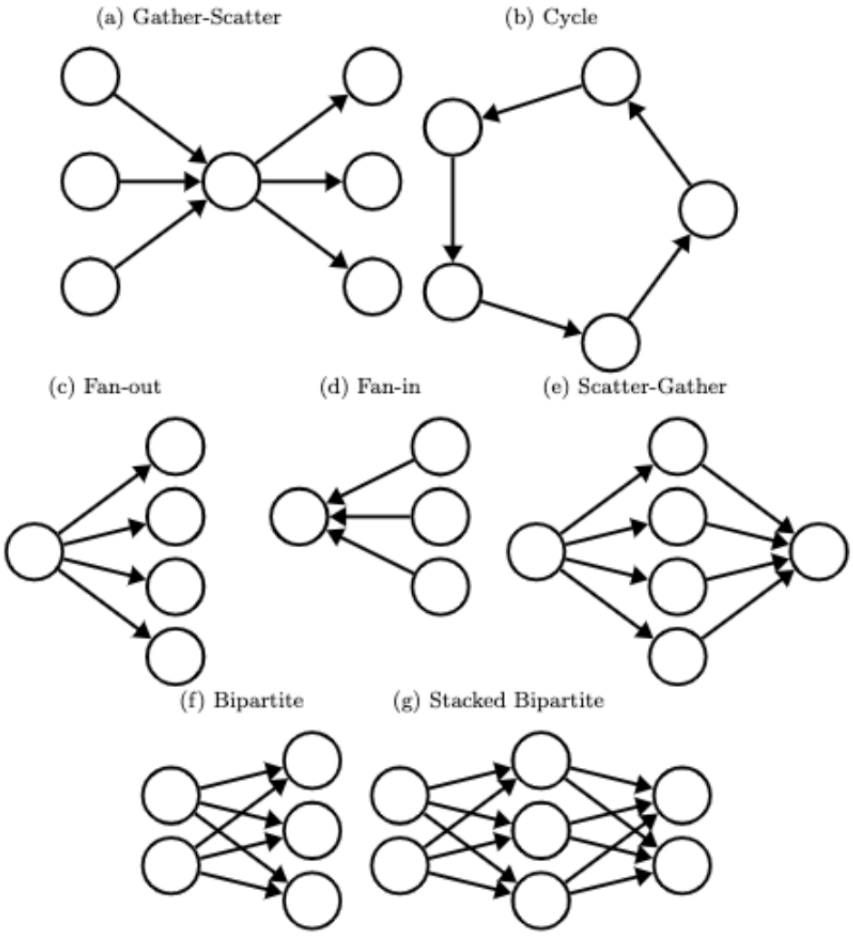

Figure C.1: Collection of Alert Patterns