DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
48

PAVLO TERTYCHNYI

Machine Learning Methods for Anti-Money Laundering Monitoring

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

48

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

48

PAVLO TERTYCHNYI

Machine Learning Methods for Anti-Money Laundering Monitoring

UNIVERSITY OF TARTU
Press

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on October 25, 2023 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Prof.	Marlon Dumas University of Tartu Estonia

Opponents

Prof.	Fethi Rabhi The University of New South Wales Australia
Prof.	Branka Hadji Misheva Bern University of Applied Sciences Switzerland

The public defense will take place on November 20, 2023 at 14:15 in Narva mnt 18-2049.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906 (print) ISSN 2806-2345 (PDF)
ISBN 978-9916-27-389-0 (print) ISBN 978-9916-27-390-6 (PDF)

University of Tartu Press
`http://www.tyk.ee/`

To my family and friends

ABSTRACT

Money laundering (MoL) poses a significant threat to global financial systems, enabling criminals to disguise the illicit origins of funds and integrate them into the legitimate economy. It not only has financial consequences but also undermines the stability of financial systems, threatens national security, and erodes public trust in financial institutions. Governments and law enforcement agencies worldwide are concerned about identifying and disrupting these illicit activities, as MoL accounts for a significant portion of the global Gross Domestic Product (GDP), enabling criminal proceeds, terrorist financing, and tax evasion. In turn, financial institutions deploy monitoring mechanisms to detect and report potential MoL activities. The traditional approach for such monitoring mechanisms is in deploying rule-based systems which are commonly used due to their simplicity and interpretability, but suffer from shortcomings, particularly when it comes to detecting complex and emerging MoL schemes. Complementing these systems with machine learning algorithms can help to capture complex relationships and improve the effectiveness of detection and prevention efforts.

The goal of this thesis is to create a set of frameworks that, combined, provide a comprehensive solution for automated MoL detection using machine learning algorithms. The development of such a solution using machine learning techniques is complicated by multiple challenges that originate from the nature of the MoL phenomenon and the data available to detect this phenomenon. Modern society's shift towards a cashless state and globalization has led to a significant increase in digital payments, resulting in a massive amount of data to analyze for MoL detection. The diversity of financial products further complicates MoL detection, as multiple channels can be used for illicit activities. Additionally, MoL is a rare and deliberately concealed phenomenon, making it challenging to train machine learning models due to data imbalance and the need for sophisticated detection techniques. Furthermore, the constantly evolving nature of MoL schemes requires regular updates and retraining of machine learning models to keep up with new patterns and features.

This thesis makes four main contributions to the research area. The first contribution is a framework for detecting individual MoL patterns. The framework focuses on scalability and imbalance resistance, utilizing a two-layered architecture and various approaches for feature extraction. The second contribution is a monitoring system that is designed and developed to raise alerts when potentially illicit MoL behavior is detected. The system generates accurate, non-redundant, and timely alerts, and employs a family of metrics that capture the time aspect in contrast to standard machine learning metrics. The third contribution is an interpretability module that provides textual explanations for the raised alerts. This enhances the understanding of why alerts are generated and guides the investigation process. The fourth contribution is a framework for detecting group MoL behavior. The framework constructs a social network based on financial connec-

tions derived from transaction histories and identifies illicit group behavior without assuming a prior knowledge of such patterns or having complete transaction data.

Collectively, the presented contributions form a comprehensive solution for automated MoL detection designed to meet the following business requirements: (i) accuracy - the solution must provide accurate outputs; ii) robustness - the solution must tackle a wide range of MoL pattern classes; (iii) timeliness - the solution must provide its output in a proper time; (iv) actionability - the outputs of the solution should be feasible to analyse by humans; (v) interpretability - the solution must provide interpretable outputs; and (vi) scalability - the solution should be scalable relative to the increase of the financial data that needs to be handled. All four contributions have been tested on a real-life large-scale dataset containing customer profiles, transaction histories, and labels provided by AML experts from three different jurisdictions. The results were evaluated through computational experiments on historical data and interactive feedback from domain experts at a financial institution.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Anti-Money Laundering	AML
Extreme Gradient Boosting	XGBoost
Categorical Boosting	CatBoost
Confusion Matrix	CM
Decision Trees	DT
Deep Learning	DL
Edited Nearest Neighbors	ENN
Feature Group	FG
Financial Intelligence Unit	FIU
Gradient Boosting Trees	GBT
Graphics Processing Unit	GPU
Isolation Forest	IF
Label Propagation Algorithm	LPA
Logistic Regression	LR
Machine Learning	ML
Money Laundering	MoL
Monitoring System	MS
Neural Networks	NN
Positive and Unlabelled	PU
Precision Recall Area Under Curve	PRAUC
Random Forest	RF
Receiver Operating Characteristic Area Under the Curve	ROCAUC
Suspicious Activity Report	SAR
SHapley Additive exPlanations	SHAP
Support Vector Machines	SVM
Synthetic Minority Over-sampling Technique	SMOTE

1. INTRODUCTION

1.1. Research Problem

Financial crime has become a widespread and complex issue in today's global economy, affecting both individuals and institutions. With the growth of technology and globalization, financial crime has evolved to include a wide range of illegal activities such as bribery and corruption, cybercrime, money laundering (MoL), fraud, and embezzlement. The impact of financial crime is not only financial, but it also undermines the stability and integrity of financial systems, threatens national security, and damages the trust and confidence of the public in financial institutions. In this thesis, we will focus on MoL and its aspects.

MoL, as an action, is a process of hiding ill-gained funds in a legitimate financial environment which consists of three steps: placement, layering and integration (see Fig. 1). In the placement stage, money received via criminal activities is introduced to a financial system. To hide the origin and confuse financial institutions, the money is then passed through a convoluted network of actors in a process called layering. At the very last step, integration, the "cleaned" money is transferred to the owners, who then can legally use the funds. The aim of these steps is to hide the origin and the real beneficiaries of financial transactions and funds.

MoL is a global phenomenon that has become increasingly prevalent in recent years. As the act of disguising the proceeds of criminal activity as legitimate funds, MoL poses a significant threat to the integrity of financial systems and the rule of law. As such, it has become a pressing concern for governments and law enforcement agencies worldwide, who are grappling with the challenges of identifying and disrupting these illicit activities. According to United Nations Office on Drugs and Crimes' (UNODC) estimations, MoL constitutes 2 to 5% of global GDP, which equates to 800 billion to 2 trillion US dollars per year [UNO]. Such actions enable the realization of criminal proceeds, terrorist financing and wide-scale tax evasion [Ung17]. MoL has a major impact on society and the impact is felt in many ways - directly by governments under-receiving tax money thus some of the critical social infrastructure such as schools and hospitals are not going to be constructed; and indirectly by laundered money being used for ill purpose, e.g. terrorist financing, drugs, arms and human trafficking.

Driven by regulatory requirements, financial institutions deploy a variety of monitoring mechanisms to detect MoL activity across the above three phases (placement, layering and integration) so as to report it to the relevant financial authorities. The task of such monitoring systems is to sift through the financial data and detect potentially illicit activity in customers' accounts that might be associated with MoL. In order to automate the detection of potential MoL activity, financial institutions make use of various tools, including Machine Learning (ML)-based monitoring systems. Such systems raise alerts whenever the behavior

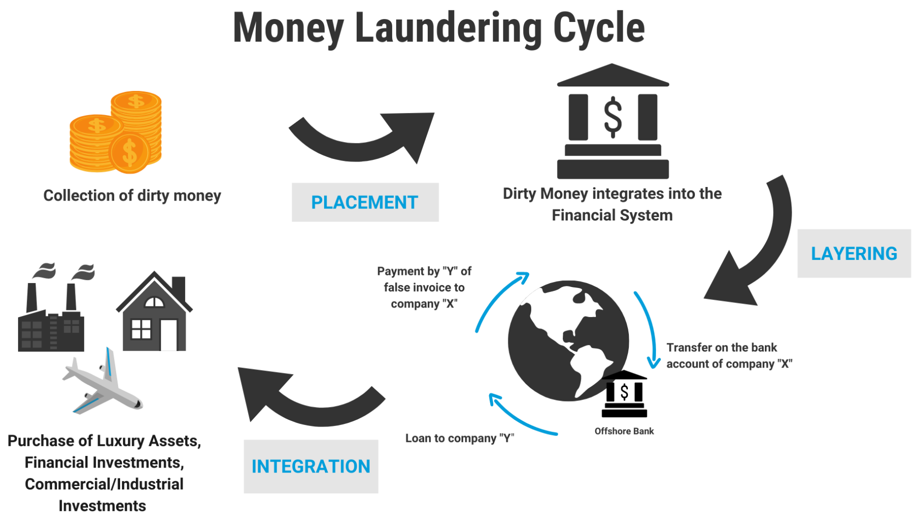

Money Laundering Cycle
Collection of dirty money
PLACEMENT
Dirty Money integrates into the Financial System
LAYERING
Payment by "Y" of false invoice to company "X"
Transfer on the bank account of company "X"
Offshore Bank
Loan to company "Y"
INTEGRATION
Purchase of Luxury Assets, Financial Investments, Commercial/Industrial Investments

Figure 1. Schematic view on MoL cycle (copied from [UNO])

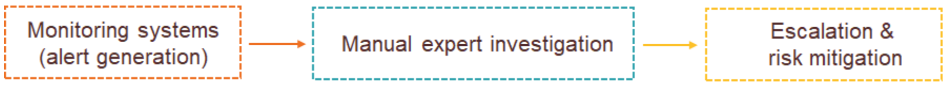

Monitoring systems (alert generation)
Manual expert investigation
Escalation & risk mitigation

Figure 2. Schematic view on alerts processing

of a customer suggests possibly illicit activity, as schematized in Fig. 2. Nowadays, such automated monitoring systems are indispensable since manual transaction monitoring is infeasible due to large amounts of generated transaction data.

Traditional monitoring systems for AML (Anti-Money Laundering) are rule-based [Gao+09]. Rule-based systems are essentially a direct continuation of the logic of AML specialists investigating MoL cases but systematized to the form of an algorithm. These systems are easy to design without deep technical knowledge and relatively easy to interpret, but at the same time, they generate a large proportion of false alerts, which in turn creates a high workload for AML experts. According to McKinsey & Co [Bre+], fundamentally, this drawback stems from the fact that rule-based systems are not able to capture complex relationships. Meanwhile, criminals are trying to find weak and unregulated or less controlled places in the financial system and legislation, thus MoL schemes are constantly evolving and improving. Given this fact and the availability of vast volumes of transaction data in banking systems, there is fertile ground for complementing rule-based systems with ML algorithms.

1.2. Research Method

The goal of this research work is to design a series of artifacts that together provide a complete and all-encompassing solution for automated MoL detection using ML techniques. We follow the Design Science in Information Systems research method proposed by Hevner et al. [Hev+08] when designing such a solution. The method proposed by Hevner et al. outlines the following seven guidelines to address a research problem in the area of Information Systems:

- **GL1. Design as an artifact.** The final outcome of the research work should be one or several purposeful IT artifacts. Proposed artifacts should help answer the first research question stated in this thesis *RQ1. "How to design an efficient system for MoL detection?"*.
- **GL2. Design evaluation.** Each of the presented artifacts has to be correctly evaluated while the quality, utility and efficacy of the artifacts clearly demonstrated. By following this guideline we are answering the second research question *RQ2. "How to measure the quality of the MoL monitoring system?"*.
- **GL3. Problem relevance.** Presented research work should provide a problem solution for one or more relevant problems and at the same time the problem should be such that has not been addressed before or there is still space for improvements in state-of-the-art provided solutions. By following this guideline we are answering the third research question *RQ3. "How to cater for practical imperatives when designing the MoL monitoring system?"* and the fourth research question *RQ4. "How to make the designed system resistant towards different classes of illicit behavior?"*.
- **GL4. Research rigor.** The construction and evaluation of the presented artifacts must be defined in a formal way using rigorous methods for the sake of reproducibility, consistency and coherence.
- **GL5. Research contributions.** The produced artifacts must provide a novel solution to the problem or solve the problem more efficiently than it has already been solved.
- **GL6. Design as a search process.** Problem solution should be a result of the iterative search process with the goal of finding the optimal solution. A problem space, as well as the solution criteria, should be defined and the solution to be improved if necessary.
- **GL7. Communication of research.** All the research outcomes must be presented to both academics and practitioners.

In this thesis, the above guidelines are implemented as follows.

(*GL3*) The problem of automated MoL detection has been studied by numerous researchers but the research area is constantly changing - criminals are never resting and developing new methods for bypassing the existing financial security, thus making the research in MoL detection always relevant. The financial sector is growing as well with electronic financial instruments dominating the market and being much more widely used by society than it was even in the previous decade. This opens new challenges for automated MoL detection that are largely unexplored such as scalability and practical usability. Lastly, the solution space dramatically increased since in recent years ML technologies experienced a huge leap in algorithmic and technological advances. Answering *RQ3* implies designing the MoL detection system that is compliant with the business requirements (is elaborated in the next subsection) and possesses ease-of-use. By the ease-of-use, we understand the ability for the outputs of the system to be understood by the end users, without special knowledge of data science methods. Answering *RQ4* is necessary in order to enhance the designed system by adding the ability to detect different classes of illicit behavior as well as the ability to detect both previously observed illicit behavior and new, previously unforeseen.

(*GL1, GL5*) Accordingly, several novel artifacts are set to be delivered to the research community: (i) a framework for individual MoL patterns detection, (ii) a MoL monitoring system that defines when and how to raise alerts when potential MoL behavior is detected, together with the family of performance metrics that estimate its performance, (iii) interpretability module that provides explanations to the raised alerts, (iv) a framework for group MoL patterns detection. Addressing the *RQ1* is complicated by several major challenges related to the input. First is a huge data imbalance, meaning that MoL patterns are very rare events with more than 99% of transactions being completely legitimate. Second is massive volumes of transnational data to be processed - depending on a financial institution, there could be millions of transactions per day to be monitored. In light of the former, the designed system for MoL detection is constrained by its ability to be run on a commodity infrastructure (non-GPU infrastructure) commonly owned by financial institutions.

(*GL4*) In the thesis, we formally described all developed artifacts and clearly explained how to reproduce the results (except for the parts of the solution purely based on the private intellectual property of the financial institution that accommodated the research). The produced artifacts descriptions are supplemented with instructions on how to tune them for better use.

(*GL2*) The artifacts are evaluated on the real-life large-scale transactional data covering multiple jurisdictions and accompanied by actual MoL cases reported to the authorities by the financial institution that accommodated the research. The performance measuring of the designed MoL detection system is answered in *RQ2*. Since default techniques for measuring the performance of ML-based solutions are not precise enough for MoL monitoring systems, we must accompany such a system with hand-made performance metrics which are designed specifically for the business needs.

(*GL6*) All the presented artifacts are the results of a systematic iterative design process, where various approaches were considered and tested before the optimal one was selected.

(*GL7*) Two conference papers are published and one journal article during the research project. The rank of the conferences and journals that published the research work ensures that it will reach both the research community and practitioners in the field of automated MoL detection using ML techniques.

1.3. Non-Functional Requirements of a Solution

MoL prevention is a very regulated area of the fincrime thus the automated ML-based solution should comply with very strict business requirements in order for the system to be actually used when deployed in production:

1. **Accurate**. First and foremost, the MoL monitoring system has to generate alerts that are likely to lead to identifying illicit behavior that has not been previously detected. This requirement entails two sub-requirements. First, the monitoring system needs to produce accurate alerts. A high number of false positive alerts will put a high workload on subject-matter experts who must act upon them. A high number of false negatives, in turn, makes the system useless since it misses a lot of illicit activity. A second sub-requirement is that the system should not generate alerts over and over again for the same customer if the new alert does not bring any additional information over previous alerts. Redundant alerts consume valuable time from the investigators.
2. **Robust**. The system must be robust, in the sense that it can detect a wide range of illicit behavior. In particular, since MoL can occur in individual accounts (individual MoL behavior) or via a collection of accounts (group MoL behavior), the system must be able to detect both individual MoL patterns as well as group MoL patterns. At the same time, the system should be able to detect both known MoL patterns and previously unforeseen patterns.
3. **Timely**. A MoL monitoring system needs to generate timely alerts. Indeed, the earlier the MoL activity is detected the sooner a financial institution will be able to take corresponding mitigation actions. Late alerts may bring monetary and reputational losses for the financial institution, while premature alerts may result in insufficient evidence of MoL to proceed with an investigation. Timeliness requirement needs to be considered in the assessment of the quality of a MoL monitoring system. This requires us to adjust the way the quality of the ML-based monitoring systems is evaluated, as standard approaches for assessing the quality of ML models do not take the time aspect into account.
4. **Actionable**. The detected MoL patterns must be investigated by AML domain experts before actual mitigation measures are taken by the financial institution. Thus, the raised alerts have to be such that they can be investigated and validated by a human.
5. **Interpretable**. A MoL monitoring system needs to produce interpretable alerts so that the AML domain experts can determine which alerts are worth further investigation and in what direction that investigation should proceed. Well-explained alerts shorten the time needed to investigate an alert, and hence allow more alerts to be checked. Related to the above, an ML-based monitoring system needs to provide accurate (calibrated) estimates of the

probability that a given customer engages in illicit behavior so that a domain expert can use this information to assess the related risks.

6. **Scalable** The developed MoL detection system must be able to process large amounts of customers' transactions in a reasonable time. This includes training the underlying ML model on large transaction history datasets and applying the trained ML model in such a way that running it periodically with relatively standard computing resources is feasible. Scalability also implies that with the growth of a customer base, the solution is still usable within a meaningful time period, especially when it comes to graph-based algorithms with non-linear complexity.

1.4. Contribution to the Research Area

This thesis presents four main contributions to the research area. The first contribution is a framework for individual MoL patterns detection and the first contribution answers RQ1 stated in this thesis. We address the question of how to train an accurate ML model for MoL detection while focusing on two aspects: scalability and imbalance resistance. A two-layered architecture is proposed for training the ML model for detecting potentially illicit MoL behavior. In this architecture, two classifiers are applied sequentially to the input samples. The first layer classifier relies on a simple Logistic Regression and a relatively small number of features related to the type of customers and their transaction volumes, which filters out clearly non-illicit customers. The second layer then uses a much larger set of features and a more complex classification approach based on extreme gradient boosting trees, in order to classify heightened risk customers into potentially illicit and non-illicit. The proposed framework incorporates a range of approaches for feature extraction, including mean-encoded categorical features, statistics based on an aggregation of time series data, customer ego-network statistics, and features extracted from stochastic models used to capture the dynamics of transaction histories.

The second contribution includes the design and development of a MoL monitoring system which defines when and how to raise alerts in case potentially illicit MoL behavior is detected. Generated alerts are designed to be accurate, non-redundant and timely. Together with it, we designed a family of metrics that estimate the performance of the presented monitoring system more precisely than standard ML metrics since standard ML metrics do not have the capacity to capture the time aspect. The second contribution answers RQ2 and RQ3 of this thesis.

The interpretability module that extends raised alerts with textual explanations constitutes the third contribution to the research area. The interpretability of the alerts is a crucial business requirement since it allows to shed light on the reasons for alert generation as well as hints at where to start the investigation process from. The third contribution partially answers the RQ2 raised in this thesis.

The fourth contribution is a framework for group MoL behavior detection which answers RQ4 stated in this thesis. The developed MoL detection system uses data about financial connections between actors derived from transaction histories, to construct a (partial) social network and then detect illicit group behavior over this network. The targeted problem is to detect group MoL behavior patterns from transaction data without assuming: (1) that such patterns have been previously observed and/or detected; (2) that all transactions between all parties are known. The first assumption cannot be made in order to develop an algorithm that is robust against old MoL schemes which are modified and new emerging MoL schemes which are constantly developed. The second assumption essentially cannot be made in the context of a single financial institution. A single financial institution has access only to transactions conducted within its boundaries – it cannot see money flows between one or more parties that are performed outside of the boundaries.

All the artifacts have been developed and evaluated on a real-life large-scale dataset consisting of customer profiles, transaction histories and labels provided by AML experts from three separate jurisdictions. The outcomes have been evaluated in vitro - via computational experiments on historical data and in vivo - using live feedback from actual domain experts.

1.5. Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 provides the necessary background to understand the scope of the problem addressed in the thesis, including the definition of MoL, strategies for MoL tackling, challenges related to it and how the automated solution looks in general. Chapter 3 reviews the state-of-the-art with a focus on previous achievements in the area of automated MoL behavior detection. Chapter 4 presents the framework for individual MoL patterns detection. Chapter 5 focuses on ease-of-use for the ML-based MoL detection system, including monitoring system design and alerts interpretability layer development. Chapter 6 describes the framework for group MoL patterns detection. Chapter 7 concludes the thesis by summarizing the contributions and outlining directions for future work.

2. BACKGROUND

This chapter provides the information necessary to understand the scope and context of the thesis as well as provides the prerequisites for understanding the contribution to the research area. Section 2.1 gives a wider overview of MoL as a phenomenon including its history and common MoL schemes. Section 2.2 makes an overview of how MoL is tackled worldwide and provides examples of high-profile MoL cases. Section 2.3 presents the high-level AML lifecycle and describes how a monitoring system's outcomes are processed when used by a financial institution. Section 2.4 describes challenges related to MoL detection with an emphasis on ML-based MoL detection systems. Lastly, Section 2.5 reviews different approaches for MoL monitoring system designs and makes their comparison.

2.1. Money Laundering

MoL is a sophisticated illegal activity, where a group of economic actors collaborate to obscure the origin and the real beneficiaries of monetary funds and transactions. It can, for example, involve convoluted transaction schemes, accounts in off-shore centres, multi-layered ownership structures and other concealment techniques.

No one knows when the very first MoL case occurred but historians are arguing that MoL exists for as long as financial instruments exist. American historian Sterling Seagrave, in his book "Lord of the Rim" [Sch96], states that the history of MoL goes back over 2000 years ago when Chinese entrepreneurs were trying to hide finances gained from the banned forms of commercial trading. They were doing it in a form that still exists nowadays - by purchasing physical goods, moving them to remote Chinese provinces and reinvesting in the local economy.

The history of MoL, as it is known today, started in the early 1930s during Prohibition in the United States. The government of the USA have prohibited any activity related to alcohol trading which created a shadow lucrative niche rapidly occupied by organized crime. Criminals had to legalise funds obtained by bootlegging and one of the common strategies those days was to create an outwardly legitimate business based on a large number of small transactions and blend in the illegal funds there. It is speculated that the term "Money Laundering" comes from this strategy when Al Capone was prosecuted for the usage of a network of public laundries to legitimize his bootlegger income. Big attention to MoL has been drawn after the 9/11 attack in the USA which led to the Patriot Act - the act that directed financial institutions to monitor financial transactions and increase their due diligence thoroughly. Similar actions were taken by other countries shortly after.

There are multiple forms and variations that MoL can take. A few of the most common schemes are the following:

- **Structuring.** This scheme involves breaking up large sums of money into smaller transactions and depositing them into multiple financial accounts. Doing so allows for avoiding raising red flags and triggering reporting requirements that apply to larger transactions.
- **Smurfing.** This scheme is similar to structuring but involves using multiple individuals to conduct the transactions to avoid detection and not to arouse suspicion. Each person handles a small portion of the funds (usually in cash) and sends them through different channels to the ultimate beneficiary.
- **Shell companies.** This scheme involves using a complex network of companies to disguise the source, movement, and ownership of illegally obtained funds. This can be done through a variety of methods, but generally, the scheme involves setting up one or more shell companies with no real business operations or assets. These companies are often located in countries with lax regulations and laws that allow for the easy creation and operation of businesses.
- **Cash-intensive business.** This scheme involves funnelling illegally obtained cash into legitimate businesses to conceal its source and avoid detection by law enforcement authorities. The business should be such that accepts large amounts of cash on a regular basis such as restaurants, car wash stations, salons and barber shops, etc.
- **Offshores.** This scheme involves hiding the proceeds of illegal activities, such as drug trafficking, embezzlement, or tax evasion, by moving them through a complex network of offshore bank accounts, shell companies, and other financial intermediaries. Offshore jurisdictions are chosen due to strict bank secrecy laws, such as the Cayman Islands, the British Virgin Islands, or Switzerland.
- **Cryptocurrency-based MoL.** This scheme involves using digital currencies to conceal the proceeds of criminal activity and make transactions that are difficult to trace. For example, criminals might use a virtual private network to hide their IP address and use a fake identity to open a cryptocurrency wallet followed by steps to distance the cryptocurrency from its criminal origins.
- **Gambling.** This scheme involves using casinos or other gambling establishments to launder money, such as by exchanging illicitly gained cash for chips, playing games, and then cashing out the cleaned funds.
- **Dormant accounts.** This scheme involves using accounts in financial institutions that have been inactive for a long time, typically for a period of several months or years. First, money obtained through illegal activities is deposited in the account and the account is left dormant. After a period of time, the criminal withdraws the money from the account in smaller amounts, usually less than the threshold that triggers suspicious activity reporting by the financial institution.

- **Trade-based laundering.** This scheme involves using the trade of goods or services to disguise the transfer of funds, such as by over- or under-invoicing the value of goods or by using multiple invoices for the same transaction.

MoL is often described as an action involving three steps: (a) placement – transferring illegal gains into the financial system in an unobtrusive fashion; (b) layering – a set of convoluted transactions for hiding the origin of the money; (c) integration – re-investment of the laundered funds into the real economy. In light of this, structuring, smurfing and shell companies are MoL patterns that are mostly concerned with the layering and integration of transactions and detection mechanisms for these patterns are incorporated via network encoding in Contribution 4. In turn, cash-intensive businesses, offshores, cryptocurrency-based MoL and dormant accounts are MoL patterns that are mostly concerned with the placement and integration of transactions. Detection mechanisms for these patterns are incorporated via robust feature engineering used for transaction encoding and presented in Contributions 1 and 2. Trade-based laundering is an example of a MoL pattern that is out of the scope of detection in this thesis.

It is important to note that these MoL patterns are just a few examples of the many different MoL schemes that exist and that criminals are constantly designing new ways to launder money.

2.2. Anti-Money Laundering

Governmental authorities fight against MoL, in part by setting regulations on financial institutions. In case of violation of the regulator's requirements, the regulator imposes a punishment on a financial institution that fails to meet the expected financial security level. Financial fines are among the most common punishments imposed but in rare cases, there are restrictions on certain financial services as well. There are several examples in modern history where financial institutions were proven to expose weak financial security against MoL and received punishment from the financial authorities. In 2022 Danske Bank was fined €470 million over an international MoL scandal and also agreed to pay $2 billion to resolve fraud investigations against US banks. For the failure in MoL prevention, HSBC was fined $1.9 billion in 2012. Goldman Sachs violated MoL prevention regulations in several countries at the same time and as a result, it was fined $2.9 billion by the US government, $3.9 billion by the Malaysian government and an additional $350 million by Hong Kong. Bank of Credit and Commerce International in 80th and 90th of the last century was found to be associated with MoL in the amount of $23 billion and eventually went into liquidation.

Not only financial institutions are discovering MoL cases, there are numerous organisations that specifically target identifying global MoL such as the Organized Crime and Corruption Reporting Project (OCCRP), the International Consortium of Investigative Journalists (ICIJ) and Transparency International (TI).

"Troika Laundromat" where billions of dollars of illicit Russian funds were moved through the network of secret off-shore companies has been identified by the OC-CRP in 2019 [OCC]. Tremendous work was done by the journalists from ICIJ, including Panama Papers in April 2016 [ICIc], Bahama Leaks in September 2016 [ICIa] and Pandora Papers in October 2021 [ICIb]. In those investigations, millions of documents (of multiple terabytes worth of data) from offshore law firms were leaked to the public and later on analysed by experts. The investigations revealed the involvement of world leaders, heads of state and government and high-ranking politicians in the laundering of funds, the total worth of which is estimated in tens of trillions US$. TI in November 2017 published findings [Int] of the United Kingdom registered companies being involved in laundering an estimated £80 billion from 13 countries.

2.3. AML lifecycle

A generalized AML lifecycle in a simplified view consists of three stages: monitoring, investigation, escalation and risk mitigation (see Fig. 3). As a first step of the AML lifecycle, a MoL monitoring system raises an alert based on the signal from a MoL detection system. A MoL detection system could be based on ML model, a rule-based detection system, escalations from branch offices or even tips from external sources. Then AML domain experts manually analyse all the alerts in the wider context of the customers' financial behavior, history and counterparties where the standard procedure is to check at least past half a year of financial history. Some alerts are immediately classified as non-illicit while others require deeper investigation, for example, an AML domain expert may ask for additional documents and explanations from a customer. Finally, when an investigation is done, the AML expert decides whether there are reasonable grounds for filing a suspicious activity report (SAR) to a Financial Intelligence Unit (FIU), otherwise, the case closes in the absence of such evidence. Additionally, to minimise risks related to potential MoL, certain mitigation measures could be taken, such as limiting financial products allowed to be used by the customer or even interrupting customer relationships.

Focus of this thesis is the design and development of the MoL monitoring system - the first stage of the AML lifecycle. The monitoring system raises alerts using ML-based MoL detection systems targeted at individual MoL patterns and group MoL patterns. Individual MoL patterns detection system constitutes Contribution 1 and is described in Chapter 4 while group MoL patterns detection system constitutes Contribution 4 and is described in Chapter 6. The design of the MoL monitoring system constitutes Contribution 2 and an interpretability module to enhance raised alerts with correspondent explanations constitutes Contribution 3, both of them are presented in Chapter 5. The interpretability module does not belong to the first stage of the AML lifecycle rather it is a part of the investigation stage since it helps domain experts with the investigation of raised alerts.

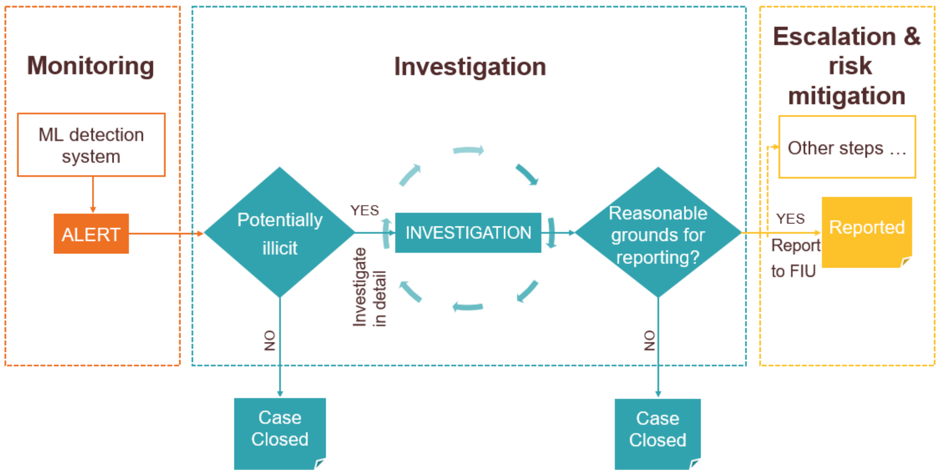


Figure 3. AML lifecycle

Every raised alert about a potential MoL case has to be checked within a period of time defined by the financial regulator. The length of this time window could vary for different jurisdictions but the standard number is 30 days. Otherwise, the regulator has to be notified as soon as possible about the breach and the financial institution may or may not be punished because of the breach.

From the legal point of view, only FIU can confirm MoL activity while financial institutions could only suspect it. At the same time, there is no two-way communication between financial institutions and FIU. This means that whenever a SAR is submitted, there is no feedback on whether the SAR was considered to be related to MoL or not. The information could only be retrieved from alternative sources such as open court cases or news in media about discovered MoL scandals.

Submitted SARs are used as labels for ML model training. In addition to SARs, it is possible to use investigated but not reported MoL cases for some of the use cases and typologies. Investigated MoL cases went through an initial check by AML domain experts where they requested additional information from customers, so there was a human validation. Investigated cases are generally considered to bring a weaker signal than the reported cases but in the absence of a sufficient number of SARs to train the model that could be a working alternative.

2.4. Challenges related to MoL detection

This thesis is concerned with the use of data and ML techniques to detect MoL. The availability of large datasets in combination with ML techniques opens numerous opportunities for AML such as improved detection capabilities, broader generalizability, better risk assessment, etc. However, data-driven MoL detection is challenging due to multiple reasons that are coming from technological advances of the world and the nature of the MoL phenomenon:

- **Massive amount of data.** Due to the present technological leap, modern society is moving towards a cashless state with notable speed. Together with it, the globalisation trend has significantly increased the amount of funds circulating around the globe. All of those factors have made digital payments to be dominant nowadays and, therefore, there is a massive amount of data to be sifted through if the task is to detect MoL. There are millions of transactions to monitor every day, even within one financial institution.
- **Extremely diverse data.** As a consequence of the above, there are numerous financial products that could be used as potential tools for laundering funds: wire payments, card payments, international payments, crypto payments, cash, etc. This means that ML models designed to detect MoL patterns should potentially have huge feature sets since there are multiple products that could be used. And for some MoL patterns, it is impossible to split the detection into several models focusing on different products because actors could launder money through different channels, e.g. placement happens via cash deposits and integration happens via wire transfers.
- **Huge data imbalance.** MoL is an extremely rare phenomenon and only tenths or hundredths of a percentage point of all transactions from a financial institution are falling under suspicion of being related to MoL, even less of them eventually appear to be related to it. This makes the usage of ML techniques for AML purposes very challenging since the ML model will be trained on enormously imbalanced data.
- **Sophisticated patterns.** At the same time, MoL patterns are not only extremely rare, but they are also deliberately hidden by money launderers. In most cases, MoL is a complex action and is usually done by organized criminals whose task is to mask illegitimate funds among legitimate ones, making them harder to detect.
- **Dynamic patterns.** Money launderers are searching for weaknesses in financial security and developing new MoL schemes, thus the modus operandi of the MoL world is constantly changing. This means that if ML models are used for automated MoL detection then new features describing new MoL schemes should be regularly designed and incorporated, new patterns labelled by the domain experts should be introduced into model training, and finally, the ML models should be often retrained.

Massive amounts of data and data imbalance challenges are directly addressed by **RQ1** and uncovered in Chapter 4, while dynamic MoL patterns challenge is addressed by **RQ4** and uncovered in Chapter 6. The rest challenges - data diversity and patterns' complexity - are indirectly addressed in both Chapters 4 and 6

2.5. MoL Monitoring

An ML-based monitoring system is a system that uses an ML model to predict whether certain events are going to happen to monitored entities or to classify whether the monitoring entities achieved certain states. A monitoring system used in the context of MoL implies a system that uses a MoL detection system to monitor MoL behavior over time meaning that at every moment of time (discrete or continuous), the monitoring system must provide its decision regarding each monitored entity. A monitoring system consists of the input data orchestration part, the ML-based detection model that associates risk scores with monitored entities and a policy that dictates when and how to raise alerts based on those risk scores. In this thesis, the input data orchestration part is out of scope so by MoL monitoring system we understand an ML-based MoL detection system and a correspondent policy for alerts generation.

Monitoring systems are common in various domains, including equipment failure monitoring, predictive machinery maintenance, card fraud detection, MoL detection, etc. Such systems are particularly useful in domains where manual monitoring is infeasible and where it is not possible to codify all possible behaviors that may lead to an alert in the form of a rule set. This is the case in the context of MoL detection, as there may be millions of transactions per day (not suitable for manual classification) and where it is not possible to characterize all possible behaviors of interest via a set of rules.

Essentially, there are two conceptually different approaches when it comes to automated ML-based MoL patterns monitoring - online (or proactive) and offline (or reactive). Both approaches have their own advantages and disadvantages, which will be discussed further. For both online and offline monitoring, the ML detection model training is the same - static historical data (transactions and labels) is extracted from the database and classified entities are formed, whether those are transactions or customers making those transactions. After that, features describing the entities are calculated and finally, the ML model is trained, where feature selection and hyperparameters tuning are also considered to be stages of the model training. The difference in online and offline approaches comes from the model inference stage, the stage where an already trained model is actually applied in production to detect/prevent MoL.

Chapter 5 of this thesis focuses on the design and development of the offline MoL monitoring system as well as tackling challenges related to it which essentially constitute **RQ2** and **RQ3** raised in this thesis.

2.5.1. Online monitoring

In the online approach, the analysed entities are transactions and the model inference happens in a streaming format. As soon as the transaction has happened, it is characterised by a feature vector, which is then passed to an ML model for classification. Features extracted are mostly focusing on the description of the

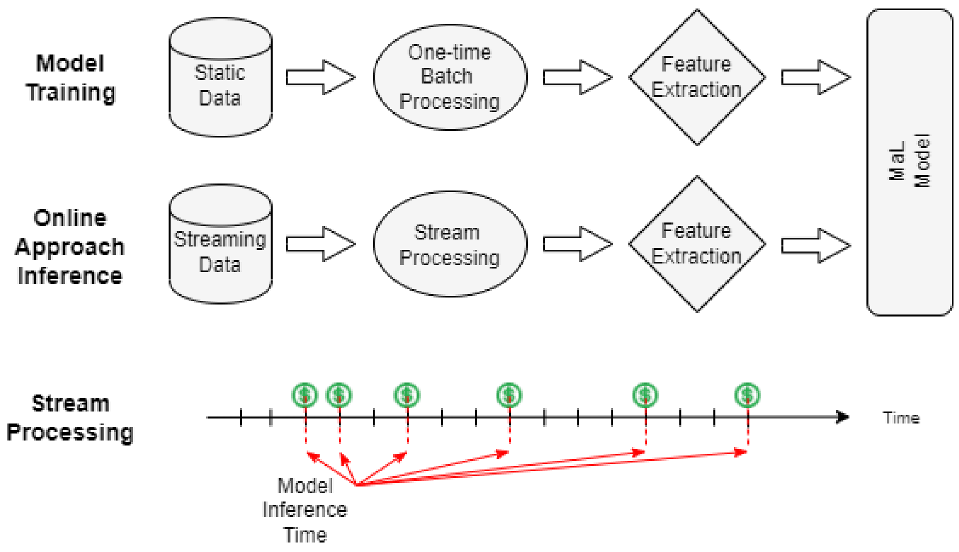


Figure 4. Online monitoring schematic view

transaction itself, as well as the sender and receiver of the transaction. The approach allows suspending the analysed transaction, in case there is a suspicion that the transaction is related to MoL, or to decline if there are signs of certain MoL activity detected. That undeniably is the main advantage of such an approach - it allows to suspend or even decline suspicious transactions. Stages of the online monitoring schematically depicted in Fig. 4.

Despite its advantages, there are several technical challenges and complications with online monitoring. First, since it happens in real-time, the classifying ML model must make a decision in a very short period of time, hence the feature vector cannot be extremely large or include computationally heavy features. Consequently, the online monitoring approach does not allow for taking into consideration the long financial history preceding a monitored transaction. For monitoring systems targeting different financial products, this particular limitation could be less strict or more strict, depending on the financial product. For example, for international payments, the SWIFT service level agreement allows to analyse transactions for a longer period of time than for card payments. Whereas VISA/Master Card service level agreement implies card payments happen almost instantly, thus ML model must make a near-instant decision on analysed card payments. Additionally, online monitoring requires significant investments in the development of the streaming infrastructure that allows the processing of millions of transactions per day within the above-stated requirements.

In the fincrime area, an online monitoring approach is usually used for scam and card fraud prevention. Mostly that is due to the fact that customers of a financial institution are victims and the financial regulator requires financial institutions to reimburse customers for not preventing transactions related to scams or card fraud. Thus, it is in the direct interest of a financial institution to prevent such transactions to minimise the amount of reimbursement and minimise the overall customer dissatisfaction caused by unpleasant user experience. In addition to the above, scams and card fraud schemes are mostly based on social engineering and do not use complex financial operations, which makes such patterns somewhat easier to detect for an ML model and more suitable for online monitoring.

As for MoL detection, online monitoring is not often used. First of all, not in every jurisdiction does the financial regulator require suspending or declining transactions suspected of being part of MoL activities. What is required from financial institutions is to detect potentially illicit behavior in time and submit a SAR to FIU as soon as possible. Hence, online monitoring for MoL purposes is not always needed. Secondly, it is much harder to implement online monitoring for MoL behavior since MoL patterns are much more complicated than the ones of card fraud and scams. The standard procedure for a domain expert to investigate a potential MoL case requires checking at least the past half a year of the customer's financial history. Therefore, the approach requires a much bigger feature set and hand-made complex features which are technically challenging to calculate in a streaming format. Lastly, not every financial institution has the proper infrastructure to perform online MoL monitoring since ML-based detection is a relatively new strategy while financial institutions may exist for centuries and have rather old technologies in-house.

2.5.2. Offline monitoring

Offline monitoring, in contrast to online monitoring, happens post factum, i.e. the MoL has happened and the task of the offline monitoring system is to detect it, not to prevent it. The analysed entities could be transactions or customers who made those transactions and the model inference happens in a batch-processing format, see the schematic illustration of the offline monitoring in Fig. 5. When the analysed entities are transactions, transactions are accumulated until the scheduled model inference time and then analysed in batches, the logic of the feature extraction is similar to the one of the online monitoring setup. In case the analysed entities are customers, then by the moment of scheduled model inference, the snapshots of the recent financial history are formed for every customer (or a segment of customers if the monitoring system targets a specific segment, e.g. private, business, high-net-worth individuals). Extracted features describe customers' financial behavior in general - spending behavior, patterns of income, usual counterparties, cash usage behavior; rather than any specific transaction of a customer. The biggest advantage of the offline monitoring approach is that entities are not monitored in real-time which gives more time for data gathering, feature engineering and model inference. Plus, this approach doesn't require high-throughput infrastructure to be used in the inference stage.

Despite its advantages, the offline monitoring approach is exposed to a number of technical challenges. An obvious challenge comes from the fact that the anal-

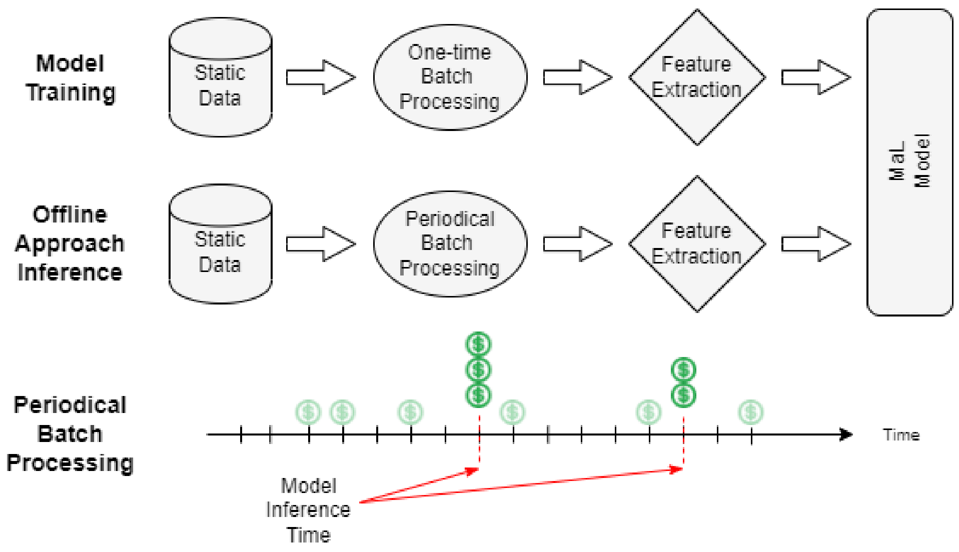

Model Training
Static Data
One-time Batch Processing
Feature Extraction
MaL Model
Offline Approach Inference
Static Data
Periodical Batch Processing
Feature Extraction
Periodical Batch Processing
Time
Model Inference Time

Figure 5. Offline monitoring schematic view

ysed snapshots of customers' financial behavior belonging to one customer are highly overlapping with each other (if the monitoring periodicity is shorter than the considered history of the entity). This means that if at some point an event appears on the customer's account, this event will be present in several consecutive snapshots of the customer's financial activity. Therefore, this brings the issue of reoccurring alerts for the same customer, when a customer gets consequent alerts for the same target event detected. The reoccurring alerts provide intensified manual workload demand, while usually, those consequent alerts do not reveal unseen illicit behavior.

An additional level of complexity comes from the fact that alerts on illicit financial behavior should be given at the moment when they are useful, which opens an issue of alert timing. The issue of alert timing implies the fact that the alerts must be raised in a proper time period. Late alerts may bring monetary and reputational losses for the financial institution and very early alerts may result in insufficient evidence of MoL. While alert timing is not an issue for online monitoring where an alert is raised as soon as the potentially illicit transaction took place, for offline monitoring it is not obvious at which model inference time to raise an alert.

In the end, the evaluation of the offline monitoring system itself is not trivial since classic ML performance metrics do not capture aspects of offline monitoring: timing and reoccurring alerts. Standard ML metrics do not grasp business requirements and can misrepresent actual monitoring system performance since the monitored entities are overlapping snapshots of customers' financial history.

In the fincrime area, offline monitoring is usually used for MoL detection due to the complexity of MoL patterns, which require extensive and complex feature engineering as a prerequisite for ML models used. At the same time, for card

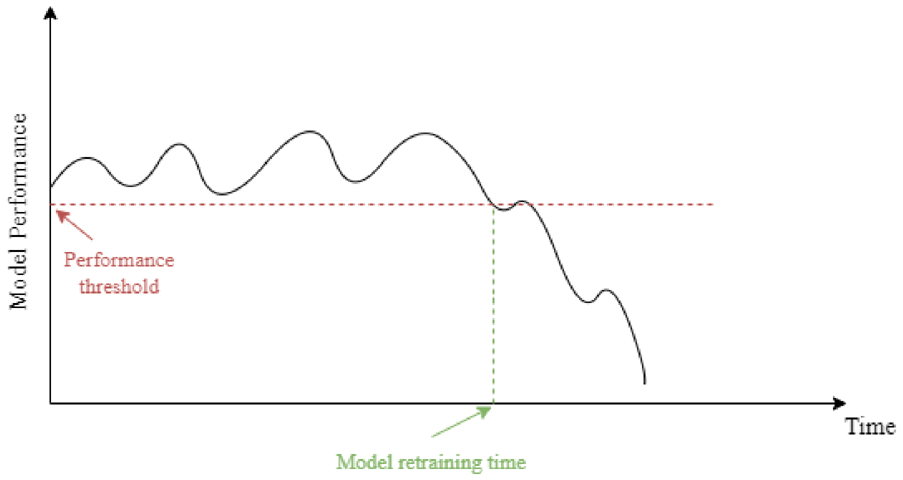


Figure 6. Performance monitoring dashboard example

fraud and scam detection, the offline monitoring strategy is not applicable since it does not prevent fraudulent transactions from happening and financial institutions still have to reimburse financial losses to their customers.

2.5.3. Monitoring system maintenance

The monitoring system requires careful maintenance after being deployed in production. The first aspect to be taken into account is model retraining which needs to be done due to several reasons. First and foremost, MoL patterns are changing with time since money launderers are quickly adapting to defence strategies set up by financial institutions. To be prepared, financial institutions have to incorporate up-to-date labels into the retraining process, and those labels have to include MoL patterns that are desired to be detected. The second aspect lies in the general change in customers' financial patterns - the phenomenon most commonly known as data drift. The most notable examples are inflation, due to which customers' average spending increases, and changes in financial product usage, e.g. cryptocurrency usage boom and cashless trend. Not often retrained ML models are prone to degrade in performance since they were trained to detect outdated patterns. Retraining can be done either scheduled, i.e. in a predefined moment of time, or based on performance monitoring dashboards when a significant performance drop is detected (see Fig. 6).

One non-obvious critical aspect of the retraining ML models used for monitoring is the change in the models' predictions after retraining. Although that is expected that after retraining the model, it should provide different risk scores but at the same time, newly incorporated labels can shift the model's prediction in such a way that it doesn't flag the behavior it was flagging before. Hence the retrained model is often put in a shadow mode alongside the old model still running in production while the proper analysis of predictions of the two model versions is done.

Last but not least, the MoL monitoring system requires often tuning of the alerts interpretability module in order to get as precise explanations as possible. This, in turn, requires a constant feedback loop between ML model developers and the consumers of those explanations - AML domain experts.

3. STATE OF THE ART

There is an extensive variety of publicly available research works that study certain aspects of MoL detection and prevention, propose algorithms that were modified to specific business needs, propose completely novel algorithms or present the entire frameworks that describe the full cycle of a MoL problem solution. All of the research works have their aspects depending on the problem they solve, the data they possess, the countries the data is from, and which financial product they target (international payments, card payments, cash, crypto, a combination of those).

At the same time, a lot of research works are simply not exposed since for some companies, the solution for a MoL problem is just a core business model. As for the financial institutions, the topic is quite sensitive and not every financial institution is willing to take one of the many risks while publishing the research work:

- A risk of accidentally leaking sensitive customers' data
- A risk of exposing the fundamental vulnerability or a systematic weakness of the institution
- A risk that the published research will help criminals to bypass the financial security of the institution in the future
- A reputational risk that the presented financial security will appear not as strong as the customers of the financial institution expected

One of the biggest challenges in the research in this field is data availability and data quality [JI23]. There are a few open datasets on the internet but they do not represent actual MoL data (too small dataset, data imbalance is far from actual, MoL patterns are outdated or significantly deviate from the one seen in practice). Elliptic dataset [Web+] and Ethereum dataset [Ali] are the most common publicly available datasets used by the research community. Those are the two largest public datasets with labelled cryptocurrency transactions related to MoL. The benefit of using either of those datasets is that it is possible to compare the results of the proposed MoL detection framework against results obtained by other researchers that used the same data. The drawback is that those datasets contain data about cryptocurrency transactions and correspondent MoL which is hardly comparable to data coming from financial institutions. Otherwise, if a researcher is not related to a financial institution or financial authority, then the only available option is to rely on synthetic data. A massive downside of using this option is that building a solution on synthesized transactions and synthesized suspicious patterns can show fairly good results which simply could be an indication that the proposed solution has reverse-engineered the data synthesizer, thus making the research questionable.

A lot of aspects of MoL can be studied: impact on society (direct as losing money and indirect as the influence of laundered money on subsequent crime);

separate stages of MoL (placing, layering and integration); the connection between MoL, terrorist financing and drugs trafficking; detection of MoL with knowing the predicate crime (i.e. how to design algorithms which will target specific MoL schemes) and without knowing predicate crime (i.e. using data mining techniques only); automation of MoL detection using data mining and ML techniques. The analysis of state-of-the-art is focused on studying the achievements of the research society on the former question - automatization of MoL detection using data mining approaches and ML usage specifically.

This chapter makes an overview of research done in the area of automated detection of individual MoL patterns (Section 3.1), MoL behavior monitoring (Section 3.2) and automated detection of group MoL patterns (Section 3.3).

3.1. Automated detection of individual MoL patterns

This section makes an overview of the state-of-the-art in the area of automated detection of individual MoL patterns, both ML-based and non-ML-based approaches and summarizes the achievements of the research community in answering **RQ1** raised in this thesis.

3.1.1. Non-ML approaches

The application of data analytics techniques for financial fraud and MoL detection goes back to the 1990s [PA14]. A notable pioneer is the FAIS expert system [Sen+95], developed in the Financial Crimes Enforcement Network (the US Financial Intelligence Unit). FAIS relies on a manually maintained set of rules to identify parties with potentially illicit transactional patterns. According to [Che+18] such rule-based AML systems have dominated industrial standards for decades. Rule-based systems have two strong advantages: they are easy to interpret and can be designed by domain experts with a little technical background. However, they offer limited flexibility, are cumbersome to update and adjust to changing environments and rely solely on expert knowledge of criminal behavior [Che+18]. Even nowadays, with a boom of ML techniques usage in the industry, rule-based systems are irreplaceable and still constitute a big share of MoL detection/prevention in both financial authorities and financial institutions.

A modern example of rule-based system usage can be found in [BLS20] where authors share their experience of such a system used for MoL detection by an Italian FIU. The authors rely on a rule-based approach, mainly due to its explainability and the ability to directly incorporate the domain expertise. Rule-based monitoring in combination with behavior detection monitoring and link analysis was used as a framework in [Hel+16] to tackle MoL. In addition to it, the authors applied clustering techniques to reduce false alerts raised by the framework. Stepping ahead, some researchers have moved away from strict rules and studied fuzzy rules for the detection of MoL behavior. One of the examples is presented in [CM11]. Rule-based approaches were also used to detect specific MoL typologies. For example, authors in [But21] study the detection of funnel accounts in South Africa using a rule-based system that consists of 12 red flags. By definition, a funnel account is a business or corporate account where money is deposited in cash and then within a short time period transferred to another account in a different geographical location.

It is quite clear that rule-based detection engines cannot be an effective solution for MoL detection/prevention on their own with such an excessive growth of the financial sector and the constant development of new MoL schemes [Bre+]. Accordingly, modern literature has been moving towards statistical and ML approaches for MoL activity detection [Che+18]. A wide range of approaches has been investigated in this field, including Logistic Regression (LR), Support Vector Machines (SVM), Neural Networks (NN) and tree-based techniques such as decision trees (DT), Random Forest (RF), and gradient boosting trees (GBT).

3.1.2. ML approaches

The problem of individual MoL patterns detection using ML algorithms has been studied in many research works. A lot of the research was based on real data [Jul+20; Har+22; JI23; Che+21; AB16; Edd+21; TK19] which brings additional value and reliability to research works. But due to complications in accessibility to real-life data from financial institutions, a big part of the researchers were using public datasets such as Elliptic dataset [Ahm21; PA22; APN20; Alo+22] or Etherium dataset [Ahm21; Azi+22]. Some of the MoL detection solutions are developed and tested on artificially created data [Rai21; Lok22; MA18; LA12]. This approach is questionable since it is hard to distinguish whether the results are fairly good or the proposed algorithm just reverse-engineered the data simulator and the performance will be significantly lower if the framework faces the real data.

The real data used for individual MoL patterns detection research varies by size. Some works are based on large datasets (with hundreds of thousands or millions of transactions) [JI23; AB16; Edd+21] and some are based on relatively small datasets (under a hundred thousand transactions) [Jul+20; AQA21; Har+22; Che+21; TK19]. The research performed on large data possesses a qualitative upper hand, first of all, because the conditions are much closer to the conditions of real financial institution operation. Secondly, ML models build on large data exposed to two potential issues - huge data imbalance and model scalability - both of which must be answered in order to use the solution in real-life practice. In turn, frameworks built on small data usually do not face the issues mentioned above.

From a technical perspective, there are two approaches for individual MoL patterns detection - on the transaction level where the analyzed entity is a transaction [Jul+20; JI23; Che+21; Edd+21] and on the customer level where the analyzed entity is an aggregation of customers' transactions [Har+22; AB16; TK19]. The approaches are conceptually different, they require different feature engineering strategies and different labels to be trained on. ML models trained on transactions by the design generate timely alerts, while ML models that analyze a customer as an entity rely on additional algorithms to define where actual MoL took place and when it is the best to raise an alert. On the other side, the transaction level approach is prone to have problems with scalability since the ML model has to score each transaction, while the customer level approach does not possess this issue at such scale since the model is scoring customers as aggregations of customers' transactions.

From the perspective of ML algorithms, there is a great variety of both supervised and unsupervised approaches used. On the supervised side, most of the research works are using tree-based models such as GBT [Jul+20; Ahm21; Lok22; Har+22; Edd+21; APN20], RF [Rai21; Lok22; Edd+21; Azi+22; Alo+22] or even simple DT [WY07]. DT-based solutions provide the benefit of interpretability but they have rather limited learning capabilities due to algorithm simplicity. While ensemble tree-based models by the design lost the advantage of interpretability due to the ensemble architecture but this family of classifiers have better predictive capabilities. In general, the common conclusion is that tree-based ensemble models are performing better than other ML algorithms in the MoL domain which was confirmed by experiments of many authors.

LR, Lasso regression and Ridge regression model were tried in the following works [Rai21; Lok22; Har+22; Edd+21; PA22]. Although LR is a powerful tool, practice shows that this family of algorithms does not have the capacity to solve problems of such complicated nature as MoL. On the other hand, regression models have the advantage of being partially interpretable - the coefficients of a model give an indication of the contribution of features to the prediction. Some of the researchers applied SVM in order to detect MoL [Rai21; AQA21; PA22; Azi+22]. Usually, SVMs are used on small and medium datasets due to time constraints required for model training, especially when the problem requires the usage of non-linear kernels. This makes it practically impossible to use this family of algorithms in large-scale real-life applications.

NNs were also used to tackle MoL detection [Lok22; JI23]. The researchers exploited a wide spectrum of NN architectures from feedforward NNs [Lok22] to recurrent NNs [JI23] such as Long Short-Term Memory networks and Gated Recurrent Unit networks. NN-based solutions can outperform conventional ML algorithms (with a prerequisite of a sufficient amount of data) while also not requiring extensive feature engineering. At the same time, the main drawback of NN-based solutions for MoL tackling is their poor interpretability which is crucial for usage by financial institutions.

There is a significantly smaller number of research works available that study unsupervised ML algorithms for individual MoL patterns detection. The authors of [MA18] studied the usage of Stacked Auto Encoders and Restricted Boltzmann Machines for real-time financial fraud detection on a synthetic dataset of mobile money transactions. Autoencoders and Variational Autoencoders for MoL detection were used in [Che+21] on real-life data from a Malaysian bank. To enhance the performance and tackle class imbalance, Wasserstein Generative Adversarial Networks were used to generate artificial fraudulent transactions. Authors of [AB16] were experimenting with K-means clustering that was followed by decision rules. The research was conducted on large data of 14.5 million transactions.

The general trend in the state-of-the-art in the area of supervised ML-based systems for MoL detection is to make a proof-of-concept type of research, meaning that the potential for the ML model is analyzed in an isolated environment with one-time-of model training and testing. Unfortunately, the next step, i.e. integration of the developed proof-of-concept into the real environment of a financial institution with a study of the subsequent challenges is poorly covered in publicly available research. Another crucial aspect to cover is the ML model performance validation. The dominant majority of publicly available research works on the topic of ML-based individual MoL patterns detection perform in vitro validation. For research done on real data, it is to measure ML model performance on historical data that succeeds the data from a training time period. While for the research performed on synthetic data, it is to measure ML model performance on a synthesized holdout test set. Alternatively, it is possible to perform in vivo validation - meaning to use a trained ML model to generate a certain number of alerts and request domain experts to investigate them and give their feedback. Such an approach is very expensive and rarely used since it requires having access to the domain experts but some of the research done in cooperation with financial institutions has followed it [JI23]. For all the artifacts developed and presented in this thesis, we rely exclusively on real-life data extracted from banking systems across three jurisdictions and covering both private and corporate sectors.

Building upon the aforementioned state-of-the-art techniques, the approach outlined in Chapter 4 provides a solution for individual MoL patterns detection that was tested using real-life data from a multinational financial institution. The proposed framework uses robust feature engineering and layered architecture which enables enhanced scalability while effectively managing the prevalent high class imbalance encountered in the realm of AML.

3.2. MoL monitoring systems

This section gives an overview of the state-of-the-art in the area of MoL behavior monitoring and summarizes the achievements of the research community in answering **RQ2** and **RQ3** raised in this thesis. A MoL monitoring system is responsible for raising alerts when suspicious behavior is detected, which are then analysed by the AML domain specialists. It is crucial that these alerts are raised in a timely manner, as raising alerts too late leads to less choice of risk mitigation options (if any). This timeliness requirement has an impact on the way the quality of the MoL monitoring system should be evaluated, as discussed below. Another concern in the area of ML-based MoL monitoring is the interpretability of raised alerts. Below, we will discuss how these concerns have been addressed by the research community.

This research area possesses the biggest gap with almost no research works about offline or online MoL monitoring system designs and their aspects being published. Thus, in this section, we often refer to solutions to partially similar problems in the area of fraud, the general financial sector or predictive maintenance area.

3.2.1. Timeliness of MoL monitoring systems

The problem of designing and developing ML-based monitoring systems in the AML domain has been addressed in several previous research studies. In [Ket+21], the researchers tackled the problem of MoL detection as a problem with time factor by proposing a novel set of tailor-made time-aware features based on time-frequency analysis. The detection model was tested on real-life data of 6680 Akbank customers, 26% of which have been engaged in MoL activity. This ratio is very far from a real-life distribution, where the ratio of suspicious customers is usually very low. Thus, in typical real scenarios, an AML monitoring system needs to handle datasets with strong class imbalance. Although the [Ket+21] provides a solution for MoL detection, it does not tackle the problem of determining the time when the output of the classification model should lead to an alert.

ML-based monitoring techniques have also been applied for MoL behavior monitoring in [Cha+19]. The authors monitored a capital flow to tackle trade-based MoL using Bayesian networks. The proposed Bayes net is represented by a directed acyclic graph and a conditional probability table which are built on training data. The evaluation of the developed tool has been done by using Receiver Operating Characteristic Area Under Curve (ROCAUC), F-score and Geomean measure, without considering the timing of the prediction.

Monitoring techniques were also used to detect cellular fraud in a work which was presented in [FP97]. The authors have developed a rule-based model for alert generation to detect cellular cloning fraud. Their research has been performed on a database of call records and model performance was measured as accuracy averaged by the time of model runs. As an additional performance metric, they calculate the cost of alerts. While this latter study considers the cost of false alerts (i.e. false positives), it does not take into account the cost of generating an alert too early or too late (timeliness of alerts).

Predictive monitoring based on ML models is a common practice in the financial industry in general. One of the examples is churn prediction where customers are monitored in order to control whether they intend to break customer relationships. The problem of churn prediction by using a sequential manifold learning approach was investigated in [KL12]. Applied LR and NN models were evaluated on the E-commerce customer dataset using ROCAUC, lift and hit rate metrics.

Another example of the monitoring task in the financial sector is bankruptcy monitoring which has been investigated in [VBV17]. The authors utilized Markov for discrimination (MFD) model in the context of bankruptcy prediction modelling on a Bel-first Finance database by Bureau van Dijk. The data contains financial information of Belgium and Luxembourg bankrupt companies, excluding cases of temporary financial difficulties. The performance of the model has been evaluated by F2 score, ROCAUC and accuracy. The methods in [KL12] and [VBV17] address the corresponding monitoring tasks as "snapshot" problems (will churn or bankruptcy occur within X days). They do not consider the cost of a prediction being made too late (or too early), nor the fact that the prediction for a given sample needs to be done sequentially (multiple times) and hence that the prediction model may switch from making a correct prediction to making an incorrect one at different points in time. In other words, these studies assume a setup where an end user would want to get all predictions (for churn/bankruptcy) at some random points in time, which does not match the scenario where the end users wish to get the predictions only when such predictions require their attention.

3.2.2. Evaluation of MoL monitoring systems

In a monitoring task, an essential question is how to properly evaluate the performance of the model. Traditional performance metrics do not represent the real state of a model very precisely and can highly overestimate its performance because of several reasons. First of all, standard performance metrics do not possess the notion of time, meaning that those metrics do not distinguish between a perfectly timely alert or a correctly raised alert but with temporal lag (too late or too early alert). Secondly, it might be that the same entity (a customer or an account) is the subject of multiple alerts over a time period, as opposed to traditional classification approaches, which consider the situation where each entity is classified only once (positive or negative) and not repeatedly or continuously. The way monitoring system evaluation was targeted is described in Chapter 5.

Meanwhile, the research community has approached this problem in several works. The question of the timing of alerts has been addressed in the context of early classification of time series in [DBC15], where authors proposed a method to balance the expected gain in the classification cost in the future and the cost of delaying the decision. The method has been tested on both synthetic data and real-life TwoLeadECG datasets. The performance of alert generation systems in the predictive maintenance domain has been assessed in [Sip+14]. The authors are predicting equipment failures by mining the data from event logs of medical devices. To fulfil the timing of alerts requirement, they redesigned the definition of true positives as cases where an alert occurred in a predefined interval before failure and false positives as cases where an alert occurred outside of this interval. Final model performance is measured by a Predictive-Maintenance-based Area Under Curve as usual Precision Recall Area Under Curve (PRAUC) but with custom recall and precision based on a redefined understanding of true positives and false positives. A similar approach of performance metric redefinition has been chosen in [Zha+16]. The authors also took into account the problem of alerts re-occurrence in predictive maintenance tasks. The proposed approach was validated on two enterprise systems' log traces: the web server cluster and the mailer server cluster. The framework's performance has been estimated by custom PRAUC metric (made from custom-defined recall and custom-defined precision), predictable interval metric and predictable frequency metric. The aim of introducing those metrics is to provide a more precise estimation of an ML model performance by penalising late and early alerts.

3.2.3. Interpretability of MoL monitoring systems

A crucial requirement in monitoring systems is interpretability, since in most cases, the alerts generated by such systems have to be analyzed by subject-matter experts. We must conclude that there are no publicly available research works that describe a design or an approach for an interpretability module used to enhance alerts raised on detected MoL behavior with alert explanations. Certain works are claiming that their approaches are interpretable since the ML algorithms used there are inherently interpretable (like DT-based algorithms or linear models) and the extracted features by the design are understandable to a human [Hsi+21; Kut+21]. In practice, no approach for explanation generation nor the tests of claimed interpretability is presented.

Alerts interpretability was addressed in the area of fraud detection. Applicability of local interpretations for fraud detection model based on anomaly detection techniques was studied in [WW21] where LIME package [RSG16] was used. Another approach for local interpretations was exploited in the work of [LG22] where the famous SHapley Additive exPlanations (SHAP) package [LL17] was applied.

In the financial sector in general, the interpretability of ML-based models was studied in the area of credit scoring. In [CYY23] authors provide three types of global and local explanations for the ML model used for credit risk scoring. Model-agnostic DALEX system for ML models explanations was used in credit card defaulters detection domain by [SG22]. Counterfactual explanations were presented as a solution for credit score ML model interpretability by [Wan+23]. A counterfactual is an approach where local feature importance is defined by using the fact that a model's prediction would change if certain input features were modified while keeping the rest of the features intact.

Following the latest achievement in state-of-the-art mentioned above, the methodology outlined in Chapter 5 proposes a design for an offline MoL monitoring system which orchestrates the alert generation. It is advocated that the proposed system solves the issue of alert timeliness and the issue of an entity potentially being the subject of multiple repetitive alerts. The monitoring system is accompanied by performance metrics which are enhanced with the possibility to account for the time aspect. Raised alerts are supplemented with textual interpretations with the interpretability module being based on the SHAP algorithm.

3.3. Automated detection of group MoL patterns

This section gives an overview of the state-of-the-art in the area of automated detection of group MoL patterns and summarizes the achievements of the research community in answering **RQ4** raised in this thesis.

The problem of detecting illicit group behavior associated to financial crime has been tackled in three broad application contexts: MoL [RS18; Sav+17; Bah+18; BPI19; Li+20; Sol+16; SN18; MGS21; Sta+21], e-commerce and financial fraud [Liu+22; Leb+17; Li+21; Mag+18], and cryptocurrency fraud [CT22; PDG18; Xue+21]. In the MoL area, groups are used to provide a complex layering structure thus allowing to dissimulate the origin of funds. In the fraud detection domain, the purpose is to rapidly hide fraudulently acquired money by having it circulate through as many "hops" as possible, possibly across multiple jurisdictions, to make it difficult for law enforcement authorities to retrieve the ultimate destination of funds. Therefore, the application of group behavior in the MoL domain and the fraud domain are conceptually different - the former is focusing on hiding the source of funds while the latter is focusing on hiding the ultimate destination of funds. In the case of cryptocurrency, the goal might be either to hide the origin or the destination of funds, but the caveat is that funds typically cross the boundaries between traditional financial institutions and cryptocurrency networks.

Most of the research in the field of group behavior detection is done using real-life financial data [Sav+17; Bah+18; Li+20; Leb+17; Sta+21] gathered from financial institutions or financial authorities, or in the case of crypto analysis – from the publicly available datasets. In many studies, though, researchers rely on synthetic data due to the inaccessibility of real data [BPI19; Sol+16; MGS21].

From a technical perspective, the proposed approaches use a combination of ML and graph analytics. A subset of approaches relies on the idea of detecting known MoL topological structures, chiefly smurfing patterns and cliques [Li+20; CT22; Sta+21], isomorphic subgraphs [SAJ20] or connected components [Sav+17]. A drawback of these approaches is that actors who engage in illicit behavior are aware that financial institutions have systems in place to detect such patterns, and thus they explicitly avoid leaving such traces behind.

Another subset of approaches relies on the analysis of egocentric networks (the network consisting of a node, its direct neighbours and all their connections) [RS18; Sav+17]. The main problem with egocentric networks is that they require seed nodes to be built around, which is either computationally expensive if it is built around every graph node or requires prior knowledge to select the limited number of such seed nodes. Also, egocentric networks only allow us to analyze groups that are centred around some nodes, or densely connected groups. In some MoL schemes, a group of illicit players will rather form a sparse sub-network, where some pairs of players are distant from each other.

A larger group of approaches relies on community detection algorithms [Li+21; ABB20; Bah+18; PDG18; Xue+21; Liu+22; BPI19; MDR15; Lin+16]. In this setting, a community is a group of nodes that are densely linked between them and sparsely linked to other nodes outside the community. The idea underpinning these approaches is that groups of actors who engage in illicit behavior will typically correspond to communities. Some of the approaches in this category use Label Propagation Algorithm (LPA) for community detection [Li+21; ABB20], the Louvain community detection method [Bah+18; PDG18] or Random Walks on graphs [Xue+21; Liu+22]. Other methods propose a tailor-made adaptation of an existing community detection algorithm or a completely new one [BPI19; MDR15; Lin+16]. Finally, a non-vital but important feature of illicit group behavior detection frameworks is the ability to detect overlapping communities, examples of those can be found in [CT22; ABB20]. The ability to identify overlapping communities is beneficial for developing stronger solutions as it enables the analysis of graph nodes (such as customers or customers' accounts) from diverse viewpoints, thereby enhancing robustness.

In the case when a connected group of actors is identified using known MoL topological structures, the group by definition is considered to be of high risk. Otherwise, the detection of communities of actors must be followed up by an analysis of whether this group of actors is engaged in illicit behavior. Usually, such analysis is done using ML techniques: supervised learning [Sav+17; MGS21], which require labelled communities; unsupervised learning [BPI19; Li+20; CT22; Sol+16; PDG18; Mag+18], which do not require any labelled input; and semi-supervised learning [Leb+17], which is a mixture of both. The analysis can also be done using non-ML methods – an example of this can be a social network analysis (SNA) [SN18; MGS21] where a graph structure of a detected community is analyzed using certain metrics such as degree centrality, weighted degree centrality, betweenness centrality, and modularity.

- The degree centrality focuses on the level of input and output associated with the nodes.
- The weighted degree centrality takes into account the impact of the total weight of edges on the node's degree.
- Betweenness centrality facilitates communication between nodes.
- Modularity assesses the quantity of network modules and the membership within each module.

In line with the above state-of-the-art, the approach presented in Chapter 6 starts by constructing a graph of known connections between actors. It then applies a community detection method followed by an anomaly detection method to identify which of the detected communities are suspicious. In this respect, the outlined approach belongs to the category of approaches that combine community detection with unsupervised learning methods. The rationale for using unsupervised learning methods is twofold: the amount of reliably labelled data available is small relative to the entire graph, and actors who engage in financial behavior continuously adapt and change their behavior, in a way that renders the available labelled data outdated after some period of time.

One crucial aspect of illicit group behavior detection frameworks is their scalability, i.e., the ability to be adapted to larger networks. Certain research studies [RS18; MDR15; Li+20; CT22; Lin+16; ABB20; Mag+18] are directly addressing scalability as one of the key requirements for proposed frameworks. Usually, scalability is addressed by applying distributed computation frameworks such as Apache Spark or by developing algorithms in which computational complexity is linear with respect to the network size [Lin+16].

Another aspect of illicit group behavior detection frameworks that must be taken into account is actionability. A framework can be called actionable if its output (detected potentially illicit groups of actors and their relations) may help a domain expert to isolate the possible source(s) of illicit behavior. Actionability was framed as a requirement in multiple works and addressed by providing pruned networks [Sol+16], sparse networks [MDR15], or networks where missed links were predicted and added [Bah+18].

3.4. Summary

Every research paper that addresses the MoL detection problem mentioned in the state-of-the-art chapter is summarized in Table 1. The papers are compared based on whether they are addressing business requirements to the solution that were described in Chapter 1: accuracy, robustness, timeliness, actionability, interpretability, scalability.

Essentially, every research work keeps as a goal to make the developed solution as accurate as possible, so every paper addresses the *accuracy* requirement. Papers that address MoL detection using unsupervised techniques are classified as *robust* since provided solutions theoretically should detect both known and unknown MoL patterns. Papers that focus on group MoL patterns detection were also considered to be addressing *robustness* requirement.

No papers directly address the requirement of alerts *timeliness*, i.e. how to identify the most suitable time for an alert generation, but the research works that target MoL patterns detection on the transaction level inherently provide timely solutions. Similar logic applies to the *actionability* requirement - very few papers directly address the problem of alert actionability. But the research works that target MoL patterns detection on the transaction level are inherently providing solutions for actionable alert generation since one alert represents only one transaction.

As mentioned earlier, there are no publicly available research works that present a design for local *interpretability* of ML-based MoL detection models, meaning interpretability of an ML model outcomes (i.e. alerts) rather than interpretability of an ML model itself. *Scalability* requirement was addressed in several research works, mostly in the research works that target group MoL behavior detection since the scalability problem is more prominent for the graph-structured data and correspondent algorithms.

Given the above considerations and limitations seen in the state-of-the-art, we can refine the RQs stated in Chapter 1 of the thesis:

- **Refined RQ1.** How to design an ML-based system for MoL detection that does not require computationally expensive model training/execution and that can handle populations with extreme class imbalance?
- **Refined RQ2.** How to measure the quality of a MoL monitoring system over time and in a way that links the outputs of the monitoring system to decisions and actions that its users need to take?
- **Refined RQ3.** How to cater for practical imperatives when designing the MoL monitoring systems with respect to interpretability of its outputs by AML investigators?
- **Refined RQ4.** How to make the designed system resistant towards different classes of illicit behavior, particularly individual vs group MoL behavior?

Table 1. Comparison of the 38 research works addressing MoL patterns detection from the perspective of the non-functional requirements stated in state-of-the-art Chapter

Paper	Year	Accurate	Robust	Timely	Actionable	Interpretable	Scalable
Jullum et al. [Jul+20]	2020	+	-	+	+	-	-
Ahmed et al. [Ahm21]	2021	+	-	+	+	-	-
Raiter et al. [Rai21]	2021	+	-	+	+	-	-
Alkhalili et al. [AQA21]	2021	+	-	+	+	-	-
Lokanan et al. [Lok22]	2022	+	-	+	+	-	-
Harris et al. [Har+22]	2022	+	-	-	+	-	-
Jensen et al. [JI23]	2023	+	-	+	+	-	-
Mubalaike et al. [MA18]	2018	+	+	+	+	-	-
Chen et al. [Che+21]	2021	+	+	+	+	-	-
Lopez et al. [LA12]	2012	+	-	+	+	-	-
Alexandre et al. [AB16]	2016	+	+	-	+	-	+
Eddin et al. [Edd+21]	2021	+	-	+	+	-	-
Pettersson et al. [PA22]	2022	+	-	+	+	-	-
Tai et al. [TK19]	2019	+	-	-	+	-	-
Alarab et al. [APN20]	2020	+	-	+	+	-	-
Alotibi et al. [Alo+22]	2022	+	-	+	+	-	-
Aziz et al. [Azi+22]	2022	+	-	+	+	-	-
Robinson et al. [RS18]	2018	+	-	-	-	-	+
Savage et al. [Sav+17]	2017	+	-	-	+	-	-
Bahulkar et al. [Bah+18]	2018	+	-	-	+	-	-
Baltoiu et al. [BPI19]	2019	+	+	-	-	-	-
Li et al. [Li+20]	2020	+	+	-	-	-	+
Soltani et al. [Sol+16]	2016	+	+	-	+	-	-
Shaikh et al. [SN18]	2018	+	-	-	-	-	-
Mahootiha et al. [MGS21]	2021	+	-	-	-	-	-
Starnini et al. [Sta+21]	2021	+	-	-	+	-	+
Liu et al. [Liu+22]	2022	+	-	-	-	-	-
Lebichot et al. [Leb+17]	2017	+	-	-	-	-	-
Li et al. [Li+21]	2021	+	-	-	-	-	-
Magomedov et al. [Mag+18]	2018	+	+	-	+	-	+
Chen et al. [CT22]	2022	+	+	-	-	-	+
Prado-Romero et al. [PDG18]	2018	+	+	-	-	-	-
Xueshuo et al. [Xue+21]	2021	+	-	-	-	-	-
Sangkaran et al. [SAJ20]	2020	+	-	-	-	-	-
Magalingam et al. [MDR15]	2015	+	-	-	+	-	+
Ling et al. [Lin+16]	2016	+	-	-	-	-	+
Ketenci et al. [Ket+21]	2021	+	-	-	-	-	-
Chao et al. [Cha+19]	2019	+	-	-	-	-	-

4. INDIVIDUAL MOL PATTERNS DETECTION

In this chapter, we address the **RQ1**: *"How to design an ML-based system for MoL detection that does not require computationally expensive model training/execution and that can handle populations with extreme class imbalance?"* and specifically, focus on the problem of MoL detection on the individual level using ML techniques. We describe the applied problems that essentially arise in the domain of automated MoL detection, namely high amounts of data to be processed and extremely low number of positive cases to train the model on. And we propose the design of the solution to the aforementioned problems that has been tested in a real-life setup.

The development of ML models for MoL detection raises a tension between scalability and accuracy. AML monitoring approaches based on LR or DT cannot effectively capture the complex patterns typically found in MoL, while approaches based on SVM or NN are computationally expensive. Tradeoff solutions are tree-based techniques such as RF and GBT which have been shown to achieve high levels of accuracy for a wide range of classification tasks in various domains while providing suitable levels of computational efficiency. However, the computational efficiency of such methods highly depends on the complexity of calculating the feature set. In this setting, the question of how to train accurate ML models for AML, while fulfilling two requirements: scalability and imbalance-resistance is addressed. By scalability, we meant the ability to train the models to very large transaction datasets, in such a way that applying the model periodically with relatively standard computing resources is feasible. This requirement precludes approaches that entail generating sophisticated feature sets for every customer or training deep learning models on the entire dataset. By imbalance-resistance, we refer to the ability of the model to achieve suitable accuracy despite high class imbalance, i.e. the low number of instances of potentially illicit behavior relative to a large number of features that may characterize potentially illicit behavior. To improve imbalance-resistance, ML models use well-known approaches to handle class imbalance via under-sampling or over-sampling. However, in our context, this random sampling approach does not exploit the fact that a very large proportion of customers have transactional patterns that make them unlikely to engage in illicit behavior.

To address the above requirements, a two-layered architecture for training ML classifiers for detecting potentially illicit financial behavior is proposed. In this architecture, two classifiers are applied sequentially to the input samples. The first classifier relies on a highly efficient LR classification technique coupled with a small number of simple features capturing the customer profile and aggregate transaction volumes. This layer is intended to filter out clearly non-illicit customers. The second layer then uses a larger and more sophisticated set of features and a more complex XGBoost classifier in order to classify heightened-risk customers into potentially illicit and non-illicit. The proposed framework incorporates a range of approaches for feature extraction, including mean-encoded categorical features, statistics based on an aggregation of time series data, customer ego-network statistics, and features extracted from stochastic models.

In this chapter, a study where the above problem is tackled in the context of a multinational financial institution which is operating over three separate jurisdictions is presented. The proposed framework is evaluated on a real-life large-scale dataset consisting of customer profiles and transaction histories, together with labels provided by AML experts within a universal financial institution.

The rest of the chapter is organized as follows. Section 4.1 depicts the data used for model development. Section 4.2 describes the methodology including layered model definition, feature extraction and tackling the dataset imbalance. Section 4.3 shows the experimental setup we designed. Section 4.4 shows the results of the experiments. Section 4.5 opens the discussion of ML model limitations, sketches the directions of future research and summarizes the main conclusions and contributions.

4.1. Data

In the following, the labels used for supervised learning are defined and a sketch of the database structure is provided.

As labels, we use information about instances of customer activities that AML experts have previously deemed to as having reasonable grounds for reporting to the local FIU. In the research, two labels are used: customers who have been previously reported to the FIU and those who have not. For the latter a random sample from the customer base is used, thus in the following, they are called *randomly sampled* customers. In this research, the randomly sampled customers were used as a negative class and reported customers as a positive class.

Essentially, the problem at hand can be classified as a Positive and Unlabelled (PU) learning problem. In the PU learning problem, there is a small subset of data points labelled to belong to the positive class while the rest of the dataset remains unlabelled and considered to belong to the negative class. One of the approaches to tackle the PU learning problem is a two-step algorithm [JS19] where in the first step the ”non-reliable” negative examples are filtered out and in the second step, the ”reliable” negative examples and the positive examples are used to train the ML classification model.

Fig. 7 gives a sketch of the database structure, which is used for the present research. It consists of four tables: customers’ identifiers table, demographic data table, transactions data table, and reports table. In the demographic data table, there is static information which describes customers, while in the transactions data table, there are transactions of those customers. A transaction has several properties like a direction – whether the transaction is incoming or outgoing, activity type – whether the transaction is foreign or intrabank, etc., channel – whether the transaction was made through physical or electronic channels, etc., and others

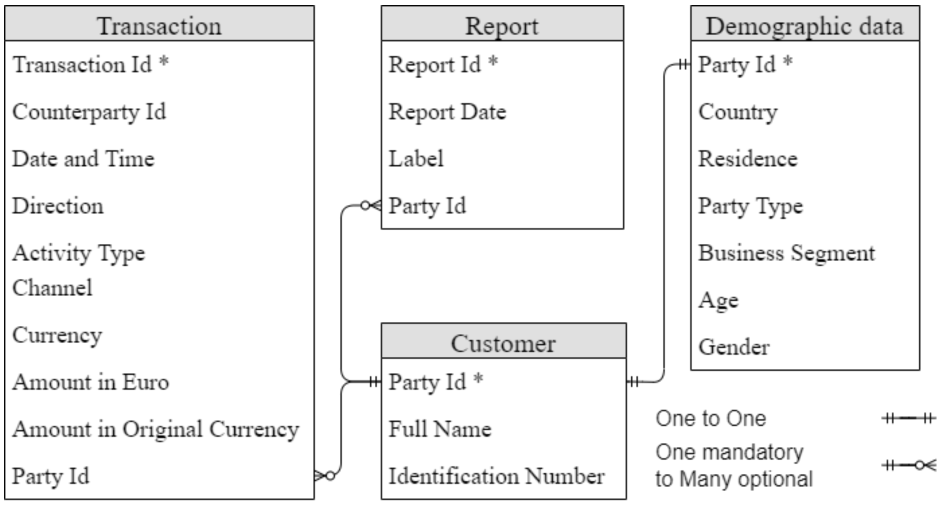


Figure 7. The sketch of a database structure

which are self-explanatory by their names. The report's table holds information about reports on those customers who are deemed to be involved in MoL activity.

Obviously, potentially illicit activity is uncommon, thus we have high class imbalance in the present research. The detailed explanations of how the class imbalance was tackled can be found in Section 4.2.5. MoL is a complex activity which very often spans over a long period of financial activity. Thus, it was decided to approach the problem on a long-term customer activity level, not on a single transaction level. Accordingly, the labels used are on a customer activity period level, not on a transaction level. Note, that for banks from different countries, the criteria for reporting to the FIU could be different due to differences in legislation. The possible implications from this are discussed in Section 4.4.2.

4.2. Approach

This section describes in detail the framework developed to detect potentially illicit financial behavior. First, the framework with an explanation of its architecture and the reasons which lead to it is defined. Then, the types of features generated based on raw demographical and transactional customers' data together with the reasoning for generating each type of those features is described. In the end, tackling the imbalance of data is illustrated.

4.2.1. Two-layered Model Concept

Obviously, the overwhelming majority of customers are clearly non-illicit, and only a small portion of the customer base is worth to be investigated. If we filter out such clearly non-illicit customers, we can significantly narrow down the problem. From another point of view, such filtering solves some part of the computational cost problem, since it reduces the number of customers who has to go

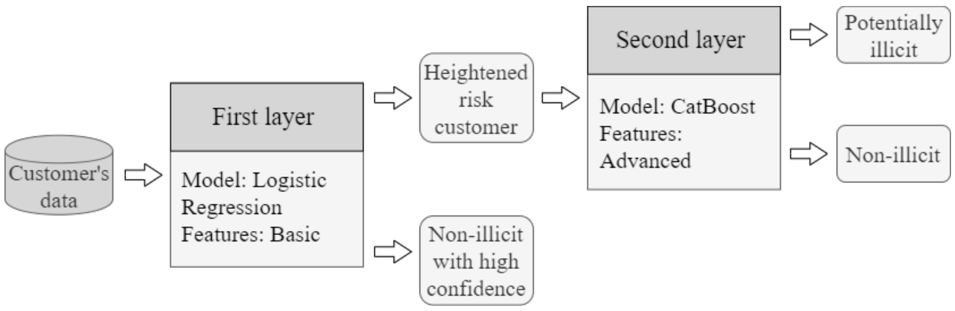


Figure 8. Classification model architecture

through deeper analysis. For these purposes, we introduce a two-layered model concept, that is inspired by the PU learning approach tackling techniques. The first layer model is fast and simple but less accurate. It is focused on dropping "obviously non-illicit" customers. In turn, the second layer is using a more robust classifier trained on customers represented with a larger set of complex features. The aim of the second layer model is to distinguish customers who are worth to be checked by AML experts among all heightened risk customers. In Fig. 8 shows the architecture of the final classification model.

As a first layer, we use a linear model trained on a set of simple features: standard deviation, mean, and count of transactions; business segment type, customer type; simple statistic of an egocentric network of each customer. These features are fast to compute, and they clearly describe customers with a small turnover. As a first layer classifier, LR is used because of its simplicity and speed. As the second layer classifier, XGBoost model is used, specifically the CatBoost implementation of this algorithm [Pro+18], which is tailored to handle mixtures of numerical and categorical features.[1] XGBoost algorithm was chosen over deep learning (DL) classification algorithms, which are widely considered to be more efficient, because of several reasons. First, DL techniques are hardly interpretable which is one of the key requirements for MoL detection solutions. Second, the usage of DL techniques on large data requires special GPU-based infrastructure which in the context of the financial institution was unavailable.

4.2.2. Framework

All used techniques are combined into a monitoring framework (see Fig. 9). As input data, *K* months snapshots of customers' activity was used. Then for each of the layers, we extract own feature sets. Feature selection is not performed for the first layer. Parameters of the first layer model are tuned by a grid search of parameters. For the second layer, we perform feature selection and use Bayesian Optimization for hyperparameter tuning. For the first layer, the fast LR model is

[1]We also conducted experiments using another implementation of XGBoost, but these experiments consistently led to lower accuracy. In the evaluation reported below, we only report on results obtained using CatBoost.

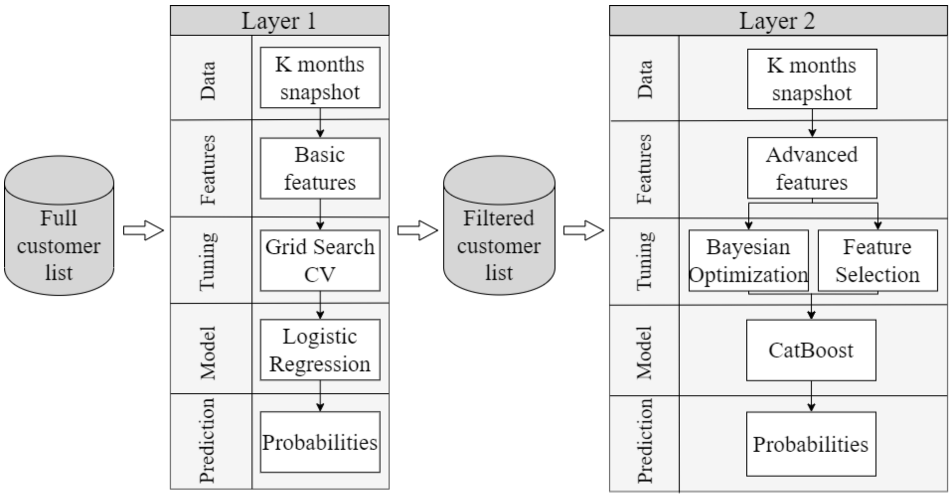


Figure 9. Proposed framework for AML monitoring

used and for the second layer, the CatBoost model is used. Customers for whom the first-layer model produces a class probability above a predefined threshold, form the customer base for the second layer (heightened-risk customers). This first-layer threshold is tuned to minimize false negatives as explained later (see Section 4.3). The second layer takes as input *K* months snapshots of heightened risk customers. Alerts are generated about customers for whom the second layer produces a class probability above a threshold, which is determined by the AML domain experts.

4.2.3. Feature Extraction

The approach is focused on customer-level prediction while having two sources of raw data: demographical and transactional. Demographical data is relatively static and customer-based, therefore, it does not require complex feature engineering. The main issue arises in utilizing dynamic time-series data of customers' transactions. Below, you may find a table (see Table 2) with a description of the different types of features generated for the classifier, the reasoning behind them and correspondent examples. All features were calculated by standard group-by-apply operations, except for sequence-based features. For them, the more detailed calculation is provided separately in Section 4.2.4.

Eventually, there are more than 400 features for the second layer, and it is very computationally costly to extract all of them, thus a robust feature selection is needed. All features are firstly filtered through high correlation and low variance filters. After, a BoostARoota algorithm - an extension of Boruta algorithm [KJR10] for all relevant feature selection is applied. In this feature selection technique importance of each feature is compared to the importance of its "shuffled version". If a feature is more important than random, with a predefined confidence level, then it is kept, otherwise not. We created an automated features

Table 2. Features types with their reasoning and examples

Name	**Definition**	**Reasoning**	**Example**
Demographical Features	Static demographical characteristics of customers.	Potentially illicit patterns vary significantly depending on the type of a party (business or private), country, business segment, etc.	Business segment, country of residence, standard industry classification code.
Simple Descriptive Statistics	Descriptive statistics of transactions amount (min, max, mean, std, count, sum, median) aggregated by various categorical columns: by direction, by channel or through the whole data range.	Based on an exploratory analysis of the data, we discovered that even simple statistical measures have different distributions for potentially illicit and non-illicit customers.	Mean amount of the inbound transactions a customer made during the last month.
Transaction Time Difference Features	Statistics that are calculated based on the difference between timestamps of two consecutive transactions.	An observation was that in some cases, potentially illicit customers made very frequent transactions over a small period of time.	Min, max, mean time between consecutive transactions.
Transaction Amount Difference Features	Differences between sums and counts of aggregations for different time periods.	Potentially illicit behavior may drastically change over some periods of time. For example, in one month criminals might transact very frequently and with large volumes of money, but after that, there can be quite a long period of lull.	The difference between the sum of incoming transactions for September and October, the difference between the sum of incoming versus outgoing transactions for August.
Counterparty Features	Statistics based on information from counterparties for those customers, who have at least one transaction with a defined counterparty.	Since MoL often is a group activity we need to have some characteristics of customers' counterparties.	Ratios between unique counterparty types and amount of transactions, the number of unique counterparty types (i.e. accounts, names, etc.).
Proportion Features	Proportions of channels and activity types, aggregated by time and direction.	The distribution of customers' transactions may be different for potentially illicit than for non-illicit customers, we assume that they can have some abnormalities in transactions type and channels.	Proportion of Payments in October from all transactions in October, a proportion of cash withdrawals in June from all transactions in June.
Sequence-based Features	Generative log-odds features [Leo+13], where we estimate transition probabilities between each transaction state separately for potentially illicit and non-illicit customers and then compare them. The feature is calculated as $$\log \frac{P(Y = potentially\_illicit \mid x_1, \ldots, x_n)}{P(Y = non\_illicit \mid x_1, \ldots, x_n)}$$ where $x_1, x_2, \ldots, x_n$ are some discrete properties of transactions (e.g. direction or used channel). Probabilities in the numerator and denominator are calculated using Bayes theorem and probabilities of sequences estimated with the chain rule and simplified assumptions of Markov property.	Sequences of customers' transactions are not random and should follow some hidden structure.	Log-odds feature where the sequence is direction, channel, activity type, and discretized transaction amount.

selection method that allows us to automatically choose the most relevant features and reduce overfitting. The features are selected on a validation set with respect to the best Precision Recall Area Under Curve (PRAUC). By applying BoostARoota algorithm we reduced approximately 25% of features and gained marginal improvement in PRAUC. At the same time, by reducing the number of features, CatBoost model is trained faster, allowing us to utilize deeper trees within a reasonable time.

4.2.4. Sequence-based features

There is an assumption that sequences of customers' transactions are not random and follow some hidden structure. Therefore, we used a way to encode this information to the model by so-called generative log-odds features [Leo+13], where transaction probabilities between each transaction state separately for potentially illicit and non-illicit customers are estimated and then compared. This approach allows us to capture the dynamics of the transaction history for our classification task while introducing less overhead than methods based on neural networks (e.g. Boltzman machines) or autoencoders. In the log-odd feature extraction method, we want to generate features based on sequential probabilities. We are interested in the following probability of a sequence of transactions coming from a potentially illicit customer:

$$P(X) = P(x_1, x_2, \ldots, x_n) \tag{4.1}$$

where $x_1, x_2, \ldots, x_n$ are some discrete properties of transactions (e.g. direction or used channel). One particular way to estimate this probability is to use the chain rule:

$$P(X) = P(x_1, \ldots, x_n) = p(x_1)p(x_2 \mid x_1) \ldots p(x_n \mid x_1, \ldots, x_{n-1}) \tag{4.2}$$

In some cases, it is practically impossible, so the assumptions using Markov property can be simplified:

$$P(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = P(X_n = x_n \mid X_{n-1} = x_{n-1}) \tag{4.3}$$

This way, formula 4.2 turns into:

$$P(X) = P(x_1, \ldots, x_n) = p(x_1)p(x_2 \mid x_1)(x_3 \mid x_2) \ldots p(x_n \mid x_{n-1}) \tag{4.4}$$

But for our task, we are more interested in finding that a particular set of transactions is more illicit than a set of randomly taken non-illicit transactions. Mathematically, we want to estimate:

$$\underset{y \in (potentially\_illicit; non\_illicit)}{\operatorname{argmax}} \quad P(Y = y|X) \tag{4.5}$$

One way to calculate this probability is to use Bayes theorem:

$$\underset{y}{\operatorname{argmax}} P(Y = y \mid X) = \underset{y}{\operatorname{argmax}} P(X \mid Y = y)P(Y = y) \tag{4.6}$$

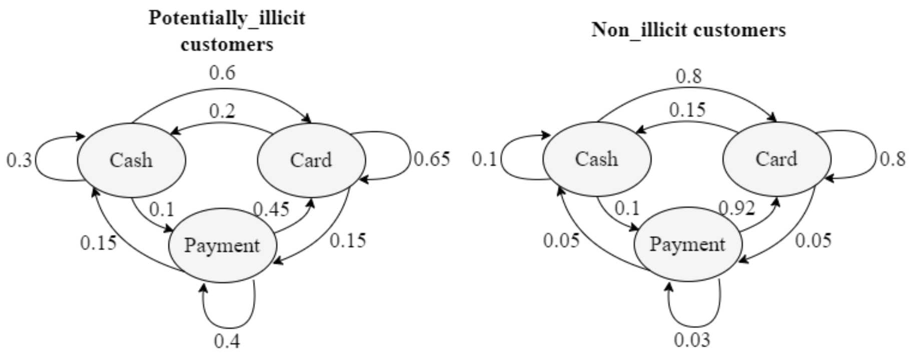


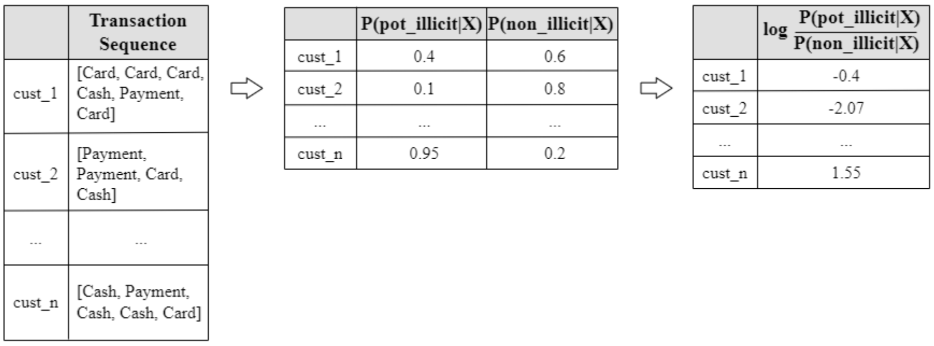
Figure 10. Example of transition probabilities for the transaction type property that has three states: cash, card, payment

The only thing left is to calculate $P(X \mid Y = y)$ and $P(Y = y)$. $P(X \mid Y = y)$ can be calculated using the train set and then calculating transition probabilities separately for potentially illicit class and non-illicit class. $P(Y = y)$ is the prior probability of being potentially illicit, which is simply a proportion of potentially illicit customers in a full customer set for train data.

Finally, instead of outputting a binary label 1/0 (potentially illicit sequence or not), we can plug this as a feature into a classifier along with other features. We can use the so-called log-odds ratio instead of a binary feature, defining it as:

$$\log \frac{P(Y = potentially\_illicit \mid X)}{P(Y = non\_illicit \mid X)} = \log \frac{P(X \mid Y = potentially\_illicit)P(Y = potentially\_illicit)}{P(X \mid Y = non\_illicit)P(Y = non\_illicit)} \quad (4.7)$$

In practice, the calculation of sequence-based features looks as follows. First, on the holdout set that is not used for the general model training, the transition probabilities for the transitions between discrete states of transactions is calculated. As an example, for the case of transaction type which has three states (cash, card, payment) we will get the following transition probabilities (see Fig. 10). Transition probabilities for a discrete state are calculated separately for the cases of potentially illicit customers and non-illicit customers. For each customer, we take a sequence of discrete states and calculated estimated probabilities as stated in formula 4.4 for a distribution of potentially illicit customers and the distribution of non-illicit customers. Finally, the log of odds that such sequence coming from the distribution of potentially illicit customers to the distribution of non-illicit customers is calculated using formula 4.7 and schematically depicted in Fig. 11. The log odds of probabilities for such properties of a transaction as direction, transaction type, channel, and binned transaction amount are calculated and used as separate features for the second layer classification model.



	Transaction Sequence
cust_1	[Card, Card, Card, Cash, Payment, Card]
cust_2	[Payment, Payment, Card, Cash]
...	...
cust_n	[Cash, Payment, Cash, Cash, Card]

	P(pot_illicit|X)	P(non_illicit|X)
cust_1	0.4	0.6
cust_2	0.1	0.8
...	...	...
cust_n	0.95	0.2

	$\log \frac{P(pot\_illicit \mid X)}{P(non\_illicit \mid X)}$
cust_1	-0.4
cust_2	-2.07
...	...
cust_n	1.55

Figure 11. Schematic example of sequence-based features calculation for the *transaction type* discrete property

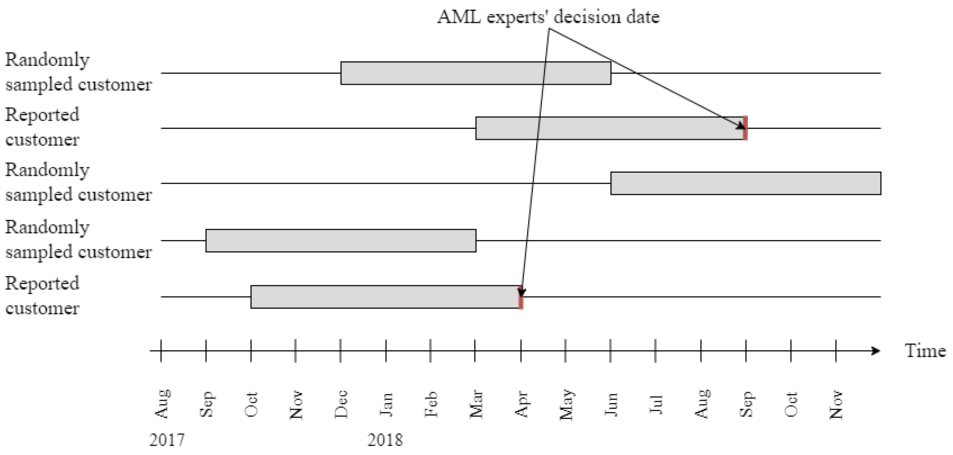
4.2.5. Tackling Class Imbalance

As it was described earlier, we are dealing with a highly imbalanced dataset. In order to build a meaningful model and correctly estimate its performance, we need to pay attention to the class ratio. In the approach, two common ways of dealing with imbalanced datasets are used: cost-sensitive learning and sampling techniques. Later the metrics used to estimate model performance are discussed.

Cost-sensitive learning. Cost-sensitive learning is a common approach to deal with class imbalance by penalizing a model more for incorrectly predicting the minority class versus the majority class. In this approach, weights for each class in a cost function are added, therefore, the more weight a particular class has, the higher loss will be given for an incorrectly classified example. In the study, we wish to penalize the model more for incorrect prediction of the reported case, rather than for the randomly sampled case.

Combination of Undersampling and Oversampling. There exist different ways of dataset sampling: undersampling of a majority class, oversampling of a minority class and their hybrid approaches. In all cases, the ratio of classes is artificially changed, targeting less imbalanced class ratios. Several approaches are tried, including Synthetic Minority Over-sampling Technique (SMOTE) [Cha+02] and Edited Nearest Neighbors (ENN) undersampling [Wil72].

The experiments showed that none of the undersampling, oversampling or mixture of them improves the performance of the framework. Accordingly, the final version of the framework was trained without changing the class ratio. Nevertheless, due to the architectural decision of the classification model, the first layer of the model serves as the knowledge-driven undersampling for the second layer. It filters out clearly non-illicit customers, and the experiments show that the performance increases compared with the model trained without this filtering.



AML experts' decision date
Randomly sampled customer
Reported customer
Randomly sampled customer
Randomly sampled customer
Reported customer
Time
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
2017
2018

Figure 12. Example of snapshots extraction

4.3. Experimental Setup

In the experiments, a subset of reported and a subset of randomly sampled customers is used. Total bank's database is in order of magnitude of millions, but for experimental purposes, $\sim$330000 customers from 3 countries with more than 51 million transactions are used. The percentage-wise distribution is 0.4% of reported and 99.6% of randomly sampled customers; 8% of corporate and 92% of private customers. All customers have a transaction history from August 2017 to December 2018.

As samples for classification, we take snapshots of customers at a given period of time, and for each customer, we take K months history. In this study, K is set to be equal to 6. For the reported customers – K months prior to the date of AML experts' decision, for randomly sampled customers – K random consequent months. For clarity, the schematic explanation of snapshot extraction is presented in Fig. 12.

The training schema for the framework is inspired by the training schema of stacked models, but in our case, the second layer model doesn't use the output of the first layer directly as features. Instead, it uses a richer feature set than the first layer and it is trained and tested only on prefiltered customers. Data split on train and test is stratified by country, customer type (business or private) and label.

Since the dataset is highly imbalanced, the PRAUC as a measure of accuracy is used. Both model layers are trained to maximize PRAUC separately because they have different thresholds, purposes, and constraints. The purpose of the first layer is to filter out clearly non-illicit customers and at the same time not miss potentially illicit ones, thus, the probability threshold is set in such a way that the first layer does not miss more than 1% of potentially illicit customers on a validation set. In turn, the probability threshold for the second layer could be tuned by AML or compliance officers in a risk-based approach in accordance with their institute's risk appetite. To illustrate the options for risk-based tuning we also use precision and recall at fixed thresholds. Both of those metrics are crucial since precision defines how many alerts made are correct and recall defines what is the proportion of potentially illicit customers for whom alerts are generated.

4.4. Results

As stated in the introduction of Chapter 4, it is anticipated that the proposed two-layered model architecture outperforms a single-layered one. To validate it, an experiment where the performance of the two-layered model was compared against baseline RF and CatBoost models was conducted. Also, the execution time of full model training and application pipelines for all three models was measured.

4.4.1. First Layer Results

Firstly, the first layer model is applied to the full customer set for filtering. The decision threshold is chosen in such a way that with this threshold we filter out no more than 1% of reported customers on the validation set. Thereby, on a test set, we filter out 48% of randomly sampled customers and miss 6.6% of the reported customers. Below is the confusion matrix for the first layer (see Table 3).

Table 3. Confusion matrix for the first layer of the two-layer model

		Predicted	
		randomly sampled	reported
Actual	randomly sampled	48.07%	51.93%
	reported	6.6%	93.4%

4.4.2. Second Layer Results

Table 4 shows the observed performance for the second layer of the two-layer model for each of the countries. As can be seen, the results are different from country to country. The classifier has the best performance for Country 1. One of the potential reasons is that the labels for three different countries are created by different groups of AML experts, and it is possible that the criteria used by them to report customers in Country 1 are better recognized by the features than in Country 2 and Country 3. Another potential reason is the dissimilarity in legislation. Banks from different countries have different reporting requirements and guidelines from respective authorities. Accordingly, they might have different sets of rules by which they raise alerts. Some of those rules are unique and ad hoc, so it creates differences in the distributions of labels for different countries. For example, in some countries, cash-related business is less regulated than in others, which may make it so that cash-related features in an ML model would be less important in one country than another. The definitions of MoL risk in different countries may differ, for example, trading in several currencies may be considered to be suspicious by authorities in one country but not in others. Again, this has an impact both on the importance of multi-currency features, as well as on the distribution of suspicious behavior across multiple countries.

Table 4. Performance of the second layer on three countries separately

	Threshold 0.25		**Threshold 0.5**		**Threshold 0.75**		
Metric	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**PRAUC**
Country 1	0.859	0.321	0.703	0.576	0.546	0.813	0.732
Country 2	0.627	0.412	0.450	0.575	0.267	0.759	0.517
Country 3	0.582	0.291	0.373	0.472	0.208	0.730	0.422

We also observed an even stronger decrease in performance for all three countries when different models were trained for each country separately. This proves that fraudsters in all countries share some patterns, and it is crucial to train one model for the whole dataset, even though the way customers are labelled is slightly different for the countries.

4.4.3. Overall Results

Below the overall performance of the two-layered model is presented. From the first layer, we take customers who have been classified as "non-illicit with high confidence" (TN and FN from the confusion matrix in Table 3) and add those to "heightened risk" customers classified by the second layer. To get the final confusion matrices, we have to sum up TNs and FNs from both layers and take TPs and FPs only from the second layer.

To measure the performance of the two-layered model, we compared it to baseline models – RF model and CatBoost model which were trained and tested on the not filtered customer base. As features, we used the complex feature set which in the two-layer model we used only for the second layer. The results for all countries together can be found in Table 5.

Table 5. Performance on three countries together

	Threshold 0.25		**Threshold 0.5**		**Threshold 0.75**		
Metric	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**PRAUC**
Two-layer model	0.618	0.348	0.448	0.548	0.287	0.772	0.497
CatBoost model	0.615	0.298	0.430	0.531	0.209	0.775	0.467
RF model	0.615	0.218	0.439	0.447	0.3	0.622	0.428

The execution time of major stages of model training was also measured. The execution times are shown in Table 6. We note that the two-layered model is faster to train than the single-layer RF and CatBoost models since the two-layered architecture allows us not to extract complex features for all customers, which is the most time-consuming stage. Also, one can notice that the CatBoost model is slightly slower to train than the RF model.

It can be observed that the two-layered model outperforms the CatBoost model and RF model by both the time of training and application as well as by the performance. Also, it can be noticed that the RF model is beaten by the CatBoost model in performance.

All experiments were run on a single PC with Intel Core i5-6300U CPU (x64-based 2.40GHz processor), 64GB RAM, with a 64-bit Operating System.

Table 6. Execution time by stages

	Train	**Application**
Number of samples	$\sim$230k	$\sim$100k
Stages included	Samples feature extraction + model(s) training	Samples feature extraction + model(s) application
Two-layer model time	89 min	42 min
CatBoost model time	148 min	66 min
RF model time	143 min	65 min

4.5. Summary

In this chapter, we addressed **RQ1** posed in the thesis. We studied the problem of MoL detection on the individual (customer-centric) level and technical issues associated with it - severe dominance of non-illicit behavior compared to potentially illicit one; and huge data amounts to be monitored. Proposed solutions to those issues were evaluated on real-life data.

We presented and evaluated a two-layered approach to train ML models for detecting potentially illicit behavior in the context of AML monitoring. The key idea developed is that instead of training a single classifier using an extensive set of features, we can train a first simple model to discard "clearly non-illicit customers" and pipeline it with a second model that uses a more sophisticated feature set and classifier learning method. We instantiated this general two-layered concept using LR for the first layer, and extreme gradient boosting trees CatBoost methods for the second layer. The first layer relies on "static" customer-level features and transaction volume aggregates, while the second layer combines several feature sets, ranging from aggregates for different time windows and attributes to Markov Chain-based stochastic models to capture the temporal dynamics of transactions. We advocated that this architecture allows us to achieve better scalability while allowing us to handle the high class imbalance typically found in the field of AML.

From the experiments we made, we found out that the two-layered approach outperforms a single-layered approach based on a single classifier (CatBoost and RF) both in terms of accuracy and execution time for model training and application. Also, we showed that the CatBoost model outperforms the RF model in terms of accuracy. It was also discovered that training a single model to make predictions across multiple countries leads to higher accuracy than training separate models for each country.

A fundamental limitation of the proposed approach is that it is a purely supervised learning solution, which means that the framework is able to detect only those illicit behavior patterns that have been previously identified by the AML experts who provide the labels. In the reported experiments, we observed that the information encoded in the labelled data could bias the model significantly. A possible direction to enhance the proposed approach would be to combine it with anomaly detection methods, so the method will help to identify new potentially illicit behavior together with the behavior that investigators have previously labelled as potentially illicit.

Another limitation of the approach is that it focuses on extracting features from tabular data. In the case when AML experts are not certain regarding the status of a customer, they may ask for additional documents, such as invoices, contracts, purchase orders, etc. Exploiting this information, via text mining techniques, is another possible extension.

Finally, the proposed study focused on calculating the probability that a given customer (at a given point in time) engages in behavior that may be considered to be potentially illicit. We did not address the question of when to trigger an alert (i.e. trigger an investigation) based on the generated predictions. When designing a monitoring system in this context, one needs to take into account the availability of resources to conduct investigations, and the fact that generating alerts earlier is generally preferable than later, but generating an alert too early may lead to there not being sufficient information to conduct an investigation. This question is addressed in the next chapter.

The proposed approach was developed and evaluated on a real-life large-scale dataset which includes customer profiles and transaction histories whereas labels were provided by AML experts within a financial institution.

5. MOL MONITORING SYSTEM

In this chapter, we address **RQ2**: *"How to measure the quality of a MoL monitoring system over time and in a way that links the outputs of the monitoring system to decisions and actions that its users need to take?"*; and **RQ3**: *"How to cater for practical imperatives when designing the MoL monitoring systems with respect to interpretability of its outputs by AML investigators?"*. We present the design and development of an ML-based offline monitoring system for AML. The monitoring system uses the customers' MoL risk scores generated by a MoL classification ML model (designed in Chapter 4) to define when it is the most efficient to fire an alert. The correspondent alerts are then manually checked by the domain experts.

The purpose of the research is to design and assess the performance of the offline MoL monitoring system as well as to make the system usable for the end users. The proposed monitoring system addresses three key requirements: (i) generating accurate and non-redundant alerts; (ii) generating timely alerts; and (iii) associating explanations and risk estimates to each alert. The first requirement is addressed by an ML classification model that was trained on customers' financial behavior history. An alerting policy designed to prevent redundant alerts was built on top of the classification scores generated by this classification model. The second requirement is addressed by the design of the monitoring system as well as custom metrics for assessing the performance of the classification model, which take into account the timeliness requirement and are used for monitoring system tuning. Finally, the third requirement is addressed by an interpretability layer based on Shapley values [Sha53] and by applying a method for class probability calibration.

The proposed offline monitoring system has been designed based on requirements provided by an investigation unit at the financial institution and evaluated using real-life data as well as multiple rounds of feedback from specialized domain experts.

The rest of the chapter is organized as follows. Section 5.1 describes the proposed ML-based monitoring system for AML. Section 5.2 discusses an evaluation of the proposed system undertaken in a real-life environment and summarizes feedback given by subject-matter experts in this setting. Finally, Section 5.3 sketches the directions of future research and summarizes the main conclusions and contributions.

5.1. Approach

In this section, the design of an ML-based monitoring system (MS) is presented [1] that addresses the three requirements spelt out in the introduction to Chapter 5,

[1] in this chapter we use the acronym MS for "monitoring system" since here we discuss the design of one specific monitoring system. Throughout the rest of the thesis, we deliberately have

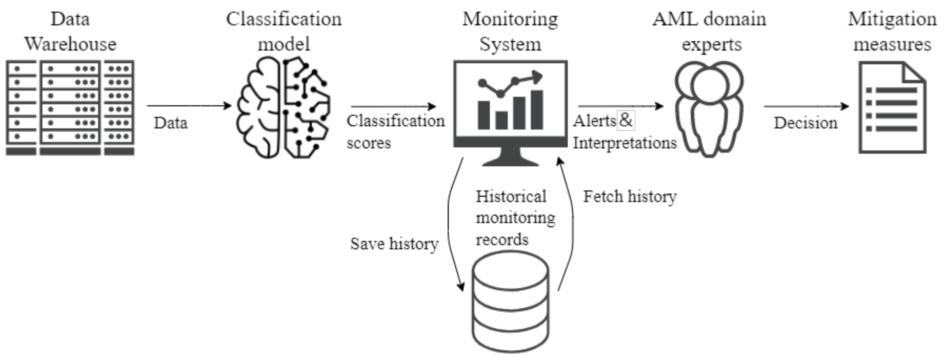


Figure 13. Simplified view of a pipeline for ML-based MoL monitoring system

namely: (i) accuracy and non-redundancy of alerts; (ii) timeliness of alerts; and (iii) interpretability of alerts.

The simplified view of a pipeline for ML-based MS is presented in Fig. 13. Like any other ML-based system, the first component of the system is a data extraction and pre-processing component, which extracts data from a warehouse in order to construct feature matrices that are then used to train a classification model. This classification model produces as output classification scores for every entity at any point in time (e.g. every week). The classification scores are then used by the MS itself, which implements a policy for turning the classification model results into alerts. This policy takes into account previous alerts and classification scores (to prevent redundant alerts) and is calibrated to maximize certain custom classification metrics that take into account the alert timeliness requirement. The MS relies on probability calibration and explainable ML techniques to provide risk scores and explanations to domain experts, together with each generated alert. Based on this input, the domain experts then proceed to investigate each alert and take mitigation measures when required.

The rest of this section presents the design of the MS, the custom metrics to account for alert timeliness, the probability calibration strategy, and the interpretability layer associated with the proposed MS.

5.1.1. Monitoring System

The input of the proposed MS is a dataset of customers and their financial activities in the financial institution (see Subsection 5.2.1 for a detailed description of the dataset). The dataset contains the history of transactions of each customer during a period of time and in this study, 6 months long transaction histories are handled. If a customer joined the financial institution less than 6 months ago, then the customer is still monitored and treated as if it was dormant at the beginning of 6 months period. The dataset includes two groups of customers:

1. Customers who were reported (herein called "reported") to the relevant au-

not replaced "monitoring system" with the acronym because it subjectively complicates reading.

thority, such as FIU, because of a detected potentially illicit activity. Reported customers have at least one report, and each report has a corresponding report date.

2. Those who were never reported. Out of the whole customer base of never-reported customers, a set of customers was sampled, those are called "randomly sampled".

The task is to monitor the financial activity of the bank's customers and to raise alerts if and when potentially illicit activity is detected. A customer can perform multiple potentially illicit actions, therefore, the customer can get multiple alerts. The alerts are generated on the basis of an ML classification model trained on historical data of customers' financial activity and historical labels. The problem in hand was treated as a classification problem, rather than a use case for survival analysis. The main goal, ultimately, is to find if a customer is likely to be found potentially illicit by an investigator at time T, rather than finding out whether the customer will eventually be found potentially illicit. Survival models are suitable when we wish to estimate the time until an event happens, which in this context would mean the time until a customer engages in or is found to engage in illicit behavior. But in this chapter, we focus on generating alerts that investigators would find useful.

In this approach, customers are classified on the basis of a range of profile features as well as their history of transactions during a given period of 6 months. This approach uses two predictive models chained one after the other (the model was described in detail in Section 4.2.2). The first predictive model is an LR model designed to filter out customers who clearly do not exhibit behavior similar to that of reported customers. It is based on a small number of features that can be computed in a scalable manner, which is essential in order to be applicable to millions of customers. The second model is a tree ensemble CatBoost model trained on a larger number of features (see Section 4.2.3) including aggregate features, demographic features, and features extracted from Markov Chain Models trained on a series of inbound and outbound transactions, intended to capture temporal transaction patterns. Such a two-layered architecture of the classification model provides better scalability since the most computationally demanding calculations are done for a filtered set of customers. Additionally, this architecture ensures imbalances-resistance of the model by doing knowledge-driven undersampling for the second layer model, which is performing customer classification. More detailed information on data, features extraction, model selection and comparison can be found in Chapter 4.

The classification model is run periodically – in our case, once per week. In other words, the model is run weekly and after each run, customers are classified as being potentially illicit or non-illicit. In each model run, customers are classified based on their history of transactions of the previous 6 months (herein called a *customer snapshot*, or a *snapshot* for short). This means that in each model

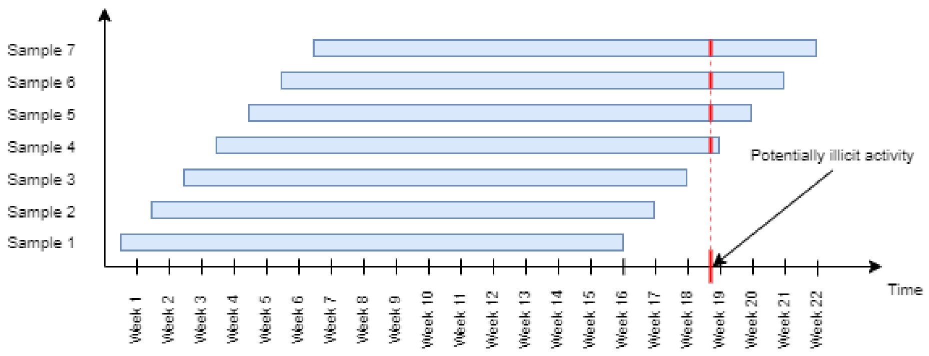


Figure 14. Example of consequent historical snapshots of a single customer which is under monitoring

run, the customer has a week of new transactions while one week of transactions will be forgotten (see Fig. 14). In other words, the classification is made on the basis of sliding windows, to ensure that the samples are comparable. This brings a complication that in sequential monitoring very similar snapshots of the same entity are classified, with one week of new transactions and without one week of old transactions. Eventually, each customer has a sequence of classification scores which represents probabilities of being potentially illicit at each time the classification model is run.

Having a sequence of classification scores for a customer for a period of time, we have to understand when to raise alerts. These alerts will be checked manually by AML domain experts. In light of this, it is impractical to raise alerts for the same customer every week or even several weeks in a row, because it would be likely that the alerts were due to the same set of potentially illicit activities. Accordingly, the custom MS with two parameters is proposed: (i) a base threshold – above which a customer's activity is declared to be unusual enough for an alert to be raised; and (ii) a jump threshold – a threshold of deltas between scores to detect sudden changes in a customer's activity. For an alert to be raised, the corresponding classification score should meet two requirements:

1. The score has to be larger or equal to the base threshold.
2. The difference between the score and the minimum score since the last alert (later *local minimum*) should be larger than the jump threshold.

The rationale behind the first requirement is that alerts should only be raised for customers associated with a high level of risk. The second requirement, in turn, ensures that alerts are only raised when there is a change in a customer's behavior.

In the current research, the choice of running the classification once per week plays more of an illustrative role (similarly to the choice of base threshold and jump threshold in the discussed example). This choice, though, was made by the domain experts given the fact that to make a decision to raise an alert the monitor-

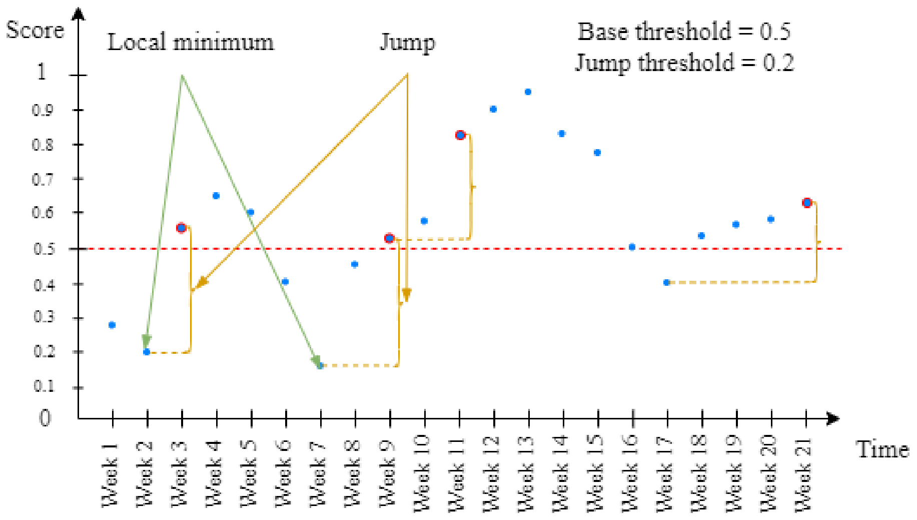


Figure 15. Example of alerts raised for a customer. The customer is monitored for 22 weeks with weekly model runs. In this example base threshold = 0.5 and jump threshold = 0.2

ing system analyses the past 6 months of customer behavior. I.e. classifying more often (every day) does not make much sense since one day of new transactions does not bring a significant amount of new information compared to 6 months worth of transactions. Classifying less often (e.g. every two weeks or every one month) results in a risk of reacting too late – giving the opportunity to money launderers to be intact for too long.

Fig. 15 illustrates an example of a customer under monitoring and the corresponding alerts raised by the proposed MS. In this example, we have a customer's monitoring history of 22 weeks. Each blue dot is the probability of the customer acting illicitly which is computed by the ML classification model. The red circles are weeks when alerts are raised for the customer. The alerts are raised on weeks 3, 9, 11 and 21. For the first two weeks, the customer does not have any alerts. On week 3 the score reaches the value of 0.55 and becomes higher than the base threshold. The local minimum on week 3 equals 0.2, so the difference between the score and the local minimum is higher than the jump threshold, thus, the alert is raised on week 3. Although the scores on weeks 4 and 5 are higher than the base threshold, the scores there do not meet the second requirement and there is almost no jump in score compared with the minimum since the last alert (week 3 for the moment of monitoring at week 4 and 5), so the alerts are not raised on weeks 4 and 5. On week 9 the alert is raised again because the score has reached the level of 0.52 and exceeded the base threshold while the local minimum is at week 7 with the value of 0.18. On week 11 the alert was raised once more because the jump in score was 0.25 which is higher than the jump threshold. In other words, the behavior of the customer has become even more unusual between weeks 10 and 11, probably due to a new unusual activity that happened. For weeks 12 – 16 including, alerts are not raised, again, because there was no jump in the score and the score is high because of the same unusual activity detected on week 11. On week 21 the alert is raised since the difference in scores between the local minimum (week 17) and week 21 got larger than the jump threshold. The score had started to rise gradually since week 17 but reached the threshold on week 21.

5.1.2. Custom Metrics

To assess the quality of an MS, we rely on the notions of recall and precision typically used in the field of classification, with some adjustments to take into account the sequential setting in which the MS operates, as explained below.

Recall. In a one-time AML classification task, recall corresponds to the question "What is the percentage of potentially illicit customers that the AML classification model classifies as such?". In the context of periodic AML monitoring, this question takes a temporal dimension. Concretely, the AML classification model takes as input a snapshot of a customer's data at a particular point in time. The model then produces an output (alert or no-alert) on the basis of this snapshot, and it is later on invoked again with another (later) snapshot. This means that the AML model will classify multiple snapshots of the same customer with a timeshift. For example, if a customer that is related to MoL is monitored for twenty weeks, and the MS is run once a week, twenty samples of this customer will be classified by the model. A naïve application of recall in this setting would not be representative because, if the AML model classifies the last of those twenty samples as potentially illicit (i.e. it raises an alert) and the first 19 as non-illicit (no alert), we get the recall of 1/20. Yet, the model did classify the customer as potentially illicit (i.e. an alert was raised) so arguably the recall, if we restrict the dataset to this one customer, should be 1, provided that the alert is raised at or around the time when the customer was reported as potentially illicit.

In light of the above, we adopt a definition of recall at the customer level (herein called *custom recall*). Specifically, we define custom recall as the number of potentially illicit customers who were alerted at least once at or around the time they were reported as potentially illicit, divided by the total number of potentially illicit customers in the dataset. To make this definition complete, we need to define the notion of "at or around the time". This is defined below as part of the definition of precision.

Precision. In a one-off classification setting, precision answers the question "What is the percentage of customers that the AML model classifies as potentially illicit that are indeed potentially illicit according to the ground truth?". In a periodic monitoring setting, we have to customize this definition to capture, instead, the proportion of correct alerts generated by the AML system. In other words, we need a custom notion of precision that penalizes over-alerting, in particular, repetitive alerts raised in a row for the same customer.

Accordingly, we propose a custom definition of precision that captures the idea that an alert should be raised close to the date when a customer was reported as having illicit activity. Specifically, we propose to capture the utility of an alert based on whether or not it falls under an isosceles trapezoid around a date when a report was made for a customer (see Fig. 16). Very early alerts have the utility in the amount of the left leg of the trapezoid and the utility grows linearly with an alert approaching the report date. Very late alerts have the utility in the amount of the trapezoid's right leg, and the utility decreases linearly starting from one week before the report date and ending at the point of two weeks after the report date. The lower base of the trapezoid defines what is the time range where the alerts get some positive utility; outside this range, the alerts get a utility of zero. The highest utility is achieved at a time between 4 to 1 week before the report date. The rationale for this choice is that earlier alerts have more value than late alerts.

The precision is calculated as

$$precision = \frac{1}{N_A} \sum_{a \in A} \max\left(u_a\right) \tag{5.1}$$

where A is a set of all alerts raised for customers from the dataset, u_a is the utility of alert a calculated as the height of the trapezoid at a time of an alert a, N_A is the size of A. A customer may be reported more than one time, thus there will be a correspondent trapezoid for each of the reports, so we calculate the utility of an alert as $\max\left(u_a\right)$ if it falls under several overlapping trapezoids.

Let's see an example in Fig. 16. The customer in the example has been reported to the FIU on week 8 and got 3 alerts raised by the MS – on weeks 3, 5 and 9 with correspondent utilities of $\frac{2}{3}$, 1 and $\frac{1}{3}$ (correspondent trapezoid heights). The impact to the nominator of the precision (sum of alert utilities) of this customer would be $\frac{2}{3} + 1 + \frac{1}{3} = 2$. And impact to the denominator of the precision is +3, since there were 3 alerts generated. If the customer gets alerts only on weeks 3 and 5 then the impact to the nominator of the precision would be $\frac{2}{3} + 1 = \frac{5}{3}$, but the impact to the denominator of the precision is +2, since there were only 2 alerts generated. That shows that the metric penalizes repetitive alerts raised for the same customer. As can be seen, such a design of precision calculation also controls the timing of alerts by reducing the utility of alerts that are not raised at the correct time. The current way of calculating the precision differentiates the False Positives – a customer who does not have a report but has 1 alert raised will get a 0 total utility and the total precision will be divided by a smaller number (the total number of alerts will be +1). And a customer who does not have a report but has 20 alerts raised, will still get a 0 total utility but the impact on the overall precision will be divided by a larger number (the total number of alerts will be +20).

The custom metrics described above are used to find the optimal parameters of the custom MS. So, the base threshold and jump threshold are taken such that they maximize the custom F1 score on the validation set (see Section 5.2.3). F1

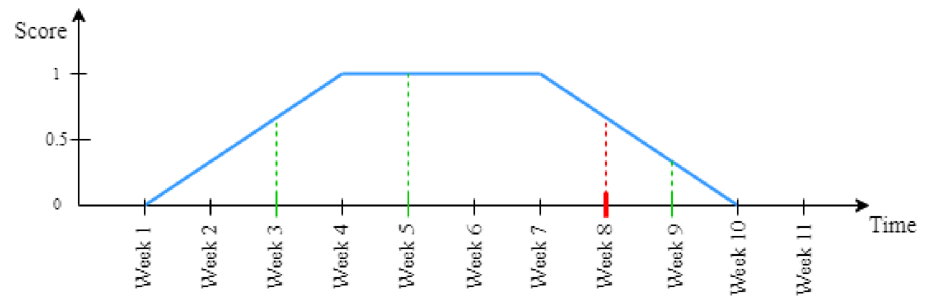


Figure 16. Example of a utility trapezoid for a customer where the customer was reported on week 8 and got alerts on weeks 3, 5 and 9

score is chosen as the optimization metric since it provides a trade-off between precision and recall of the model.

$$custom_{F1} = 2\frac{custom\_precision + custom\_recall}{custom\_precision \cdot custom\_recall} \tag{5.2}$$

5.1.3. Probability calibration

The goal of the proposed AML monitoring approach is to raise alerts when there is a high probability that a customer engages in illicit behavior. To this end, we need to estimate the probability that a given customer-snapshot extracted at a given point in time belongs to the illicit behavior class. Non-linear classification models output class probability scores that do not match the properties of a probability distribution. For example, if the model assigns an output score of 0.8 to a set of samples, one would expect that 80% of these samples would indeed belong to the positive class. However, this is not necessarily the case for non-linear classifiers, because the classification score produced by these models represents a similarity metric and not an actual probability.

In the approach, we require that the scores produced by the classification model should be calibrated, meaning that they are adjusted to reflect an actual probability. By ensuring this, the following benefits are obtained:

1. The model produces scores that match what an end-user would expect, thus allowing the subject-matter experts to follow a risk-based approach when tuning the parameters of the model.
2. The model is robust to periodical retraining because the output probabilities become comparable for models trained in different periods of time.

There are numerous algorithms for probability calibration. Experiments showed that in our task the best approach is beta calibration [KSF17] (see Fig. 17 for comparison of different calibration approaches on the test data). The advantage of beta calibration is that it does not make the assumption that scores of classes in a binary classification task are normally distributed, i.e. have a distribution with infinite support. Instead, it derives a new family of calibration functions

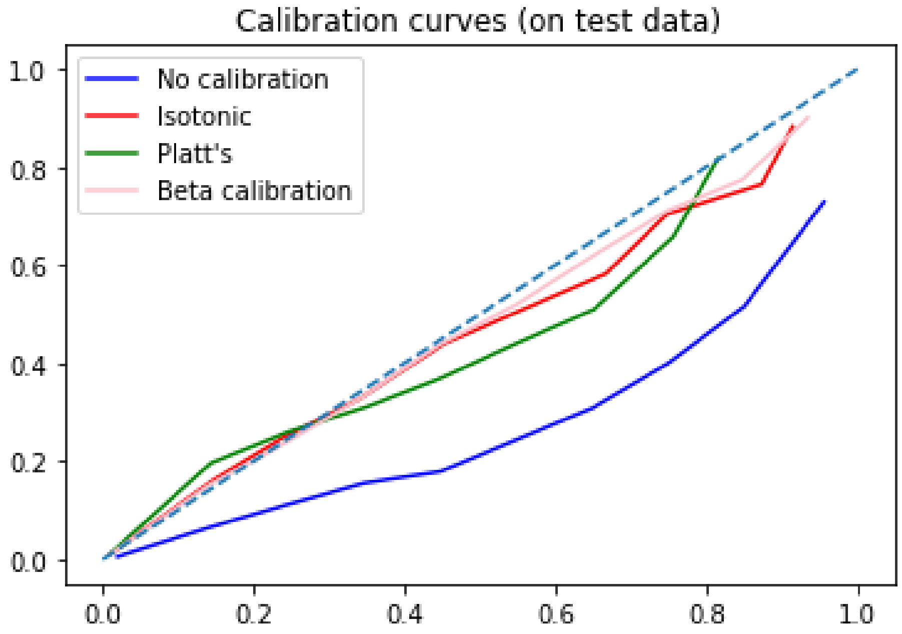


Figure 17. Calibration curves for the model calibrated by beta calibration, isotonic calibration, Platt scaling and without calibration. Dash line is the ideal case

with finite support of [0,1] which are distributed according to beta distribution with

$$p(s, \alpha, \beta) = \frac{s^{\alpha-1}(1-s)^{\beta-1}}{B(\alpha, \beta)} \tag{5.3}$$

5.1.4. Explanation of predictions

The ability to assign interpretations to the predictions generated by an ML model is a common requirement in industrial applications, particularly in situations where the output of an ML model needs to be validated by a domain expert. In the context of an AML MS, the alerts generated on the basis of the ML model are used to trigger investigations. Naturally, the investigators require not only an alert with a corresponding probability score but also clues as to why this alert was produced.

To explain the alerts, Shapley values [Sha53] are used. Shapley values define how to fairly distribute a payout among contributors in a coalitional game according to their contributions. In the case of an ML model, the contributors are features fed to a classification model and the payout is the final classification score of an instance. The algorithm underpinning the Shapley values calculates the average of all marginal contributions of a feature by all possible coalitions. The formula for calculating Shapley value for a certain feature is:

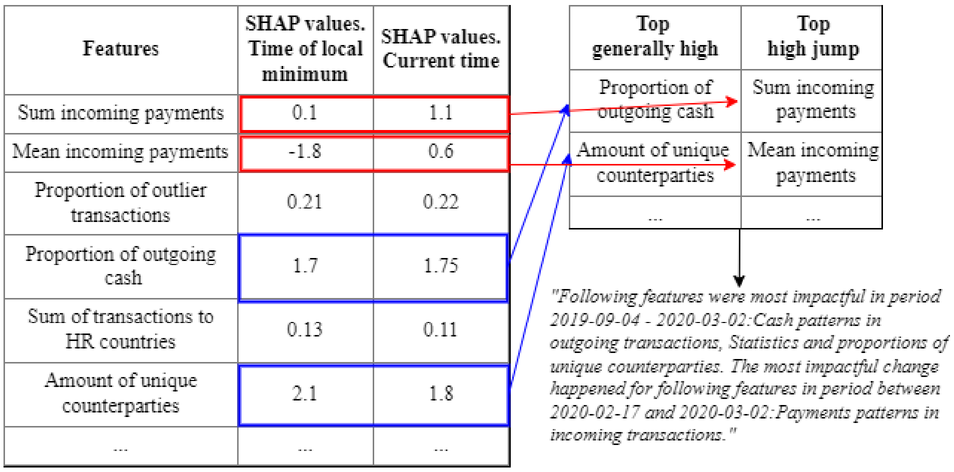
$$f_i = \frac{1}{|N|} \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} (v(S \cup \{i\}) - v(S)) \tag{5.4}$$

where N - is the set of features, v – a value function (a payout function for a coalition of features), i – the index of a feature, and S – a subset of set N. More details and intuition behind Shapley values can be found in the original paper (Shapley [Sha53]). Calculations of Shapley values according to the original paper are infeasible to make for a current set-up since it requires estimating the impact of every possible feature coalition, i.e. training a separate model for it. Thus, the SHAP framework [LL17] for calculating the approximated Shapley values was used, further addressed as *SHAP values*.

In our setup, SHAP values on each model run are calculated which gives us a picture of how each feature contributed to a model output score of each customer at each point in time. To provide an explanation of the prediction, 5 features with top SHAP values were selected – this corresponds to the first requirement of the MS (base threshold). Additionally, we select 5 features with the highest deltas of SHAP values between the time of classification and the time of the closest local minimum – this corresponds to the second requirement of the MS (jump threshold). Simply stated, the features which provided the highest contribution and the features whose contributions have increased the most are highlighted. All the extracted features are grouped into several categories (e.g., features which represent the turnover, counterparty-related features, cash transactions features, etc.) after which they are described with humanly-readable definitions. Having identified the most contributive features, we map them to the predefined feature groups and form the textual description based on feature group definitions. Eventually, all the extracted features are mapped to 24 feature groups.

Note that if the alert is raised in the very first weeks after bootstrapping the MS, we do not have features with deltas of SHAP values, since there is nothing to compare the SHAP values with. In this case, the 5 features with the highest SHAP values are used.

A schematic view of the interpretation layer of the alert generation approach is given in Fig. 18. The figure shows that features capturing the cash patterns of outgoing transactions and the number of unique counterparties have the highest SHAP values, thus contributing to the predictions made by the model the most. Meanwhile, the contributions of sum and mean of the incoming payments have increased the most - on 1.0 and 2.4 points respectively (difference between SHAP values at the time of local minimum and SHAP values at the current time for those features).



Features	SHAP values. Time of local minimum	SHAP values. Current time
Sum incoming payments	0.1	1.1
Mean incoming payments	-1.8	0.6
Proportion of outlier transactions	0.21	0.22
Proportion of outgoing cash	1.7	1.75
Sum of transactions to HR countries	0.13	0.11
Amount of unique counterparties	2.1	1.8
...	...	...

Top generally high	Top high jump
Proportion of outgoing cash	Sum incoming payments
Amount of unique counterparties	Mean incoming payments
...	...

"Following features were most impactful in period 2019-09-04 - 2020-03-02: Cash patterns in outgoing transactions, Statistics and proportions of unique counterparties. The most impactful change happened for following features in period between 2020-02-17 and 2020-03-02: Payments patterns in incoming transactions."

Figure 18. Schematic example of proposed interpretations. The first step is to calculate the SHAP values for each feature as of the current time, and as of the closest time when the monitoring score reached a local minimum. The second step is to identify the 5 features with the highest SHAP and the 5 features with the highest deltas of SHAP. The third step is to produce a human-readable presentation of those features.

5.2. Evaluation

In this section, an empirical evaluation of the proposed MS using real-life data from a multinational financial institution and feedback from AML domain experts is presented. The aim of the evaluation is to address the following research sub-questions:

- **RsQ1**: To what extent does the proposed custom MS reduce the number of redundant alerts (per customer) relative to a baseline MS that fires alerts based on a single class probability threshold?
- **RsQ2**: How to correctly estimate the predictive performance of the proposed MS under a baseline configuration?
- **RsQ3**: To what extent does the tuning of the proposed MS improve its predictive performance?
- **RsQ4**: To what extent do domain experts perceive that the alerts generated by the custom MS are relevant (perceived relevance)?
- **RsQ5**: To what extent do domain experts perceive that the alerts generated by the custom MS are useful as a starting point for an investigation?

5.2.1. Dataset

To address the above research questions, an anonymized dataset of transaction histories extracted from the bank's database was used. The extracted dataset covers a subset of ∼330000 customers from three countries, across 81 weeks, and a total of more than 240 million transactions. Among the customers in the dataset, 0.4% are "reported" customers and 99.6% are "randomly sampled" (not reported) customers; 8% of corporate and 92% of private customers. All customers have a transaction history from March 2018 to April 2020. Each customer is represented with weekly shifted 6 months snapshots (see Fig. 14). If a customer did not have transactions during some week then the corresponding 6-month snapshot was not extracted since the customer was already monitored on the previous iteration and no new information was added. The training period was September 2018 – October 2019 (56 weeks) and the testing (monitoring) period was October 2019 – April 2020 (25 weeks). We use a single temporal split since this reflects the way the model would be trained and used in practice – training occurs at a given time point T based purely on data available before or at time T, and testing occurs based on data available after point T, during a certain testing period [She13]. The dataset represents the same customer base as used in Chapter 4 but with transactions from a longer period of time.

The data was used to train a classification model that predicts whether or not a customer, at a certain time point T, is engaged in activity similar to those of reported customers or not. In other words, the label used is "reported" versus "not reported" customer. The approach used to train the classification model was described in further detail in Chapter 4.

5.2.2. Handling of reoccurring alerts (RsQ1)

The first experiment addresses RsQ1 by comparing the proposed (custom) MS with a baseline (default) MS where the alerts are raised when the probability of a customer's behavior being illicit exceeds the predefined threshold (standard way to convert probabilities into classes). With respect to this question, the working hypothesis is that the baseline MS would generate periodic (weekly) alerts for several weeks in a row, most of them redundant, whereas the custom MS would generate redundant alerts, for a given customer, only sporadically. For this experiment, a base threshold of 0.5 and a jump threshold of 0.2 were used for the custom MS and a threshold of 0.5 for the default MS (using calibrated probabilities). The parameters of the utility trapezoid parameters were set as follows: the beginning of the lower base – 16 weeks before the report date, the end of the lower base – 8 weeks after the report date, the beginning of the upper base – 6 weeks before the report date, end of the upper base – 2 weeks before the report date. This means that the alerts raised 4 months before or 2 months after the report date do not have any utility, and the alerts raised 6 weeks before to 2 weeks before the report date have the maximum possible utility. Note that if an alert was raised by the custom MS it is raised by the default MS as well, but not vice versa (see Fig. 19).

To highlight that the default MS does not handle the reoccurring alerts requirement, while the custom MS does, we plotted the distribution of the number of raised alerts per customer (see Fig. 20). It can be seen that the custom MS raised only up to 3 alerts per customer for the testing period of 25 weeks. At the same time, the default MS raised up to 25 alerts per customer, which means that some

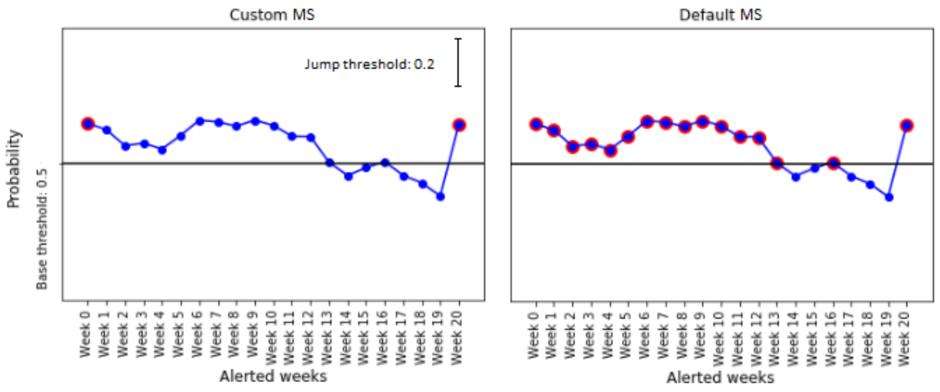


Figure 19. Example of alerts raised by MSs for a given sequence of monitoring model scores of a single customer. Custom MS (left plot) with thr_base = 0.5 and thr_jump = 0.2; and default MS (right plot) with thr=0.5. Blue dots are model output scores on each week of monitoring and red dots are weeks, when the alerts are raised

customers were alerted every week of monitoring.

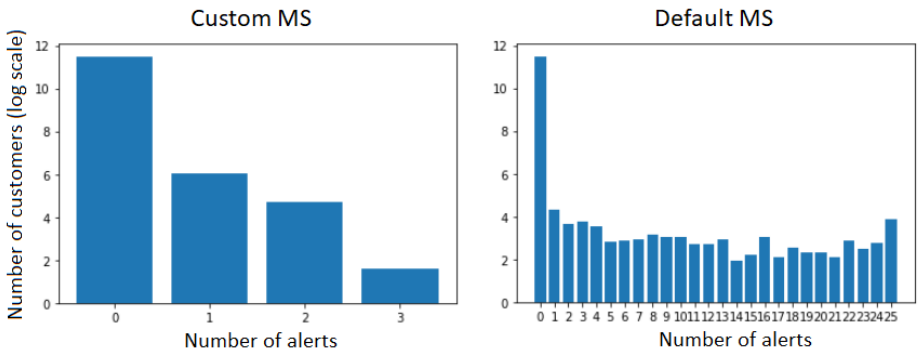


Figure 20. Distribution of the number of alerts raised per customer by custom MS (left plot) and default MS (right plot)

5.2.3. Performance of custom MS (RsQ2)

The next question is how to assess the performance of the custom MS. When performing the AML monitoring, we care not only about how many customers got alerted, but also how many alerts were raised for those customers. This means that we have to analyse two confusion matrices (CM) to understand the performance of the MS:

1. **Alert level CM** – the CM where one object is one customer-snapshot. So there are multiple objects related to one customer present in the CM. CM on alert level is defined as follows:
    - TP – an alert was raised for a snapshot of a reported customer
    - FP – an alert was raised for a snapshot of a never-reported customer

Table 7. Alert level CM (left) and entity level CM (right) for alerts raised by custom MS on customers from the test set

Alert level				**Entity level**			
	Predicted				Predicted		
True		Class 0	Class 1	True		Class 0	Class 1
	Class 0	1870981	277		Class 0	97670	241
	Class 1	9856	381		Class 1	289	298
Recall = 0.037		Precision = 0.579		Recall = 0.508		Precision = 0.553	
Custom recall = 0.433							
Custom precision = 0.401							

  - TN – an alert was not raised for a snapshot of a never-reported customer
  - FN – an alert was not raised for a snapshot of a reported customer

2. **Entity level CM** – the CM where one object is one customer. So the results of model monitoring are aggregated on the entity level by the rule that if at least one snapshot of a customer created an alert (within an appropriate time frame), the customer gets a positive prediction in the entity-level CM. Specifically, the CM at the entity level is defined as follows:
   - TP – a customer was reported at least once and an MS raised at least one alert
   - FP – a customer was never reported and an MS raised at least one alert
   - TN – a customer was never reported and an MS did not raise any alert
   - FN – a customer was reported at least once and an MS did not raise any alert

Table 7 shows the entity-level and alert-level CMs for the custom MS on a test set of 587 reported and 97911 randomly sampled customers over 25 weeks of monitoring.

In the answer to RsQ1, it was shown that custom MS handles reoccurring alerts and there were at most 3 alerts raised for a single customer for the test monitoring period of 25 weeks. Thus, from the CMs above, it can be noticed that there were 381 alerts raised (alert level TP) on 298 reported customers (entity level TP). And at the same time, there are 9856 snapshots of reported customers where alerts were not raised (alert level FN) and 289 reported customers that did not get any alert (entity level FN). We note that the standard alert level recall (calculated as $recall = \frac{TP}{TP+FN}$) is strikingly small. This was hypothesized earlier – if a reported customer got 1 alert in a test monitoring period of 25 weeks, then in the alert level CM we get +1 to TP and +24 to FN. And since recall should represent the proportion of reported customers that got alerted, we come to the conclusion that recall must be calculated on the entity level.

From Table 7 it can be seen that there were 277 alerts raised (alert level FP) on 241 randomly sampled customers (entity level FP). The precision (calculated as $precision = \frac{TP}{TP+FP}$) although it shows similar numbers for both alert level and entity level CMs, must be calculated on the alert level, since precision represents the proportion of correctly raised alerts for AML monitoring task. The end users of the MS are AML domain experts who investigate raised alerts, and correctly calculated precision (precision on alert level) gives a more correct assessment of the system's performance and helps in the estimation of human workload. To conclude, according to the business needs, the precision of the AML MS has to be calculated on the alert level while recall has to be calculated on the entity level.

As stated earlier, one of the key requirements of monitoring setups is the timing of raised alerts. Very early alerts or very late alerts should be considered as FPs. But CM and the standard definition of recall and precision do not have the capacity to reflect the timing aspect of raised alerts, thus those metrics misrepresent the actual performance of the MS. In order to provide not distorted performance estimation by taking into account timing aspects, custom recall and custom precision were designed (see Section 5.1.2). As explained above, the proposed custom recall is calculated at the entity level, and it counts as TPs only alerts raised close to a report date. Custom precision, in turn, also considers alerts raised close to a report date while penalising early and late alerts.

For the custom MS with the parameters defined above (base threshold = 0.5 and jump threshold = 0.2), we obtained a custom recall of 0.433 and a custom precision of 0.401. If to compare these values with the entity-based recall and alert-based precision (recall = 0.508 and precision = 0.579 as shown in Table 7) a substantial difference can be seen. This observation emphasizes the importance of taking into account the timing of alerts in the definition of performance measures to get the correct estimation of AML MS performance since standard metrics are highly overestimating the real performance of the MS.

5.2.4. Tuning the custom MS (RsQ3)

To analyze the behavior of custom MS we made a grid search of parameters and calculated custom recalls, custom precisions and custom F1 scores (see Fig. 21).

One can observe that the highest recall is achieved with the smallest possible threshold values. This is correct because, with the smallest thresholds, the maximum number of alerts will be raised, and thus will catch the maximum number of customers with illicit behavior. The highest precision is achieved when the base threshold and jump threshold are both in a range of 0.7-0.8 which shows that with very high thresholds there should be a very limited number of alerts raised per customer, and they are raised in the most proper time. As for the F1 score, the highest numbers are achieved when the base threshold and jump threshold are in a range of 0.5-0.6. At those ranges, the equilibrium between precision and recall

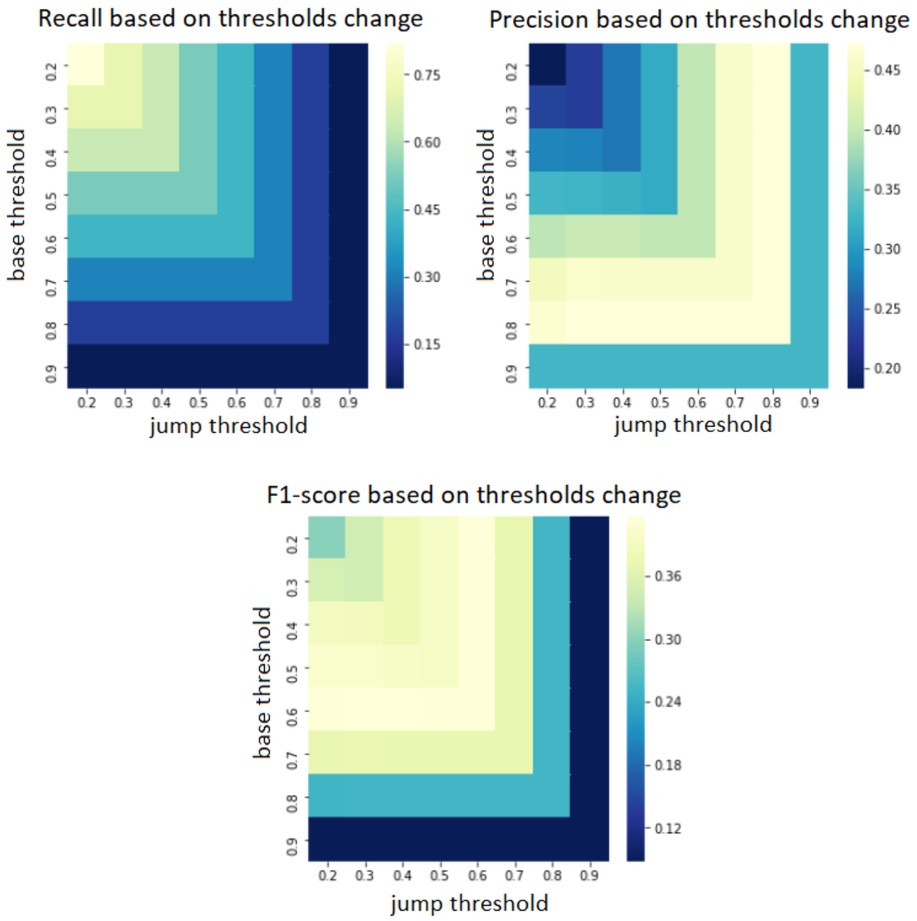


Figure 21. Heat map of custom recall (top left plot), custom precision (top right plot) and custom F1 (bottom plot) scores based on base threshold and jump threshold of the custom MS on a test set

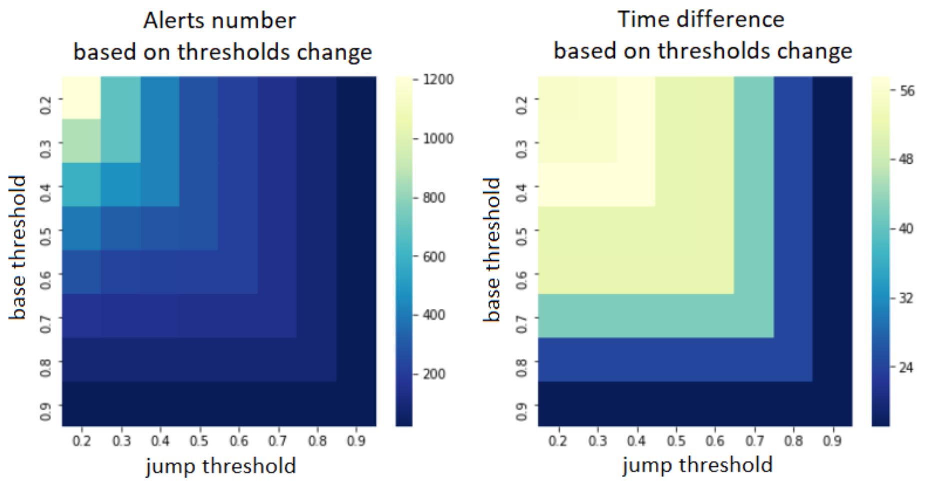


Figure 22. Heat map of alerts amount (left plot) and the average time between alert date and report date in days (right plot) based on base threshold and jump threshold for custom MS

is reached. Nevertheless, the final decision on parameters should be made by the subject-matter experts.

Since the timing of alerts is crucial in the context of an MS, we measured the average time between the report date and the date of the first alert raised for reported customers from the test set with different parameters of custom MS. Together with it, we measured the number of raised alerts which directly defines the workload of the end users (see Fig. 22).

From the plot of the time difference between an alert date and a report date, we can see that the parameters that maximize the F1-score (base threshold and jump threshold are both in a range of 0.5-0.6) has an average time difference of 45 – 50 days. This coincides well with the time period where the maximum utility for raised alerts is given – 6 weeks before to 2 weeks before the report. This observation suggests that the F1 score is an appropriate measure for parameter threshold tuning. In the alerts number heat map, one can observe that the lower the thresholds are, the more alerts are raised, as expected. The subject-matter experts may use such heat maps to make a decision on the thresholds.

5.2.5. Evaluation with domain experts (RsQ4 and RsQ5)

The usefulness of the alert explanations was assessed via 3 testing rounds. In each of these testing rounds, we randomly selected 8 alerts raised by the custom MS for each of the 3 countries covered in the study, leading to a total of 24 alerts per testing round (see Table 8). Each alert was analyzed by an AML domain expert from the country in question. Based on this analysis, the investigators stated the relevance and accuracy of the alerts and the usefulness of the associated explanations. For the purpose of this experiment, we will focus solely on the assessment

Table 8. Summary of samples testing rounds

Round	Considered period	Number of alerts
1	01.2020 - 02.2020	24
2	02.2020 - 03.2020	24
3	03.2020 - 04.2020	24

of the explanations. As explanations for the alerts, the subject experts saw 5 of the most contributive features (top SHAP values) and 5 features which had changed the most (top delta SHAP values) digested in a human-readable representation of feature groups (FG). Each alert is represented with up to 10 FGs. More than one contributive feature can fall in the same group, e.g. two features, mean of incoming payments and the sum of incoming payments fall into one FG "Payments patterns in incoming transactions". As explained above, in Section 5.1.4, if the alerts are raised on the very first week of monitoring then the explanation of the alert will consist only of the 5 most contributive features.

We took all the FGs that appeared during all 3 sample testing rounds and calculated frequencies of FGs found to be useful as the most contributive (top SHAP) characteristic and as the most changing (top delta SHAP) characteristic in explanations (see Table 9 for details). A value of 7/13 in this table means that during a testing round, a given FG appeared 13 times (for 13 different raised alerts) in the alerts explanations and 7 times it has been found useful by the domain experts. Some FGs did not appear in the explanations at all since they never happened to be the most contributive or the most changing ones. There was no notable difference in alert explanations perception among domain experts from the three countries, so it was decided to keep the final results in a common table in order to not create confusion for readers.

One can observe that some of the FGs are dominant and present almost in every alert, most notably "Transaction amount patterns" and "Cash patterns in outgoing transactions" while there are FGs which appeared only once for all three testing rounds, most notably "Statistics of intertransaction times". The dominant FGs represent general customers' behavior, while dominated FGs represent specific aspects of the behavior.

Based on received feedback from AML domain experts after the first round (and subsequently after the second round), we improved the way the explanations were presented. This can be noticed, for example, in the FG "Connection with previously reported customers". This feature group was not found useful by the domain investigators in the first testing round because it was presented just as a fact that a customer when got an alert has been in contact with known reported counterparties. It appeared that the initial explanation associated with this FG did not help the domain experts because it did not highlight the exact reported counterparties. We, therefore, updated the feedback to accompany the explanation

Table 9. Frequencies of times FGs were useful during 3 sample testing rounds

Feature group	Round 1		Round 2		Round 3	
	# of times FG useful as most contributive characteristic in explanations	**# of times FG useful as most changing characteristic in explanations**	**# of times FG useful as most contributive characteristic in explanations**	**# of times FG useful as most changing characteristic in explanations**	**# of times FG useful as most contributive characteristic in explanations**	**# of times FG useful as most changing characteristic in explanations**
Transaction amounts patterns	18/24	8/10	23/24	3/3	19/23	0/0
Proportion of Foreign vs Domestic vs Intrabank payments	7/13	1/3	13/14	1/3	17/18	0/0
Cash patterns in outgoing transactions	5/6	2/3	3/3	1/1	11/11	0/0
Cash patterns in incoming transactions	8/10	2/2	8/8	1/1	8/8	0/0
Payments patterns in outgoing transactions	0/0	1/1	0/0	0/0	1/1	0/0
Payments patterns in incoming transactions	6/10	3/5	14/14	4/4	13/18	0/0
Connection with previously reported customers	2/10	1/3	10/11	6/7	2/2	0/0
Channels usage patterns	5/11	1/1	2/2	1/1	10/10	0/0
Different currencies patterns	3/3	2/2	2/4	1/1	2/2	0/0
Smurfing patterns	2/6	3/5	1/1	2/2	2/2	0/0
Statistics and proportions of unique counterparties	0/0	1/1	0/0	0/0	0/0	0/0
Other account activity patterns (not cash nor payments)	1/3	1/2	2/3	0/0	1/4	0/0
Patterns of transaction values used (e.g. only few unique values or many round values)	0/0	0/1	0/0	0/0	0/0	0/0
Difference in aggregated sums of consecutive months	1/1	0/2	0/0	1/2	0/0	0/0
Counterparty country/bank patterns	0/0	1/2	0/0	0/0	0/0	0/0
Statistics of intertransaction times (time between consecutive transactions)	0/0	1/1	0/0	0/0	0/0	0/0

of each alert that referred to this FG with the counterparties in question. After this update, starting from round 2, we notice an improvement in the perceived usefulness of this FG. Also, in the first round, some of the FGs were named in a way that domain experts found confusing and we re-formulated them for the second and third testing rounds. To conclude, based on the feedback from end users of the MoL detection MS, the generated alert explanations appear to be helpful in initiating investigations.

5.3. Summary

In this chapter, we focused on **RQ2** and **RQ3** stated in the thesis. We addressed the problem of designing an offline MS which uses MoL risk scores from the ML classification model and determines when the alerts should be raised. In such a setup the MS is suffering from two problems: repetitive alerts for the same MoL behavior and correct timeliness of raised alerts. We presented a study of a predictive MS used in the context of AML monitoring in a multinational financial institution. The proposed MS addresses the requirements of generating accurate, non-redundant and timely alerts. The underlying ML-based MoL detection model is trained on snapshots of customers' behavior, with labels coming from historical reports sent on identified cases of MoL. To generate not-redundant and timely alerts, the output scores of the ML model are passed to a designed MS which raises alerts only when and as soon as customers' behavior changes. Custom prediction quality metrics that take into account the reoccurring alerts and timing of the alerts generated by the MS are used. These custom performance metrics are used to optimize the MS parameters, hence ensuring that the optimized MS generates alerts close to the time when the corresponding investigation is relevant and prevents raising redundant sequential alerts.

The presented offline MS is also designed to address interpretability requirements, specifically the requirement that the generated alerts should be accompanied by explanations both related to the reason for the alert and the timing of the alert. The approach for generating explanations is based on SHAP values but lifted at the level of high-level feature group descriptions, designed to help AML domain experts to start investigations.

The proposed method was designed in the context of a real-life industrial application in a multinational financial institution and evaluated using real-life data and feedback from AML investigators. The evaluation showed that a standard baseline MS generates a significant number of redundant alerts, compared to the custom MS approach. It has been shown that according to AML business needs, for correct MS performance assessment recall should be calculated at the entity level, while precision should be calculated at the alert level. In other words, false positives should be avoided at the alert level (false alerts), while false negatives should be avoided at the entity level (uncaught illicit customers). From the experiments, it was clearly seen that standard recall (calculated on entity level) and precision (calculated on alert level) substantially overestimate the actual performance of the MS, since they don't have the capacity to take into account reoccurring alerts and timing of the alerts aspects. On the opposite, custom metrics do take into account the aforementioned AML aspects, thus giving a more correct assessment of the MS performance. In summary, the evaluation showed that a predictive MS optimized using the proposed custom metrics allows us to tackle the two issues at stake: timing of alerts and avoiding redundant reoccurring alerts.

The empirical evaluation also assessed the interpretability of the generated alerts via three testing rounds with AML domain experts. This evaluation showed that the investigators perceived that the explanations were useful as a starting point to investigate the corresponding alerts. As a future work for interpretability improvements, it is considered to apply a counterfactual explanations approach for alert explanations generation. This approach may enrich generated textual explanations with "what if" type of statements, which could highlight the minimal changes in customers' behavior needed to change the classification model decision.

The empirical evaluation with domain experts has a number of threats to validity and limitations. First, the evaluation has the typical ecological threats to validity associated with the research, specifically, limited generalizability due to the fact that the study was conducted in a single organization. However, we note that the evaluation involved three countries and different domain experts from each country. Second, the empirical evaluation was made using a reduced number of alerts. A larger-scale evaluation is needed to draw more generalizable conclusions. Third, the evaluation focused on perceived relevance and perceived usefulness. A more in-depth evaluation of user acceptance in daily usage settings could lead to further insights into potential barriers to the full adoption of AML monitoring systems.

6. GROUP MOL PATTERNS DETECTION

In this chapter, we address the **RQ4**: *"How to make the designed system resistant towards different classes of illicit behavior, particularly individual vs group MoL behavior?"*. We present a design of a system for group MoL behavior detection. The developed MoL detection system uses data about financial connections between actors derived from transaction histories, to construct a (partial) social network, and then detects illicit group behavior over this network.

The development of the system was driven by two key challenges that financial institutions face in their fight against financial crime:

- The fact that group MoL patterns keep evolving over time, in such a way that relying on previously observed and/or detected patterns is ineffective in the medium to long term. Actors who engage in illicit financial behavior constantly evolve their schemes and are aware that their operations are monitored by systems that look for certain types of known MoL patterns.
- The fact that not all transactions between the actors under observation are known by the financial institution. Indeed, a given financial institution only has information about transactions performed between pairs of accounts inside the institution, as well as transactions from/to an account in the institution. It has no information about transactions that the actors holding these accounts might have performed in other financial institutions.

In addition to being designed to address the above challenges, the MoL detection engine was designed to comply with the following requirements:

1. **Accurate.** Despite being based on incomplete network information, the alerts raised by the MoL detection system must be accurate and should likely lead to new investigations followed by reports sent to the corresponding authorities.
2. **Actionable.** The detected MoL patterns must be investigated by AML domain experts before actual mitigation measures are taken by the financial institution. Thus, the raised alerts have to be such that they can be investigated and validated by a human.
3. **Scalable.** The developed group MoL detection system must be able to process large networks of customers' transactions (hundreds of thousands of nodes, billions of edges) in a reasonable time. This requirement poses a challenge, as most of the graph algorithms considered in previous work on financial crime detection have a non-linear complexity.

In this chapter, we present a study where the above problems are tackled in the context of the same multinational financial institution discussed in the previous two chapters.

The rest of the chapter is organized as follows. Section 6.1 describes the proposed solution and used data. Section 6.2 defines the model's validation strategy and reports the experimental results. Section 6.3 sketches the directions of future research and summarizes the main conclusions and contributions.

6.1. Approach

The goal of the research reported in this chapter is to find MoL patterns involving groups of actors, when those patterns may evolve over time. Accordingly, the proposed approach does not seek to identify specific patterns of interactions between actors, instead, it tries to find subgraphs of closely connected actors and analyse which of them possess a high risk of MoL.

On a high level, the process of the proposed group MoL behavior detection system is shown in (see Fig. 23). First, the data is loaded from the database and the graph of customers' interactions is constructed out of loaded transactions tables. To reduce the noise and decrease the graph's connectivity, some of the transactions are filtered out from the graph based on the level of risk they bring. Next, the graph is divided into overlapping communities, which are later analyzed in terms of MoL behavior presence. Detected communities are characterized by diverse features to be fed to an ML anomaly detection model. At a stage of post-processing, anomalous scores for detected communities are adjusted and ranked according to a risk-based approach, and alerts are raised on the scores that are deemed anomalous. In the final step, AML domain experts analyze raised alerts and make decisions about whether those alerts have to be investigated further or mark as false positives. The investigated alerts are either closed, if no evidence of MoL was found, or reports are sent to the local FIU and correspondent mitigation measures are taken if such evidence is present.

6.1.1. Graph construction

The very first step of the detection of graph-structured MoL is graph construction. As there are many ways customers may interact with each other using the infrastructure of the financial institution, there are many different types of vertices and edges in the graph of customers' relationships. For the purpose of the research, only account payments are considered as edges of the graph. Accordingly, vertices in the payments graph are accounts from which the payments are done. Since one customer may have multiple accounts, it is possible that a customer's financial behavior is represented with multiple vertices in the graph. Accounts in the graph can belong to customers of the financial institution, about whom there is data in the warehouse, or foreign customers, about whom there is almost no data. Payments which constitute graph edges can be of three types: intrabank (both sender and receiver are the customers of the financial institution); domestic (both sender's and receiver's accounts were opened in the same country, one of the parties is a customer of the financial institution); or international (one of the parties is a customer of the financial institution, an account of the other party has been opened in a foreign country and a foreign bank). Except from sender and receiver

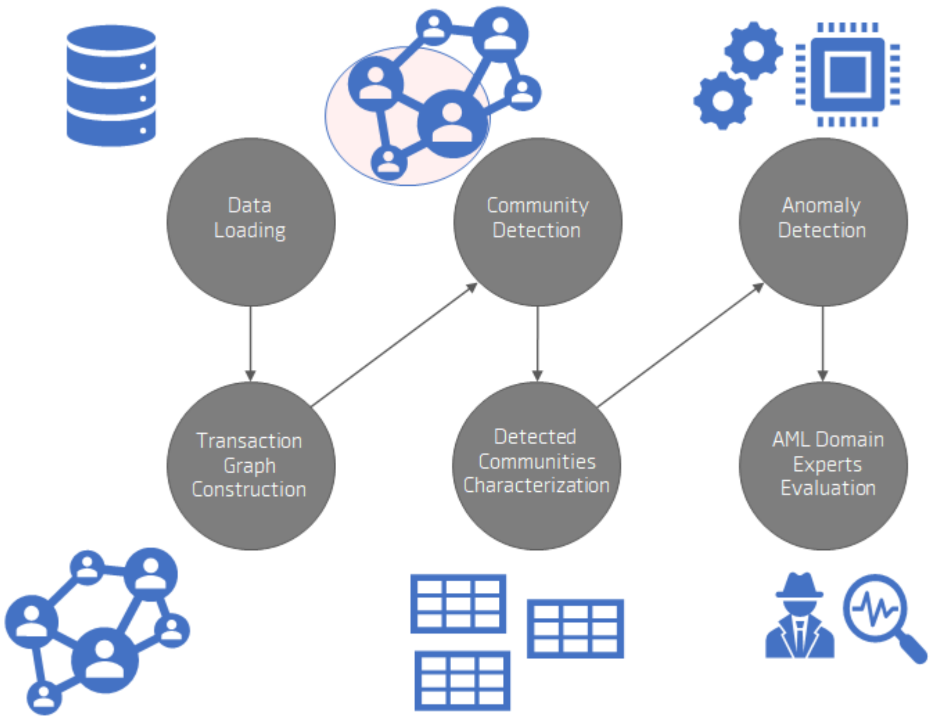

Data Loading
Transaction Graph Construction
Community Detection
Detected Communities Characterization
Anomaly Detection
AML Domain Experts Evaluation

Figure 23. Automated group MoL behavior detection process diagram

data, a transaction is also described by date, time, amount, currency, transaction type, channel and other characteristics.

The graph is built on transactions of customers from three jurisdictions where the financial institution provides services. Naturally, the transactions graph is dynamic with time. To be able to process it, the dynamic time-evolving graph was converted into a static graph by fixing the timeframe of consideration. For the purpose of this research, we took a window of transactions for 3 months for the static graph. The graph has 108M edges and 7.3M nodes within this timeframe. Out of all nodes, 60% are the financial institution's accounts and 40% are foreign accounts. As for transaction types, there are 60.8% intrabank payments, 35.6% domestic payments and 3.5% foreign payments in the constructed graph. From the customer type perspective, the graph consists of 56.5% of private accounts, 3.5% corporate accounts and the rest 40% are unknown as they belong to foreign accounts.

To decrease the connectivity of the graph we filtered out small-amount transactions which bring low MoL risk. We defined amount thresholds based on the type of relationships (private-to-private, private-to-corporate, corporate-to-corporate). Transactions lower than correspondent thresholds were removed from the graph. After all filtering and graph cleaning, the number of edges was reduced to 12.6M and the number of nodes to 4.4M.

6.1.2. Community detection

The next step after the graph creation stage is the detection of densely connected communities of customers. Practically, at this stage, the graph of customers' payments must be partitioned into smaller subgraphs so that each subgraph is composed of closely related parties. Those communities will further be characterized and passed to an ML model for anomaly detection.

Naturally, the graph of customers' transactions is heterogeneous, with denser and sparser regions. With a long period of transaction history (3 months for this research) it is clear that almost all customers are connected in one way or another (through common 3rd party payment service providers, international grocery store chains, large enterprise service providers, etc.), which makes the task of community detection more complicated. In practice, the original transactions graph used for the research consists of three huge connected components, which represent the major part of the economy, and a bunch of smaller connected components. That means it is impossible to take connected components as communities for further anomaly detection. The three largest connected components constitute 89% of all nodes and the rest 11% of all nodes are represented in $\sim$175k of smaller connected components (sizes 2 to 100 nodes). The smaller connected components can already be analyzed in terms of being anomalous, but the largest connected components have still to be partitioned into parts with denser connections. For this purpose, the LPA [RAK07] was applied, which allows us to detect communities in the largest connected components of the transactions graph.

LPA is a fast iterative algorithm for community detection with no requirements for prior knowledge. At the very first iteration, each node is assigned its own label. The labels propagate through the network and for each next iteration, a label of a node is updated with the label that the maximum of the neighbours of the node has. With such logic, densely connected groups of nodes quickly converge to the same label and the rest of the labels vanish. The algorithm stops after convergence is reached, or when the predefined number of iterations has passed. The advantage of the LPA is that it is fast to compute since calculations can be done in a multiprocessing setup. The disadvantage of the algorithm is that convergence is not guaranteed.

When applied to the transactions graph, the LPA failed to detect communities and converged into labelling all the nodes into a single label, i.e., the LPA considered the graph as an inseparable network. To make use of the LPA, we artificially reduced the connectivity of the graph by splitting the densest nodes into pseudo-nodes. For each node where the number of unique counterparties was bigger than a predefined threshold (set to be 100 for this research), we split the nodes into pseudo-nodes as shown in Fig. 24.

On Fig. 24 we can see an example where node #1 was split into seven pseudo-nodes because node #1 surpassed the unique counterparties count threshold (which, for this example is equal to 7). The pseudo-nodes are not connected to each other and are only connected to correspondent counterparties of the original split node.

Applying this splitting action allowed us to analyze large and active accounts from different angles and from different perspectives. Otherwise, the behavior of an active account, if not split, creates a huge community containing its entire financial behavior which is technically impossible to be analyzed by a domain expert and makes the solution not actionable. It is true that artificially reducing the connectivity of the graph could lead to missing out on identifying complex laundering schemes but it is a tradeoff between actionability (smaller and more densely packed communities) and having all connections in the graph. On the other hand, non-splitted communities are still missing connections due to the fundamental limitation of accessibility of transactions within the boundaries of one financial institution. One positive effect of splitting, though, is that nodes with a large number of connections, e.g. a large grocery store will be partially present in numerous communities, representing different areas of financial activity of the grocery store. In this case, such corporate customers will be analyzed from different angles as part of multiple, communities. This arguably gives strong detective capabilities for the approach.

Further on, an ML anomaly detection model will analyze multiple communities containing different parts of the behavior of one account and will signal in the case an anomaly is found.

The nodes that have been split are either extremely active private customers

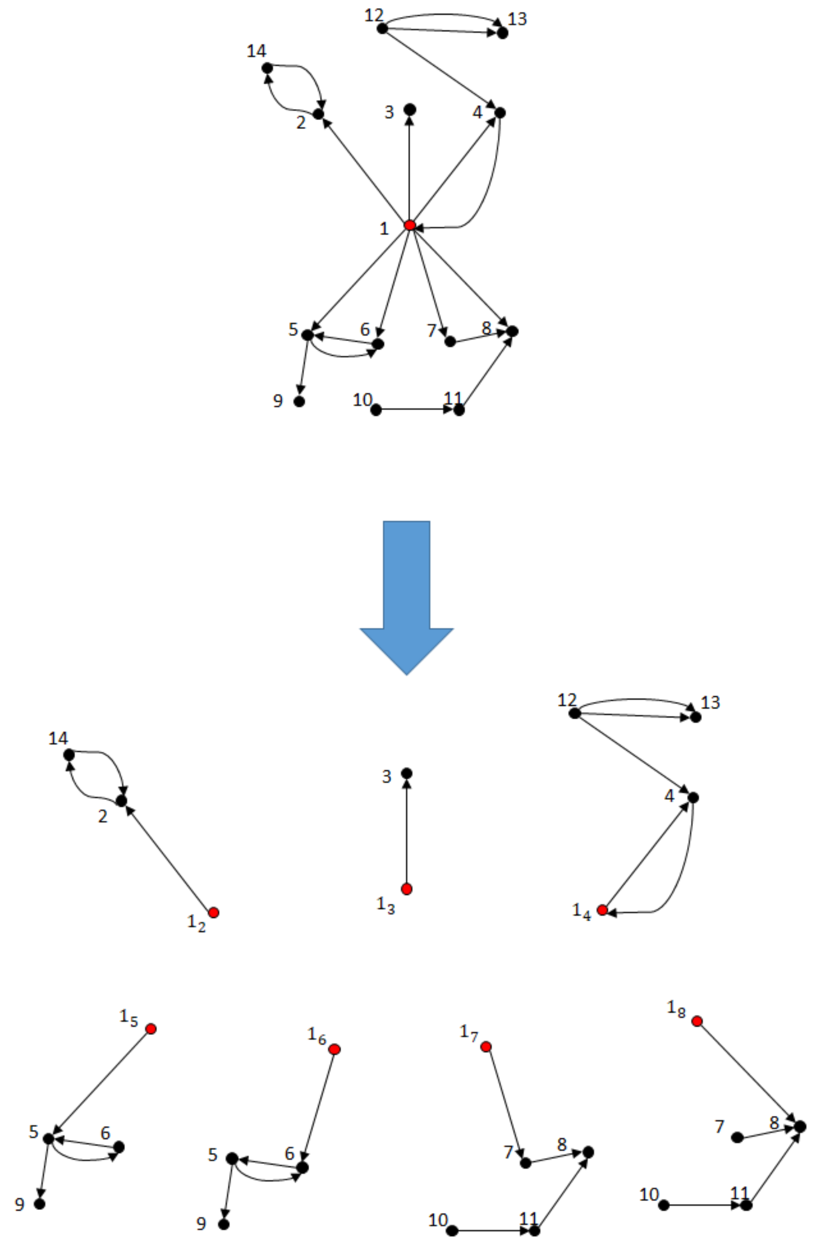


Figure 24. Example of a node split with a unique counterparty count threshold = 7

(1.5% of all split nodes), corporate customers (65% of all split nodes) or nodes representing foreign customers (the rest 33.5% of all split nodes). Eventually, there are <5k split nodes which constitute <0.1% of all nodes present in the graph. After applying the LPA on the split graph, >500k communities of sizes 4 to 100 nodes was acquired (see Fig. 25 with the distribution of the size of detected communities).

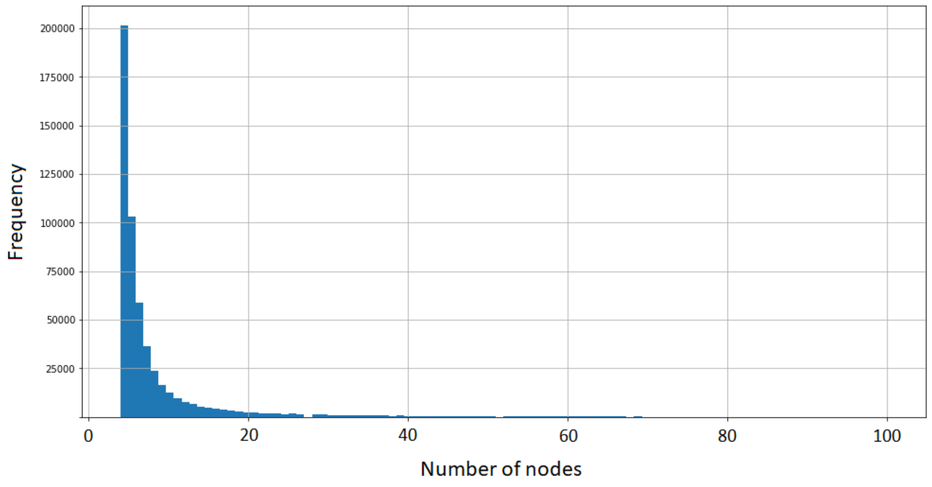


Figure 25. Distribution of size of detected communities

6.1.3. Feature extraction

In the approach, we focus on account-level transaction data and have two sources of raw data: account payments data and data about account owners. All the features can be separated into three sets: features describing the nodes of the analyzed communities, features describing the edges of the analyzed communities and features describing the graph structure of the analyzed communities. Eventually, there are 106 features that characterize detected communities. Below, each of these three groups of features is described.

Graph structure features. Based on previous work on the use of social network metrics to characterize communities [SN18; MGS21], we extracted the following set of features to represent the structure of each identified community:

- Count of nodes
- Count of edges
- Statistics of nodes' degrees – min, max, median indegree and outdegree
- Community triangle count: the maximum number of triangles a node in the community belongs to.
- Count of unique connections between each pair of source and destination.
- Community PageRank: the maximum PageRank of the nodes in the community.

- Connectivity ratio: the proportion of unique connections to the size of a complete community graph.

Node features. The above set of community-level features gives us structural information. To complement this information, we additionally extracted a set of features that aggregate information about the nodes (i.e. accounts or customers) in the community:

- Number of accounts belonging to the financial institution's customers
- Number of foreign accounts
- Number of unique customers present in the community
- Number of accounts belonging to private and corporate customers
- Proportions of customers by types (small corporate, medium corporate, large corporate, young private customer, mass private customer, etc.)

Edge features. To further complement the information about types of nodes (actors), we extracted a set of features corresponding to the characteristics of the interactions between actors, i.e. their transactions:

- Community amount features - different statistics (sum, mean, median, max, count) calculated over three types of transactions (domestic payments, intrabank payments, foreign payments) and for different original currencies in which transactions were performed
- Currency features - characteristics describing currencies used for payments. Here the number of unique currencies and the proportions by sum and by count of transactions made in different currencies is calculated
- Proportions by customer type – proportions by sum and count of transactions between corporate customers, of transactions between corporate and private customers and of transactions between private customers
- Proportions by channel – proportions by sum and count of transactions made through different channels (branch payments, ATM payments, internet payments, etc.)
- Proportion of own account payments – proportions by sum and count of transactions where source and destination accounts belong to the same customer
- Statistics of offshore payments – sums and counts of transactions to/from offshore zones
- Statistics of bridge payments – sums and counts of transactions to/from "bridge nodes", where the bridge is defined as a foreign customer who is connected to more than one unique customer from the same community

6.1.4. Anomaly detection

The next step in the pipeline is anomaly detection, where each detected community, characterized by the aforementioned features, is classified on being an anomalous community or a regular community. Several algorithms were tried for these purposes. The One-Class SVM algorithm was too slow, which made it impractical to use on such a big dataset. In turn, the Local Outlier Factor algorithm was unsuitable for the nature of the data and, expectedly, has shown a very poor performance. Eventually, an Isolation Forest (IF) algorithm was chosen as the algorithm for anomalous community detection. This algorithm uses a forest of binary trees built on a random subset of features in order to isolate data points from the rest of the data. The shorter trees are, the fewer splits are needed in order to isolate a data point. Thus samples with a shorter path from the root to leaf averaged across all the trees are considered to be more anomalous. IF algorithm possesses the key properties that are necessary for the developed MoL monitoring system:

- It is highly scalable since the model can be trained in a multiprocessing setup
- It can process high-dimensional inputs
- It returns a continuous anomaly score which corresponds well to the risk-based approach of MoL monitoring

In the post-processing stage, after the trained IF model scored communities, scores were adjusted in a risk-based approach. The scores of the communities where known risk factors were found (high amount of international transactions, transactions to/from high-risk jurisdictions, transactions to/from offshore zones, etc.) were adjusted accordingly.

6.2. Evaluation

The performance of the model was validated using the subset of MoL cases that were investigated and eventually reported to the local FIU. A customer is investigated if there is a need to contact the customer due to a lack of understanding of the financial activity. Further on, an AML investigator files a report to the FIU or closes the case if there is not enough evidence of MoL. In the research, a subset of existing investigated cases was used (from the same 3-month time window used for the graph construction) where MoL was suspected in customers' payments - $\sim$1900 investigated customers a quarter of which were reported to the FIU. The labels are obtained on a customer level while the analyzed entities are communities of customers, which means that the labels had to be broadcasted to the level of communities. According to the approach, if a customer was investigated/reported then a community a customer belongs to was considered to be investigated/reported.

For model validation, several metrics were used: precision, recall and F score. Precision and recall are the essential metrics for ML-based MoL detection models. In the AML domain, recall answers the question "What is the percentage of MoL cases that were detected?" and precision answers the question "What is the percentage of raised alerts that eventually indicated the MoL behavior?". Based on this definition, it can be concluded that the recall has to be calculated on a customer level, i.e. recall should be calculated as a proportion of alerted customers with detected MoL behavior among all such customers. At the same time, precision should be calculated on the entity level, i.e. precision should be calculated as a proportion of alerts (alerted communities) with detected MoL behavior among all alerts. More explanation and reasoning were explained in Section 5.1. The F_β score with β equals 0.5, 1, 2 was also measured since this metric considers both precision and recall in combination.

The aim of the first experiment is to measure the ability of the system to mark as anomalous the communities that must be reported. Due to the small number of available reported cases, a cross-validation strategy was used to receive unbiased performance measures. In this experiment, the anomaly detection model was trained using a 5-fold cross-validation approach with a grid search of hyperparameters where on each fold we compared communities marked as anomalous by the model against the reports obtained for the communities. Eventually, we selected three models that were maximizing $F_{0.5}$, F_1 and F_2 metrics respectively, averaged across 5 folds. The performance metrics for those three models can be found in Table 10.

Table 10. Performance of anomalous communities detection models for the case of reported communities

Model	Precision	Recall	$\mathbf{F_{0.5}}$	$\mathbf{F_1}$	$\mathbf{F_2}$
IF. Reported label. Maximized $F_{0.5}$	0.477	0.063	0.207	0.112	0.076
IF. Reported label. Maximized F_1	0.152	0.116	0.143	0.132	0.122
IF. Reported label. Maximized F_2	0.056	0.282	0.066	0.093	0.156

The aim of the second experiment is to measure the ability of the system to mark as anomalous the communities that must be investigated. During the investigation, an AML domain expert may request additional information from the customers or search other open-source repositories. So, the task for this experiment is different since not every investigated community ends up being reported. Like in the first experiment, we selected three models that provided the highest evaluation metrics averaged across 5 folds. The results can be found in Table 11.

In tables 10 and 11 one can find the performance results of anomalous community detection models, the performance of which was compared against known investigated and reported cases where MoL is suspected in customer's payments. The precision levels vary from 5.6% to 47.7%, and recall levels vary from 6.3% to 28.2% in the case of the model's ability to detect anomalous communities that have to be reported to the FIU. As for the anomalous communities that have to

Table 11. Performance of anomalous community detection models for the case of investigated communities

Model	Precision	Recall	$F_{0.5}$	F_1	F_2
IF. Investigated label. Maximized $F_{0.5}$	0.298	0.136	0.241	0.187	0.152
IF. Investigated label. Maximized F_1	0.198	0.363	0.218	0.257	0.312
IF. Investigated label. Maximized F_2	0.143	0.597	0.169	0.232	0.366

be investigated by AML domain experts, the precision levels vary from 14.3% to 29.8% and recall levels vary from 13.6% to 59.7%. The presented numbers are comparable to the performance of the transaction monitoring systems used by other financial institutions [Youb; Youa].

It appears that the detection system is better capable of identifying the communities that require an investigation, which is quite self-explainable. Not every case that goes through the investigation process is confirmed to have MoL patterns and eventually reported to the authorities, making it simpler from the ML perspective to detect a community that has to be investigated rather than reported.

Additionally, the statistical significance of obtaining the same results by a random guess was measured. A hypergeometric test was used to calculate the probability of the model to randomly claim a community to be anomalous in the case where all the communities can be of two types investigated and not investigated (reported and not reported). The hypergeometric distribution is used when sampling is done from the population of two mutually exclusive classes and sampling is done without replacement. The probability mass function of hypergeometric distribution is

$$P(x=k)=\frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \tag{6.1}$$

where in our domain:

- N – is a population size, i.e., all detected communities
- K – is the number of successes in the population, i.e., the number of investigated/reported communities
- n – number of draws, i.e., the number of communities classified as anomalous by the model
- k – number of observed successes, i.e., the number of communities classified as anomalous by the model which were investigated/reported

Hypergeometric tests presented in Table 12 show that the models' ability to identify anomalies is not random and the p-values for hypothesis testing are always practically around 0. That means that the probability of getting the same results with a model making a random guess when marking a community to be an anomaly is close to zero for all the experiments.

Table 12. Results of the hypergeometric tests for the models presented in the first and second experiments

Model	p-value
IF. Reported label. Maximized $F_{0.5}$	$2.16 \cdot 10^{-59}$
IF. Reported label. Maximized F_1	$8.72 \cdot 10^{-77}$
IF. Reported label. Maximized F_2	$3.69 \cdot 10^{-187}$
IF. Investigated label. Maximized $F_{0.5}$	$5.44 \cdot 10^{-276}$
IF. Investigated label. Maximized F_1	0
IF. Investigated label. Maximized F_2	0

6.2.1. Execution time

The system was developed using the parallel computing framework Apache Spark on Apache Hadoop Distribution File System (HDFS). As for the hardware, all the calculations were done using 1 YARN cluster, 56 cores and 503GB RAM in total. The execution time of different stages of the group MoL detection pipeline is presented in Table 13. As can be seen, the full pipeline of the proposed group MoL detection system on a graph of 108M edges and 7.3M nodes could be executed in approximately 6 hours, with the most computationally expensive stage being feature extraction.

Table 13. Average execution time for stages of the proposed group MoL detection pipeline

Stage	Average execution time	Framework used
Data collection	∼0.5h	Apache Spark
Graph construction	∼0.5h	Apache Spark
Community detection	∼1h	Apache Spark
Feature extraction	∼4h	Apache Spark
Anomaly detection	<5min	scikit-learn

6.3. Summary

In this chapter, the **RQ4** was addressed. We proposed a solution for group MoL behavior detection done under the assumption of incomplete network information available and the assumption of known MoL patterns inaccessibility. The approach uses unsupervised algorithms for community detection from the account payments graph and unsupervised ML algorithms for anomaly detection to detect suspicious communities. The system was built and validated using real-life large-scale financial data from a multinational financial institution that operates over three different jurisdictions. The model's ability to detect deviant behavior was validated using MoL cases that were investigated and reported to the authorities.

The proposed system complies with the accuracy business requirement since the performance was found to be reasonable according to standard industrial MoL detection systems' performance. The solution also complies with the actionability business requirement since the detected and analyzed customers' communities are feasible to be processed by AML domain experts. The system is scalable since it is designed using a parallel computation Apache Spark framework and could process large datasets in a reasonable time.

There are several limitations of the proposed solution. First, financial institutions are limited to accessing and analysing transactions made within their boundaries, while the transactions between 3rd parties that are customers of other financial institutions remain unknown and inaccessible. Thus, one cannot rely on local modularity community detection methods under missing connections in the graph limitation, as in the case of the account payments graph. Another limitation of the approach is splitting nodes into pseudo-nodes at the stage of community detection. On one side, it artificially reduces the connectivity of the graph and some resultant communities may appear not connected while originally they were. On the other side, it makes the detected communities to be actionable, i.e., makes it feasible for analysis by a domain expert. At the same time, the accounts of very active customers (those that fell under the splitting condition) are separated among multiple communities, which allows us to analyze their financial behavior from different angles.

A direction for extending the proposed system is to incorporate alert visualization and explanation. Alert explainability is an essential business requirement since it significantly simplifies the work of end users and shortens the time needed for an alert analysis and investigation. The potential solution for explanation generation could be adopted from Section 5.1. The biggest challenge with the explainability of MoL detection model outcomes is that the efficiency and correctness of the solutions can only be assessed by domain experts. This fact significantly complicates research in this area, since expert validation is very costly and sometimes even inaccessible.

7. CONCLUSION

7.1. Summary of contributions

In this thesis, four main contributions are presented for the research area of MoL detection, which address the four research questions (RQ1-RQ4) enunciated in Chapter 1. Below, the contributions and findings are summarized with respect to these questions.

RQ1: "How to design an ML-based system for MoL detection that does not require computationally expensive model training/execution and that can handle populations with extreme class imbalance?". This question is addressed via the first contribution, which is a framework for detecting individual MoL patterns by training an accurate ML model that is scalable and can handle class imbalance. The proposed framework uses a two-layered architecture with LR and extreme gradient boosting trees classifiers that filter out non-illicit customers in the first layer and classify heightened risk customers in the second layer. It was shown that two-layered architecture benefited to several requirements of the solution. Firstly, such an architecture significantly shortens the amount of computation needed which allows processing of more data per unit of time, thus making the solution scalable. Secondly, the architecture helps tackle extreme class imbalance thus making the solution more accurate, which was shown by the experiments.

RQ2: "How to measure the quality of a MoL monitoring system over time and in a way that links the outputs of the monitoring system to decisions and actions that its users need to take?". The second contribution is an offline MoL monitoring system that generates accurate, non-redundant, and timely alerts when potentially illicit MoL behavior is detected. We advocated that standard ML metrics cannot correctly estimate the performance of a MoL monitoring system. Therefore, the system was accompanied by a family of metrics that estimate performance more precisely than standard MoL metrics since they have the capacity to capture the time aspect.

RQ3: "How to cater for practical imperatives when designing the MoL monitoring systems with respect to interpretability of its outputs by AML investigators?". The third contribution is constituted by the interpretability module that complements raised alerts with textual explanations. The usefulness of explanations generated for raised alerts was tested by domain experts from different jurisdictions in a multiple-round session. The designed monitoring systems together with the interpretability module allow for meeting the actionability, interpretability and timeliness requirements of the solution.

RQ4: "How to make the designed system resistant towards different classes of illicit behavior, particularly individual vs group MoL behavior?". The fourth contribution is a framework for group MoL behavior detection that produces networks of potentially illicit MoL behavior as alerts. The framework uses transaction data to construct a partial social network and detect illicit group behavior MoL patterns without assuming previously detected MoL patterns or that all financial transactions are visible within the boundaries of the financial institution. The scalability of the framework is achieved by using the algorithms for community detection with a linear complexity and by using a parallel computation Apache Spark framework. The developed framework addresses RQ4 by allowing the overall solution to detect not only individual MoL patterns but also a more complex class of MoL - group MoL behavior. The framework for individual MoL pattern detection in combination with the framework group MoL behavior addresses the robustness requirement of the solution.

The framework for individual MoL patterns detection, the monitoring system that orchestrates the alert generation, the interpretability module to extend alerts with textual explanations, and the framework for group MoL patterns detection, together provide a complete and all-encompassing solution for automated MoL detection using ML techniques.

All four contributions have been evaluated on a real-life large-scale dataset with customer profiles, transaction histories, and labels provided by AML experts from three separate jurisdictions. The outcomes have been evaluated through computational experiments on historical data and live feedback from actual domain experts.

While conducting the research we followed guidelines from the Design Science in Information Systems proposed by Hevner et al. [Hev+08]. The outcome of the research work is a list of artifacts that constitute contributions 1-4 (GL1, GL5) and which are rigorously and systematically evaluated (GL2, GL4). The problem relevance is widely discussed in comparison with the current state-of-the-art in the current domain (GL3). All developed artifacts are the result of an iterative search process (GL6). The outcomes of the research are presented via two conference papers and one journal article to ensure that the results reach both academics and practitioners.

7.2. Future work

MoL is a global problem that requires continuous research and development of innovative solutions. This chapter highlights known limitations of the solution and proposes some potential areas for future work that can be explored to enhance the performance of ML-based MoL detection systems.

A fundamental limitation of the proposed solution is related to ecological validity. The problem is common in research and can limit the applicability of the study findings to other settings due to the fact that the evaluation was only performed within a single organization. However, it is worth mentioning that the evaluation was carried out across three countries and with experts from each country. Future work can focus on engaging other financial institutions in the research which will allow us to apply and test the proposed framework for MoL detection in a different setting and on richer data. Collaboration among financial institutions is needed in order to make the overall defence stronger. However, the very sensitive nature of the financial data and strict regulations in the GDPR area make such communication extremely challenging. Nevertheless, communication in limited scope exists, particularly in the private sector - some private companies are providing secure encrypted communication lines between banks. Finally, financial law enforcement is the entity which could possess cumulative information since it could request information regarding certain private and corporate entities from any financial institution it considers necessary. So FIU can have data from all the banks to initiate criminal proceedings.

Deployment of the proposed system requires addressing maintenance of the system in production. Maintenance poses a list of potential challenges both technical and non-technical. Technical challenges stem from the fact that the core of the presented solution is ML-based models which must be periodically retrained in order to match the expected performance (see Section 2.5.3). Models that are not retrained are prone to degrade with time due to the constantly changing modus operandi of the MoL actors and the data drift phenomenon. The system overall must be periodically tuned to provide the best possible performance. Future work can focus on tackling the technical challenges of solution maintenance.

Non-technical solution maintenance challenges are constituted by the fact the solution is required to provide interpretable decisions. The interpretability of ML-based systems is crucial, especially in the context of MoL detection. Similarly to MoL detection systems, interpretability modules must be updated over time using a feedback loop from the AML domain experts. Future work can focus on tackling the non-technical challenges of solution maintenance which requires additional collaboration with domain experts. Access to the domain expertise is a complicated challenge since it requires access to a financial institution. Thus the majority of publicly available research is conducted without the involvement of the experts or even on the data not related to any financial institution (like synthetic data).

Future work can also focus on enhancing the presented alert interpretability module, for example, by experimenting with a counterfactual explanations approach. This approach allows extending existing alert explanations with information on which characteristics of customers' behavior should change in order for an ML model to provide an opposite decision. For example, the counterfactual explanations approach could provide the following explanation: "To prevent a customer from being flagged as a potential money launderer, the customer could have made smaller deposits over a longer period of time, rather than a large lump sum in a single transaction." Additionally, the interpretability module can be improved by using the power of large language models, taking into account previous cases documented in the corporate knowledge base. Finally, the solution for group MoL patterns detection presented in Chapter 6 was not complemented with correspondent explanations attached to alerted groups of individuals. Future work can focus on the development of the interpretability module extension to accompany detected group MoL patterns with explanations.

Future work could also focus on improving the system for group ML patterns detection by experimenting with other algorithms and approaches for both community detection and graph-based statistics feature extraction. For example, Louvain algorithm, Random Walks on graphs or Graph Neural Networks could be studied as alternative community detection approaches.

One of the limitations of the present research work comes from the fact that the empirical evaluation of alert interpretability was based on a reduced number of alerts, and more data would be needed to draw more generalized conclusions. Future work can focus on conducting a more in-depth evaluation of user acceptance in a daily usage setting.

The present research is focused on the detection of MoL and the investigation of detected potential MoL cases was considered to be out of scope. When investigating MoL alerts, AML domain experts often request documents from customers to prove the legitimacy of their income or make an investigation using public data and social networks. The presented solution does not directly address the question of supporting the investigation phase (see Fig. 3). A challenge in this setting is that the data used in an investigation phase is usually highly unstructured, consisting of text without a fixed vocabulary and sometimes involving complex legal and commercial language, documents in raster format (e.g. scanned documents), and complex tables (e.g. complex Excel sheets). Future work can focus on the development of an object character recognition module to process documents for gathering signals of potential MoL as well as focus on incorporating more data sources such as social media. This can enable the creation of more robust models that can detect MoL activities across different channels.

BIBLIOGRAPHY

[AB16] Claudio Alexandre and João Balsa. “Integrating client profiling in an anti-money laundering multi-agent based system”. In: *New Advances in Information Systems and Technologies*. Vol. 444. Advances in Intelligent Systems and Computing. Springer, 2016, pp. 931–941.

[ABB20] Muhammad Abulaish, Ishfaq Majid Bhat, and Sajid Yousuf Bhat. “Scaling density-based community detection to large-scale social networks via MapReduce framework”. In: *Journal of Intelligent & Fuzzy Systems* 38.2 (2020), pp. 1663–1674.

[Ahm21] A Ahmed. “Anti-money laundering recognition through the gradient boosting classifier”. In: *Academy of Accountingand Financial Studies Journal* 25.5 (2021).

[Ali] Vagif Aliyev. *Ethereum Fraud Detection Dataset*. URL: `https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset`.

[Alo+22] Johrha Alotibi et al. “Money Laundering Detection using Machine Learning and Deep Learning”. In: *International Journal of Advanced Computer Science and Applications* 13.10 (2022).

[APN20] Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. “Comparative analysis using supervised learning methods for anti-money laundering in bitcoin”. In: *Proceedings of the 2020 5th international conference on machine learning technologies*. 2020, pp. 11–17.

[AQA21] Mohannad Alkhalili, Mahmoud H Qutqut, and Fadi Almasalha. “Investigation of applying machine learning for watch-list filtering in anti-money laundering”. In: *IEEE Access* 9 (2021), pp. 18481–18496.

[Azi+22] Rabia Musheer Aziz et al. “LGBM: a machine learning approach for Ethereum fraud detection”. In: *International Journal of Information Technology* (2022), pp. 1–11.

[Bah+18] Ashwin Bahulkar et al. “Integrative analytics for detecting and disrupting transnational interdependent criminal smuggling, money, and money-laundering networks”. In: *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE. 2018, pp. 1–6.

[BLS20] Luigi Bellomarini, Eleonora Laurenza, and Emanuel Sallinger. “Rule-based Anti-Money Laundering in Financial Intelligence Units: Experience and Vision.” In: *RuleML+ RR (Supplement)*. CEUR-WS.org, 2020, pp. 133–144.

[BPI19] Andra Baltoiu, Andrei Patrascu, and Paul Irofti. "Community-level anomaly detection for anti-money laundering". In: *arXiv preprint arXiv:1910.11313* (2019).

[Bre+] Stuart Breslow et al. *The new frontier in anti–money laundering*. URL: `https://www.mckinsey.com/business-functions/risk/our-insights/the-new-frontier-in-anti-money-laundering`.

[But21] Laurie Butgereit. "Anti money laundering: Rule-based methods to identify funnel accounts". In: *2021 Conference on Information Communications Technology and Society (ICTAS)*. IEEE. 2021, pp. 21–26.

[Cha+02] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[Cha+19] Xiangrui Chao et al. "Behavior monitoring methods for trade-based money laundering integrating macro and micro prudential regulation: a case from China". In: *Technological and Economic Development of Economy* 25.6 (2019), pp. 1081–1096.

[Che+18] Zhiyuan Chen et al. "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review". In: *Knowledge and Information Systems* 57 (2018), pp. 245–285.

[Che+21] Zhiyuan Chen et al. "Variational autoencoders and Wasserstein generative adversarial networks for improving the anti-money laundering process". In: *IEEE Access* 9 (2021), pp. 83762–83785.

[CM11] Yu-To Chen and Johan Mathe. "Fuzzy computing applications for anti-money laundering and distributed storage system load monitoring". In: *World conference on soft computing (2011)* (2011).

[CT22] Tianyi Chen and Charalampos Tsourakakis. "Antibenford subgraphs: Unsupervised anomaly detection in financial networks". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2022, pp. 2762–2770.

[CYY23] Dangxing Chen, Jiahui Ye, and Weicheng Ye. "Interpretable selective learning in credit risk". In: *Research in International Business and Finance* (2023), p. 101940.

[DBC15] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. "Early classification of time series as a non myopic sequential decision making problem". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*. Springer. 2015, pp. 433–447.

[Edd+21] Ahmad Naser Eddin et al. “Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs”. In: *CoRR* abs/2112.07508 (2021).

[FP97] Tom Fawcett and Foster Provost. “Adaptive fraud detection”. In: *Data mining and knowledge discovery* 1.3 (1997), pp. 291–316.

[Gao+09] Shijia Gao et al. “Knowledge-based anti-money laundering: a software agent bank application”. In: *Journal of Knowledge Management* (2009), pp. 63–75.

[Har+22] Daniel A Harris et al. “Using real-world transaction data to identify money laundering: Leveraging traditional regression and machine learning techniques”. In: *STEM Fellowship Journal* 7.1 (2022), pp. 21–32.

[Hel+16] Tamer Hossam Helmy et al. “Design of a Monitor for Detecting Money Laundering and Terrorist Financing”. In: *Journal of Theoretical & Applied Information Technology* 85.3 (2016).

[Hev+08] Alan R Hevner et al. “Design science in information systems research”. In: *Management Information Systems Quarterly* 28.1 (2008), p. 6.

[Hsi+21] Yu-Yen Hsin et al. “Interpretable Electronic Transfer Fraud Detection with Expert Feature Constructions”. In: *CIKM Workshops*. 2021.

[ICIa] ICIJ. *Bahama Leaks*. URL: `https://www.icij.org/tags/bahamas-leaks/`.

[ICIb] ICIJ. *Pandora Papers*. URL: `https://www.icij.org/investigations/pandora-papers/`.

[ICIc] ICIJ. *The Panama Papers: Exposing the Rogue Offshore Finance Industry*. URL: `https://www.icij.org/investigations/panama-papers/`.

[Int] Transparency International. *Hiding in Plain Sight: How UK Companies are Used to Launder Corrupt Wealth*. URL: `https://www.transparency.org.uk/publications/hiding-in-plain-sight`.

[JI23a] Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. “Fighting Money Laundering With Statistics and Machine Learning”. In: *IEEE Access* 11 (2023), pp. 8889–8903.

[JI23b] Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. “Qualifying and raising anti-money laundering alarms with deep learning”. In: *Expert Systems with Applications* 214 (2023), p. 119037.

[JS19] Kristen Jaskie and Andreas Spanias. “Positive and unlabeled learning algorithms and applications: A survey”. In: *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE. 2019, pp. 1–8.

[Jul+20] Martin Jullum et al. “Detecting money laundering transactions with machine learning”. In: *Journal of Money Laundering Control* 23.1 (2020), pp. 173–186.

[Ket+21] Utku Görkem Ketenci et al. “A time-frequency based suspicious activity detection for anti-money laundering”. In: *IEEE Access* 9 (2021), pp. 59957–59967.

[KJR10] Miron B Kursa, Aleksander Jankowski, and Witold R Rudnicki. “Boruta–a system for feature selection”. In: *Fundamenta Informaticae* 101.4 (2010), pp. 271–285.

[KL12] Kyoungok Kim and Jaewook Lee. “Sequential manifold learning for efficient churn prediction”. In: *Expert systems with applications* 39.18 (2012), pp. 13328–13337.

[KSF17] Meelis Kull, Telmo Silva Filho, and Peter Flach. “Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers”. In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 623–631.

[Kut+21] Dattatray Vishnu Kute et al. “Deep learning and explainable artificial intelligence techniques applied for detecting money laundering–a critical review”. In: *IEEE Access* 9 (2021), pp. 82300–82317.

[LA12] Edgar Alonso Lopez-Rojas and Stefan Axelsson. “Money laundering detection using synthetic data”. In: *Annual workshop of the Swedish Artificial Intelligence Society (SAIS)*. Linköping University Electronic Press, Linköpings universitet. 2012.

[Leb+17] Bertrand Lebichot et al. “A graph-based, semi-supervised, credit card fraud detection system”. In: *Complex Networks & Their Applications V: Proceedings of the 5th International Workshop on Complex Networks and their Applications*. Springer. 2017, pp. 721–733.

[Leo+13] Anna Leontjeva et al. “Early security classification of skype users via machine learning”. In: *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*. 2013, pp. 35–44.

[LG22] Kang Lin and Yuzhuo Gao. “Model interpretability of financial fraud detection by group SHAP”. In: *Expert Systems with Applications* 210 (2022), p. 118354.

[Li+20] Xiangfeng Li et al. “Flowscope: Spotting money laundering based on graphs”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 4731–4738.

[Li+21] Zhao Li et al. “What happens behind the scene? Towards fraud community detection in e-Commerce from online to offline”. In: *Companion Proceedings of the Web Conference 2021*. 2021, pp. 105–113.

[Lin+16] Xiao Ling et al. “Fast community detection in large weighted networks using graphx in the cloud”. In: *2016 IEEE 18th International Conference on High Performance Computing and Com-*

munications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE. 2016, pp. 1–8.

[Liu+22] Fanzhen Liu et al. "eRiskCom: an e-commerce risky community detection platform". In: *The VLDB Journal* 31.5 (2022), pp. 1085–1101.

[LL17] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[Lok22] Mark E Lokanan. "Predicting Money Laundering Using Machine Learning and Artificial Neural Networks Algorithms in Banks". In: *Journal of Applied Security Research* (2022), pp. 1–25.

[MA18] Aji Mubarek Mubalaike and Esref Adali. "Deep learning approach for intelligent financial fraud detection system". In: *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. IEEE. 2018, pp. 598–603.

[Mag+18] Shamil Magomedov et al. "Anomaly detection with machine learning and graph databases in fraud management". In: *International Journal of Advanced Computer Science and Applications* 9.11 (2018).

[MDR15] Pritheega Magalingam, Stephen Davis, and Asha Rao. "Using shortest path to discover criminal community". In: *Digital Investigation* 15 (2015), pp. 1–17.

[MGS21] Maryam Mahootiha, Alireza Hashemi Golpayegani, and Babak Sadeghian. "Designing a new method for detecting money laundering based on social network analysis". In: *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*. IEEE. 2021, pp. 1–7.

[OCC] OCCRP. *The Troika Laundromat*. URL: `https://www.occrp.org/en/troikalaundromat/`.

[PA14] Girish Keshav Palshikar and Manoj Apte. "Financial security against money laundering: A survey". In: *Emerging trends in ICT security*. Elsevier, 2014, pp. 577–590.

[PA22] Eric Pettersson Ruiz and Jannis Angelis. "Combating money laundering with machine learning–applicability of supervised-learning algorithms at cryptocurrency exchanges". In: *Journal of Money Laundering Control* 25.4 (2022), pp. 766–778.

[PDG18] Mario Alfonso Prado-Romero, Christian Doerr, and Andrés Gago-Alonso. "Discovering bitcoin mixing using anomaly detection". In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 22nd Iberoamerican Congress, CIARP 2017, Valparaíso, Chile, November 7–10, 2017, Proceedings 22*. Springer. 2018, pp. 534–541.

[Pro+18] Liudmila Prokhorenkova et al. “CatBoost: unbiased boosting with categorical features”. In: *Advances in neural information processing systems* 31 (2018).

[Rai21] Omri Raiter. “Applying supervised machine learning algorithms for fraud detection in anti-money laundering”. In: *Journal of Modern Issues in Business Research* 1.1 (2021), pp. 14–26.

[RAK07] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In: *Physical review E* 76.3 (2007), p. 036106.

[RS18] David Robinson and Chris Scogings. “The detection of criminal groups in real-world fused data: using the graph-mining algorithm “GraphExtract””. In: *Security Informatics* 7.1 (2018), p. 2.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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.

[SAJ20] Theyvaa Sangkaran, Azween Abdullah, and NZ Jhanjhi. “Criminal community detection based on isomorphic subgraph analytics”. In: *Open Computer Science* 10.1 (2020), pp. 164–174.

[Sav+17] David Savage et al. “Detection of money laundering groups: Supervised learning on small networks”. In: *The Workshops of the The Thirty-First AAAI Conference on Artificial Intelligence, Saturday, February 4-9, 2017, San Francisco, California, USA*. Vol. WS-17. AAAI Technical Report. AAAI Press, 2017.

[Sch96] Nancy L Schwalje. *Lords of the Rim: the invisible empire of the overseas Chinese*. 1996.

[Sen+95] Ted E Senator et al. “Financial crimes enforcement network AI system (FAIS) identifying potential money laundering from reports of large cash transactions”. In: *AI magazine* 16.4 (1995), pp. 21–21.

[SG22] Tanmay Srinath and HS Gururaja. “Explainable machine learning in identifying credit card defaulters”. In: *Global Transitions Proceedings* 3.1 (2022), pp. 119–126.

[Sha53] Lloyd S Shapley. “A Value for n-Person Games”. In: *Contributions to the Theory of Games II*. Princeton University Press, 1953, pp. 307–317.

[She13] Robert P Sheridan. “Time-split cross-validation as a method for estimating the goodness of prospective prediction.” In: *Journal of chemical information and modeling* 53.4 (2013), pp. 783–790.

[Sip+14] Ruben Sipos et al. “Log-based predictive maintenance”. In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*. Association for Computing Machinery, 2014, pp. 1867–1876.

[SN18] Abdul K Shaikh and Amril Nazir. “A model for identifying relationships of suspicious customers in money laundering using social network functions”. In: *Proceedings of the world congress on engineering*. Vol. 1. 2018, pp. 4–7.

[Sol+16] Reza Soltani et al. “A new algorithm for money laundering detection based on structural similarity”. In: *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE. 2016, pp. 1–7.

[Sta+21] Michele Starnini et al. “Smurf-based anti-money laundering in time-evolving transaction networks”. In: *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV 21*. Springer. 2021, pp. 171–186.

[TK19] Chih-Hua Tai and Tai-Jung Kan. “Identifying money laundering accounts”. In: *2019 International Conference on System Science and Engineering (ICSSE)*. IEEE. 2019, pp. 379–382.

[Ung17] Brigitte Unger. “Offshore activities and money laundering: recent findings and challenges”. In: (2017).

[UNO] UNODC. *Money Laundering Overview*. URL: `https://www.unodc.org/unodc/es/money-laundering/overview.html`.

[VBV17] Andrey Volkov, Dries F Benoit, and Dirk Van den Poel. “Incorporating sequential information in bankruptcy prediction with predictors based on Markov for discrimination”. In: *Decision Support Systems* 98 (2017), pp. 59–68.

[Wan+23] Dan Wang et al. “A sparsity algorithm for finding optimal counterfactual explanations: Application to corporate credit rating”. In: *Research in International Business and Finance* (2023), p. 101869.

[Web+] Mark Weber et al. *Elliptic Data Set*. URL: `https://www.kaggle.com/datasets/ellipticco/elliptic-data-set`.

[Wil72] Dennis L Wilson. “Asymptotic properties of nearest neighbor rules using edited data”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 3 (1972), pp. 408–421.

[WW21] Tung-Yu Wu and You-Ting Wang. “Locally interpretable one-class anomaly detection for credit card fraud detection”. In: *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE. 2021, pp. 25–30.

[WY07] Su-Nan Wang and Jian-Gang Yang. “A money laundering risk evaluation method based on decision tree”. In: *2007 international conference on machine learning and cybernetics*. Vol. 1. IEEE. 2007, pp. 283–286.

[Xue+21] Xie Xueshuo et al. “AWAP: Adaptive weighted attribute propagation enhanced community detection model for bitcoin de-anonymization”. In: *Applied Soft Computing* 109 (2021), p. 107507.

[Youa] Ernst Young. *AML Transaction Monitoring - 2020 Nordic Survey Report. Ernst Young.* URL: https://assets.ey.com/content/dam/ey-sites/ey-com/da_dk/topics/banking-and-capital-markets/ey-aml-transaction-monitoring-nordics-survey-2020.pdf.

[Youb] Ernst Young. *Anti-money laundering Transaction Monitoring. Ernst Young. 2018 EMEIA Survey Report.* URL: https://assets.ey.com/content/dam/ey-sites/ey-com/en_gl/topics/emeia-financial-services/ey-anti-money-laundering-aml-transaction-monitoring.pdf.

[Zha+16] Ke Zhang et al. “Automated IT system failure prediction: A deep learning approach”. In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 1291–1300.

ACKNOWLEDGEMENTS

To my loving family, whose unwavering support and encouragement have been the foundation of my journey. Your belief in me has fueled my determination, and I am forever grateful for your boundless love and understanding.

To my devoted wife, my pillar of strength, whose unwavering belief in my abilities has been my greatest motivation. Your patience, understanding, and support have been the backbone of my success. Your words of encouragement, late-night conversations, and laughter-filled moments have provided much-needed respite in the midst of intellectual challenges.

To my incredible friends, your constant presence and uplifting spirits have made this challenging path more enjoyable. Thank you for being there through the highs and lows, sharing in my triumphs, and reminding me to celebrate life beyond academia.

To my exceptional supervisor, whose guidance, expertise, and unwavering commitment to my intellectual growth have been invaluable. You have nurtured my ideas, challenged my perspectives, and inspired me to push the boundaries of knowledge. Your mentorship has shaped not only my research but also my character. I am honored to have had the opportunity to learn from you.

To the esteemed University of Tartu, a beacon of knowledge and innovation. I am humbled and grateful for the platform you have provided me to explore my intellectual curiosity. The exceptional resources, facilities, and intellectual community have nurtured my growth and enriched my academic journey.

This thesis is dedicated to all of you, for your support, encouragement, and belief in my abilities. Thank you for being with me throughout this endeavor.

SISUKOKKUVÕTE

Masinõppemeetodid rahapesu tõkestamise jälgimiseks

Rahapesu (RP) kujutab endast märkimisväärset ohtu ülemaailmsetele finantssüsteemidele, võimaldades kurjategijatel varjata raha ebaseaduslikku päritolu ja integreerida seda seaduslikku majandusse. Rahapesul ei ole mitte ainult finantsilised tagajärjed, vaid see õõnestab ka finantssüsteemide stabiilsust, ohustab riiklikku julgeolekut ja kahandab üldsuse usaldust finantsinstitutsioonide vastu. Valitsused ja õiguskaitseasutused kogu maailmas on mures ebaseaduslike tegevuste tuvastamise ja tõkestamise pärast, kuna rahapesu moodustab olulise osa ülemaailmsest sisemajanduse kogutoodangust (SKT), võimaldades kriminaaltulu, terrorismi rahastamist ja maksudest kõrvalehoidumist. Finantsasutused omakorda kasutavad seiremehhanisme, et tuvastada võimalikke rahapesuga seotud tegevusi ja nende osas teavitusi anda. Traditsiooniline lähenemine selliste seiremehhanismidele loomisel on reeglipõhiste süsteemide kasutamine, mida tavaliselt rakendatakse nende lihtsuse ja tõlgendatavuse tõttu, kuid millel on puudusi, eriti keeruliste ja esilekerkivate RP-skeemide tuvastamisel. Reeglipõhiste süsteemide täiendamine masinõppe algoritmidega võib aidata leida keerulisi suhteid ning parandada avastamise ja ennetamise tõhusust.

Selle lõputöö eesmärk on luua raamistike komplekt, mis kombineerituna pakuvad terviklikku lahendust automaatseks RP tuvastamiseks masinõppe algoritme kasutades. Masinõppetehnikaid kasutava lahenduse väljatöötamist raskendavad mitmed väljakutsed, mis tulenevad RP nähtuse ja selle tuvastamiseks saadaolevate andmete olemusest. Praegused samaaegsed arengud, liikumisel sularahavaba ühiskonna poole koos globaliseerumisega on toonud kaasa digitaalsete maksete märkimisväärse kasvu, mille tulemusel on loodud tohutu hulk andmeid, mida on vaja RP tuvastamiseks analüüsida. Finantstoodete mitmekesisus muudab rahapesu tuvastamise veelgi keerulisemaks, ja ebaseaduslike tegevuste jaoks saab samuti nüüd kasutada erinevaid kanaleid. Lisaks on RP haruldane ja tahtlikult varjatud nähtus, mistõttu on andmete tasakaalustamatuse ja keerukate tuvastamistehnikate vajaduse tõttu masinõppemudelite treeniminekeeruline. Lisaks nõuab RP-skeemide pidevalt arenev olemus masinõppemudelite regulaarset värskendamist ja ümberõpet, et olla kursis uute mustrite ja funktsioonidega.

See väitekiri annab järgmised neli peamist panust uurimisvaldkonda. Esimene panus on raamistik üksikute RP-mustrite tuvastamiseks. Raamistik keskendub mastaapsusele ja tasakaalustamatuse vastupanuvõimekusele, kasutades kahekihilist arhitektuuri. Kavandatav raamistik kasutab kahekihilist arhitektuuri rakendades logistilist regressiooni ja äärmuslike gradientide võimenduspuude klassifikaatoreid, mis filtreerivad esimeses kihis välja mitteillegaalsed kliendid ja klassifitseerivad kõrgendatud riskiga kliendid teises kihis. Näidati, et kahekihiline arhitektuur tõi kasu lahenduse mastaapsuse ja töökindluse nõuetele vastavuse saavutamisel.

Teine panus on seiresüsteem, mis on loodud ja välja töötatud hoiatama potentsiaalselt ebaseaduslike RP-käitumiste tuvastamisel. Süsteem genereerib täpseid, mitte-üleliigseid ja õigeaegseid hoiatusi ning kasutab mõõdikute perekonda, mis erinevalt tavalistest masinõppemõõdikutest fikseerivad aja aspekti.

Kolmas panus on Shapley väärtuste algoritmil põhinev tõlgendatavuse moodul, mis pakub esile tõstetud hoiatustele tekstilisi selgitusi. See aitab paremini mõista, miks hoiatusi genereeritakse, ja juhib uurimisprotsessi. Esitatud hoiatuste jaoks loodud selgituste kasulikkust testisid erinevate jurisdiktsioonide domeenieksperdid mitmeetapiliseuurimissessiooni käigus.

Neljas panus on raamistik grupi RP-käitumise tuvastamiseks. Raamistik kasutab tehinguandmeid osalise sotsiaalse võrgustiku loomiseks ja grupi ebaseadusliku käitumise RP-mustrite tuvastamiseks, eeldamata, eelnevalt tuvastatud RP-mustrite olemasolu või seda, et kõik finantstehingud on finantsasutuse piires nähtavad. Raamistiku skaleeritavus saavutatakse lineaarse keerukusega kogukonna tuvastamise algoritmide ja paralleelse arvutusliku Apache Spark raamistiku kasutamisega.

Kokkuvõttes moodustavad esitatud kaastööd tervikliku lahenduse automaatseks RP tuvastamiseks, mis on loodud vastama järgmistele ärinõuetele: (i) täpsus – lahendus peab pakkuma täpseid väljundeid; ii) robustsus – lahendus peab käsitlema laia valikut RP mustri klasse; (iii) õigeaegsus – lahendus peab andma oma väljundi õigel ajal; (iv) kasutatavus – lahenduse väljundeid peab olema võimalik inimestel analüüsida; (v) tõlgendatavus – lahendus peab andma tõlgendatavaid väljundeid; ja (vi) skaleeritavus – lahendus peaks olema skaleeritav käsitletavate finantsandmete hulga suurenemise puhul. Kõiki nelja panust on testitud tegelikus suuremahulises andmekogus, mis sisaldab kliendiprofiile, tehingute ajalugu ja silte, mille on andnud rahapesuvastased eksperdid kolmest erinevast jurisdiktsioonist. Tulemusi hinnati ajalooliste andmete arvutuslike katsete ja finantsasutuse domeeniekspertide interaktiivse tagasiside abil.

CURRICULUM VITAE

Personal data

Name:	Pavlo Tertychnyi
Date of Birth:	14.09.1994
Citizenship:	Ukrainian
Language:	Ukrainian, English

Education

2018–2023	doctor of philosophy in computer science candidate – University of Tartu
2016–2018	master’s degree in computer science – University of Tartu
2012–2016	bachelor’s degree in applied mathematics - Taras Shevchenko Kyiv National University

Employment

2022–present day	Data Scientist - Wise
2019–2022	Data Scientist - Swedbank
2018–2019	Junior Research Fellow of Data Science - University of Tartu
2017–2018	Data Science Intern - GEYCE Biometrics

Scientific work

Main fields of interest:

- automation of detection and prevention of financial crime (money laundering, fraud, card fraud)

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Pavlo Tertychnyi
Sünniaeg:	14.09.1994
Kodakondsus:	Ukrainlane
Keelteoskus:	ukrainlane, inglise

Haridus

2018–2023	filosoofiadoktori kandidaat arvutiteaduses – Tartu Ülikooli
2016–2018	magistrikraad arvutiteaduses – Tartu Ülikooli
2012–2016	bakalaureusekraad rakendus matemaatikas - Taras Shevchenko Kiievi Riiklik Ülikool

Teenistuskäik

2022–tänapäev	Andmeteadlane - Wise
2019–2022	Andmeteadlane - Swedbank
2018–2019	Andmeteaduse nooremteadur - Tartu Ülikooli
2017–2018	Andmeteadlase praktikant - GEYCE Biometrics

Teadustegevus

Peamised uurimisvaldkonnad:

- finantskuritegude (rahapesu, pettused, kaardipettused) avastamise ja ennetamise automatiseerimine

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

1. Pavlo Tertychnyi, Ivan Slobozhan, Madis Ollikainen, and Marlon Dumas. "Scalable and imbalance-resistant machine learning models for anti-money laundering: A two-layered approach." In: *Enterprise Applications, Markets and Services in the Finance Industry: 10th International Workshop, FinanceCom 2020, Helsinki, Finland, August 18, 2020, Revised Selected Papers 10*. Springer. 2020, pp. 43-58.

   Author contributions: Lead author. Conceived the idea, contributed to the study design, conducted data collection and analysis, and wrote the manuscript with support of the other authors.

2. Pavlo Tertychnyi, Mariia Godgildieva, Marlon Dumas, and Madis Ollikainen. "Time-aware and interpretable predictive monitoring system for Anti-Money Laundering". In: *Machine Learning with Applications* 8 (2022), p. 100306.

   Author contributions: Lead author. Conceived the idea, contributed to the study design, conducted data collection and analysis, and wrote the manuscript with support of the other authors.

3. Pavlo Tertychnyi, Tommy Lindström, Changling Liu, and Marlon Dumas. "Detecting Group Behavior for Anti-Money Laundering With Incomplete Network Information". In: *2022 IEEE International Conference on Big Data (Big Data)*. IEEE. 2022, pp. 2383–2388.

   Author contributions: Lead author. Conceived the idea, contributed to the study design, conducted data collection and analysis, and wrote the manuscript with support of the other authors.

Publications not included in the thesis

1. Pavlo Tertychnyi, Cagri Ozcinar, and Gholamreza Anbarjafari. "Low-quality fingerprint classification using deep neural network". In: *IET Biometrics* 7.6 (2018), pp. 550–556
2. Dogus Karabulut, Pavlo Tertychnyi, Hasan Sait Arslan, Cagri Ozcinar, Kamal Nasrollahi, Joan Valls, Joan Vilaseca, Thomas B. Moeslund, and Gholamreza Anbarjafari. "Cycle-consistent generative adversarial neural networks based low quality fingerprint enhancement". In: *Multimedia Tools and Applications* 79 (2020), pp. 18569–18589.

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS