NETWORK-BASED METHODS FOR ANTI-MONEY LAUNDERING

by

Gizem Taş

B.S., Industrial Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

Above all, I wish to express my gratitude to Professor Tınaz Ekim Aşıcı for being my thesis advisor and guiding me throughout this study tirelessly with kindship and fellowship. I am indebted to her for agreeing to be my thesis advisor even when our acquaintance was fresh. From the moment I met her in person, she has been a role model for me for her genius and enthusiasm for science. This study, by its very nature, branches out to so many research questions that there were times I felt lost in detail; she was there to light my way at every turn. I would like to emphasize my feelings for her for a particular period of time towards the end, when all of us struggled during the time of pandemic, she was still available online as it were face-to-face, and I cannot thank her enough for that opportunity.

I would like to thank Tolga Kurt and Utku Görkem Ketenci from H3M, at first for coming up with such a brilliant approach to such an interesting research question, and then for cooperating willingly throughout the entire study. Their expertise in anti-money laundering and machine learning techniques has backed me up all the time especially for remaining within the scope of the problem. I owe a debt of special thanks to Utku Abi for our brainstorming phases and pleasant conversations, without which this work would not have been possibly achieved its goals.

This study is supported by the 1501 Industrial R&D Projects Grant Programme of Scientific and Technological Research Council of Turkey (TÜBİTAK) as part of the project number 3181341 titled "Artificial Intelligence-Based Anti-Money Laundering Software". I am also grateful to the funding received through the 2210-A National Scholarship Programme for MSc Students of TÜBİTAK. The financial contribution and recognition of TÜBİTAK is truly appreciated for elevating this project up to international level. I would like to mention gladly that currently the project is pending to be invested by the European Union Research and Innovation Programme Horizon 2020 with international partners.

I am much obliged to our professors at the Department of Industrial Engineering for their unconditional support both as superiors and fellow colleagues in recent years. I wish to express my sincere appreciation to Professor Taner Bilgiç, head of our department, for caring about me so much during difficult pandemic times. Besides, I had the chance to learn from him while he was the advisor of our undergraduate final project, ever since he impresses me with his politeness and his meticulous temperament.

I would like to thank Professor Gönenç Yücel especially, both for being my ex-thesis advisor and introducing me to the curious world of networks by means of his "Agent-Based Modeling and Simulation" course. I am indebted to him for his understanding at the times I had the chance to work alongside him as a teaching assistant. He inspired me greatly with his sharpness and efficiency.

I wish to state my admiration for Professor Yaman Barlas, for all the interesting and insightful conversations we had either in his office, in SESDYN or wherever we came across. In fact, the first one of those conversations the reason why I chose to study Industrial Engineering at Boğaziçi University seven years ago. I could not thank him enough for teaching me how to see the world from a modeler's perspective and how to seize the source of problems. I feel very lucky to have witnessed him teaching.

I should not skip thanking Professor İlker Birbil, for contributing to this study greatly by means of his winter school on machine learning. I am grateful to him for giving me the opportunity to enhance my knowledge while learning from him.

I would like to thank my parents Nilgün and Ömer, and my younger sister Yağmur for giving me just the support I need every time, even remotely. Even though I have been living away from home since age fourteen, there has not passed a single day they let me feel neglected or lonely.

It would be foolish to skip thanking my fellow colleagues: Tarkan, Feyyaz, Selin, Çiğdem, Buğra, Can, Orkun and Elif. It has been a blessing working with them as

much as it has been while studying Industrial Engineering with them. I would look forward to our long lunch or coffee conversations. I value their friendship extremely and feel very fortunate to have crossed paths with them. I owe Tarkan a special thanks for his trustworthiness and for bearing a hand to me unconditionally all the time.

I am beyond grateful for Burçak and Mina's friendship which made the most difficult times both emotionally and physically endurable for me. I am falling short of words how amazed I am with their compassion, wisdom and frankness. Thanks to them, I know whom to appeal when I need soothing or celebrating.

I wish to acknowledge the great support of Begüm, or as her entourage calls her, Zehra. She has been by my side since we were sleeping side by side at our high school dormitory. She is the perfect companion to laugh with and a shoulder to cry on. I am looking forward to many priceless memories with her in Europe.

I would like to express my thankfulness to Ece, for welcoming me warmly in her family house since we were children, and for building such a supportive yet comfortable relationship between us. I would definitely not miss thanking Süveyda and Defne for being my friends for life. I am grateful for having such unique and sophisticated people in my closest circle and our priceless memories from the times we were roommates.

Beyond all, I should recognize unexpectedly cheering spirit of "Herkes Partisi" in my life through the hardest times of pandemic. I would like to thank all fellow members for being passionate conversationers, vigorous advocates of whatsoever topic and last but not least, for being there to entertain without missing any beat.

Finally, words can't describe how much I appreciate Volkan's place in my life. I am thankful to him for encouraging me unceasingly, for accompanying me when I am racking my brains and always hitting the right notes, and his natural talent for making life enjoyable.

ABSTRACT

NETWORK-BASED METHODS FOR ANTI-MONEY LAUNDERING

This study concentrates on the contributions of network-based methods in anti-money laundering. Being responsible for reporting suspicious activities, time-consuming analyses conducted by the banks through rule-based systems result in high false positive rates. Today, machine learning methods draw attention for their ability to reduce computational cost and false positive rates in anti-money laundering. In this research, the goal is to both reduce dimension of large and complex financial networks while minimizing the loss of valuable information and identify suspiciousness by feeding machine learning algorithms by network-based features using bank transaction networks. Dimensional reduction of networks in the first place enhanced the efficiency of following subgraph construction algorithms. Based on the hypothesis that money launderers disguise organizedly in licit activities, hence network relations bear the trace of hidden suspects, candidate subgraphs are constructed. Network-based features of each subgraph such as inside transaction volume, amount of outgoing transactions, size and so forth are calculated and these subgraphs are labeled as innocent or suspicious according to presumed risk aversion behaviors of banks. Created candidate subgraphs data set is used for training supervised classifiers and the evaluations on separate test data sets indicate that a satisfactory discriminative performance is obtained. These outcomes support the initial hypothesis that network-based analysis in anti-money laundering would contribute more compared to accustomed rule-based systems or individual analysis methods. These findings are promising for declining false positive rate and easing the burden on labor force in the event of a prospective implementation of the suggested system on real data to be acquired from banks.

ÖZET

KARA PARA AKLAMAYLA MÜCADELE İÇİN AĞ TEMELLİ YÖNTEMLER

Bu çalışmada, kara para aklama ile mücadelede ağ temelli yöntemlerin katkıları üzerinde durulmuştur. Şüpheli işlem bildirimi yapmakla yükümlü olan bankaların uzun çalışma saatleri gerektiren kurala dayalı sistemlerle yapılan analizleri yüksek yanlış olumlu oranları ile sonuçlanmaktadır. Günümüzde, makine öğrenmesi tekniklerinin kara para aklamayla mücadelede hesaplama maliyetini ve bahsedilen yanlış olumlu oranını düşürebilme kabiliyetleri dikkat çekmektedir. Bu araştırmada, banka işlem verilerini ağ yapısında ele alarak, hem değerli bilgi kaybını en azda tutmaya çalışarak büyük ve karmaşık finansal ağlarda boyut indirgeyebilmek, hem de makine öğrenmesi algoritmalarını ağ bazlı niteliklerle besleyerek şüphelilik sınıflandırması yapmak amaçlanmıştır. İlk aşamada ağ boyutu indirgemenin, takip eden alt çizge oluşturma algoritmalarının verimini artırdığı gözlenmiştir. Kara para aklayıcıların organize bir şekilde meşru faaliyetler içinde gizlendikleri, dolayısıyla ağ yapısındaki ilişkilerin gizlenmiş şüphelilere dair iz taşıdığı hipotezine dayanarak aday alt çizgeler oluşturulmuştur. Her alt çizgenin iç işlem hacmi, dışarıya çıkan para miktarı, ağ boyutu ve benzeri ağ bazlı özellikleri hesaplanmış olup bu alt çizgeler, bankaların risk karşısında muhtemel duruşlarını temsil eden varsayımlar dahilinde, şüpheli veya masum olarak etiketlenmişlerdir. Oluşturulan aday veri seti güdümlü sınıflandırıcıları eğitmek için kullanıldığında, ayrılan test veri setindeki değerlendirmeler tatmin edici bir ayrıştırma performansının elde edildiğini göstermektedir. Bu sonuçlar, bu çalışmanın başlangıcında ortaya koyulan kara para aklamayla mücadelede ağ bazlı analizin katkısının alışılagelmiş kurala veya bireysel keşfe dayalı yöntemlerden yüksek olacağı hipotezini destekler niteliktedir. Bu bulgular, önerilen sistem bankalardan alınacak gerçek veri üzerinde uygulandığında, yanlış olumlu oranının azalması ve iş gücünün yükünün hafiflemesi açısından ümit vericidir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

"Money laundering" defines the activities widely used by individuals or illegal organizations in order to legitimize the money gained through criminal activities such as human, drug or arms trafficking, illegal gambling, bribery, or tax evasion. The term is widely confused with the criminal activity itself which is used to make money; however, money laundering is the act of insinuating the money into the financial networks without being entrapped as suspicious. Furthermore, legitimized circulation of the money in financial networks pave the way for financing those criminal activities. Therefore, money laundering is by all means a profitable act for criminals for concealing the sources of the money gained through illegal ways [1].

Money laundering is recognized as a global threat to integrity, reliability and stability of financial systems and government structures by the United Nations Office on Drugs and Crime [2]. Restraining, identifying, even criminalizing money laundering pose significant problems related to security of the financial systems and increasing crime rate throughout the world. One of the most prominent criminal sources of the money laundered is narcotics in many countries [3]. Other important sources behind financing of criminal organizations are oil smuggling and illegal trafficking of drugs, diamonds, and weapons [4]. Money laundering disguises the spoils of these illicit activities using ways to legitimize them within the financial system; thus, encourages criminal bodies by rewarding them.

Modern judicial systems state that concealment of illicit sources of proceeds of crime and manipulation of those sources as legitimate, in other words money laundering, constitute a crime [5]. In today's world, the principal function of Financial Intelligence Units (FIU) is combatting money launderers. In Turkey, the entity responsible for anti-money laundering is the Financial Crimes Investigation Board (MASAK) which functions directly under the Ministry of Treasury and Finance [6]. The board undertakes the duties of conducting researches and taking industry-specific actions about de-

tection and prevention methods of money laundering, making provisions against money launderers, gathering data within the scope of combatting the financing of terrorism (CFT), and investigating suspicious transactions appointed by the Law on Prevention of Laundering Proceeds of Crime [7]. MASAK attributes responsibility to the banks in terms of delivering information unceasingly, keeping track of suspected activity reports (SAR) and identifying the suspects. The sanctions imposed by MASAK in case of violation of the liabilities include administrative fine and judicial punishment [6].

As it can be understood from the argument above mentioning money launderers' motivation provided by profitability of the proceeds of crime with low risks of being caught, they come up with different methods to conceal the spoils unceasingly. Paul Allan Schott, author of the book Reference Guide to Anti-Money Laundering and Combating the Finance of Terrorism, says that money launderers are very creative in finding new laundering methods when existing ones are overseen such that the boundaries of the universe of those methods are difficult to describe [1]. In related terminology, these methods in question are referred as "typologies".

Typologies are naturally subject to the characteristics of different countries such as economy, regime, criminalization of money laundering in the law, or financial market circumstance. Therefore, typologies diversify among countries which puts the international cooperation against money laundering to inconvenience. The Financial Action Task Force (FATF) is a prominent inter-governmental body established by G-7 countries in 1989 to leverage the combat against money laundering [8]. In the Report on Money Laundering Typologies published by FATF, some of the frequently observed typologies in member countries are referred [9]. Namely, as a quick way of transferring funds, complex wire transfer schemes are preferred by money launderers where they can cover the sources and destinations of the funds and confuse the auditors. Moreover, FATF reports on typologies focus on two particular industries where the regulations seem inconsistent: Insurance and real-estate industries appear to be exploited typically by money launderers. Despite the more established supervision of non-profit sector in some countries, misuse of these organizations with veiled purposes

play into the hands of financing of terrorism. On top of these, individuals involved in past entrusted prominent public functions (PEP) may be involved in financial crimes and corruption by means of shell companies and transactions via offshore banks in collaboration with money launderers.

According to the research report released by United Nations Office on Drugs and Crime in 2011, the annual amount of money laundered throughout the world corresponds to 3.6% of the global GDP [2]. The report also informs that apparently, contemporary methods are able to detect only around 0.2% of the proceeds of the crime laundered through the global financial system [10]. These reported detection percentages unfold the insufficiency of traditional efforts against money laundering; speaking of which, most of the detection methods used by banks today consist of rule-based systems. Rule-based detection fails to work efficiently as the creativity of money launderers mentioned in the introduction give the clue. Being aware of the rules, suspects are able to embed their money laundering activities in the financial system a way not to be noticed. The knowledge of the rules by the criminals causes missing suspicious transactions to a large extent while rendering it very challenging to formulate indistinguishable rules. Existing detection methods are also problematic in the sense that they substantially depend on human workforce and the high rate of cleared suspects, referred as false positives. These time-consuming analyses are usually manually performed and do not function as efficiently as needed.

Globally, the combat against financial crime is placed a premium so that many political scientists, legal experts, policymakers, financiers, data scientists gather to contribute to interdisciplinary studies addressing the issue of anti-money laundering. Aside from policy and strategy related aspects of the subject matter, the essence of algorithmic solution endeavors primarily lies in the financial customer data. In some cases, the data is not more than a transactional network containing only a few aspects of customers and asset transfers in between; in others, more attributes related to the customer accounts can be put on the table. The data on hand is usually in the form of a graph where the vertices represent individual or grouped bank accounts, and the edges

stand for the transactions realized between those accounts. Here, potential efficacy of employing network-based methods in the data should not be overlooked.

There are two challenging aspects regarding the structure and accessibility of data in the scope of this research. Generally, the responsibility of detecting suspicious accounts and transactions belongs to the banks, therefore the real data can be accessed through banks. Because of the ethical codes, banks attribute confidentiality to necessary data which certainly closes the doors for open access. In some network-based researches, using open access data such as social networks can be perfectly appropriate; however, financial networks involving money launderers constitute a distinct case and accurate networks should be generated if the real ones are not accessible.

On the other hand, the structural challenge regarding the data reside in the fact that financial networks constitute very large networks that are difficult to deal with. Furthermore, relative presence of noticed money launderers in the networks is very scarce which makes the training of learning tools harder. For instance, a raw transactional network provided by a bank includes up to 10 million nodes corresponding to accounts, and each transaction realized between these nodes corresponding to edges. The resulted size renders the network inconceivable, while activities such as plotting become vain in terms of providing a deeper understanding of the network. Therefore, preprocessing of the data in terms of its reduction down to a manageable size, or an approach that would be beneficial to perceive such a large network becomes inevitable.

Departing from the specific typologies, it can be understood that money launderers manage to hide within transactional networks by means of various methods. Disaggregating high amounts of money into smaller parts and gathering it finally in one account is an example of hiding within a tree-shaped structure inside the network. Circulating the assets among several accounts without creating a net change in flows is another example to disguise the ownership of those assets. From the perspective of traditional methods, such launderer behaviors are not subject to the risk of being caught since most analyses are isolated while focusing on catching suspicious individuals.

The hypothesis of this research is that laundering behaviors manifest themselves when they are analyzed from a network-based perspective, leaving the isolated analyses aside. The main purpose is to maximize detection of suspicious launderer activity while reducing the efforts by minimizing the false positives among the reported suspicious activities. Recalling the sanctions imposed by MASAK and the burden on analysts arising from high false positive rates, it is straightforward that this study aims to facilitate the combat against money launderers particularly by eliminating false positives. For this purpose, we claim that combination of machine learning techniques with network-based analysis is rewarding in terms of network-based suspicious activity detection. To put it another way, our objective is to make use of network-based characteristics in training machine learning algorithms for accurately classifying subgraphs which shelter suspicious financial activities. In the end of this study, we provide a classification methodology to detect suchlike subgraphs while discriminating between money laundering networks and innocent ones.

In the first place, we perform edge and node filtering sequentially for complexity reduction in large transaction networks through which we are able to preserve valuable information while managing to simplify the network effectively down to nearly half its initial size. Then, we introduce a novel subgraph extraction framework addressing both typology specific and local neighborhood search. Here, we relax the traditional definition of communities being dense subgroups, thus we are able to build meaningful subgraphs that cover a broader range of suspicious interactions, which would have been missed via usual community detection techniques. We perform typology specific extraction by building subgraphs of strongly connected components in order to catch circulating behavior. As regards to the latter, we make use of 2-step-neighborhoods for constructing ego networks of seed accounts. According to thresholds implied by the scenarios we developed assuming different attitudes of banks towards taking risk, we label the extracted subgraphs. Among the most significant findings of this study, resides the distinctive ability of several network-based features. Through an analysis of the relationship between suspicious subgraphs and network-based features, contributing potential of these characteristics comes to light. We formulate the problem of suspi-

cious subgraph identification as a binary supervised classification problem for which we choose to train random forest, XGBoost and LightGBM classifiers among which XGBoost noses out. Eventually, we manage to obtain a satisfactory predictive performance under different scenarios indicating a remarkable decrease in false positive rates, where our methodology proves most advantageous in cases when banks set a higher threshold for deeming suspiciousness.

In the following sections, the related work including computational approaches in anti-money laundering research as well as network related studies is given in depth in Section 2. It is pursued by methodology in where we go over technical details of each stage in our suggested system including network preprocessing in Section 3.1, subgraph extraction in Section 3.2, calculation of network-based features in Section 3.3 and lastly machine learning in Section 3.4. Afterwards, we explain our experimental studies starting with the data preparation step in Section 4.1, followed by the display of our experimental setup leading to the results in Section 4.3. In results, we start with introducing our evaluation metrics before presenting the outcomes of each experiment, then we discuss those thoroughly in Section 4.3.3. Finally, we conclude the report in Section 5 while enunciating our main findings, the prominent challenges we faced and the possibilities of further research in anti-money laundering.

2. RELATED WORK

This section presents the up-to-date research related to various computational approaches to the problem of money laundering detection. Additionally, apart from the context of anti-money laundering, previous theoretical and applied studies regarding the technical aspects of our study are also referred in this section.

2.1. Data Driven Methods for Anti-Money Laundering

In the beginning, we introduce data driven methods including traditional rule-based systems and more recent individual analysis methods.

2.1.1. Rule based Systems for Anti-Money Laundering

By introducing a computational infrastructure for relating bank transactions automatically, Senator *et al.* [11] introduce a suspiciousness evaluation system for detection of money laundering. Based on expertise of analysts, sets of rules such as thresholding for transaction amounts, which indicate possible money laundering behavior are collected and integrated into the suspiciousness evaluation modules within the system. Underlying large database which is composed of the transactions reported by the banks, is sourced by U.S. Department of the Treasury. A plausible collaboration with the Financial Crimes Investigation Board in Turkey could lead to construction of a similar database to be practiced upon in data-driven anti-money laundering studies. Upon each update of the transactions database, the subjects which manifest activities in parallel with previously determined money laundering characteristics are evaluated as suspicious, thus nominated for further screening. Here, two apparent ramifications fall into place to emphasize once again the reasons why early rule-based detection systems are insufficient for today's anti-money laundering endeavors. Remembering that money launderers are highly fertile in originating unusual ways of washing the spoils gained from illicit activity and placing them into the regular financial system; rule-

based detection systems lack capacity in nature to discover unprecedented suspicious actions since the expertise is rooted in experience of analysts. In addition, requirement of investigation by analysts after evaluation still connotates burden of manual labor; that is addressed under the objective of reducing false positives in succeeding detection systems, as well as in this study.

As a matter of fact, aside from their dependence on know-how, premature rule-based systems have paved the way for a shift in perspective; namely from steering the suspect from transactions to the subjects or entities. Such a changeover can be endorsed while recalling the definition of money laundering itself, as it is the act of washing the dirt of the money gained through criminal ways so that the money is sneaked into the financial system as if it is clean. In this case, it should be quite sensible to bring the responsible subjects into focus of detection efforts instead of their transactions; since spoils of crime do not appear dirty while flowing inside the legitimate financial system once they are laundered.

2.1.2. Data Mining

Taking the laboriousness of fighting money laundering into account, data mining techniques have been deemed valuable in the sense of their potential in handling loaded bank data, reducing the false positive rate, hence lessening the burden on analysts [12]. While advantages of data mining techniques in recognition of money laundering patterns are widely acknowledged by financial investigators who remain inefficient in manually handling the process; there are notable limitations of integrating these technologies in current conjuncture. The productivity of data mining techniques depends highly on upfront dedication of the analysts for customization of the models, since the notion of suspiciousness in financial analysis is fairly debatable [13]. Also, the accessibility of financial data is troublesome, which has been quite a burden on this study as well, due to reluctance or lack of authorization to share data. Competence of data mining is thereby restricted since accessibility of accurate and complete data is imperious for a favorable performance [13].

2.1.3. Anomaly Detection for Anti-Money Laundering

More recent remedies for detecting money laundering from observing and evaluating bank transactions ground on use of artificial intelligence. The main challenges regarding the substantial daily data flow in a bank along with very high number of individual accounts and customers with no obvious indicator of money laundering actions smooth the way for acknowledging the inability of traditional rule-based systems in terms of fraud detection. Kingdon [14] addresses these issues by proposing an unsupervised learning approach based on a probabilistic analysis of bank data.

The suggested method tackles the concept of suspiciousness while changing the investigative point of view by seeking uncommon behavior patterns among the customers, in a sense, anomalies. Applying anomaly detection in the domain of anti-money laundering can be justified by means of the vagueness in the notion of suspiciousness [14]. By all means, this kind of methods require derivation of normal behavior expected from a bank's customers that can lead the way towards statistically definable deviance. Normality of customer behavior is deduced using a multidimensional stochastic matrix including likelihoods of actions, where search and pattern recognition techniques can be helpful in pointing out the anomalous actions. Certainly, the system is progressive compared to rule-based systems, since behavioral norms are not set the same for each bank account; instead, the accounts are grouped with their peers to externalize unusualness in a more standard manner. Also, the system is dynamic and adaptive so to include transaction data inflow. However, use of individual customer features solely renders the examination shallow and the system remains incapable to overcome cunningness of money launderers.

2.1.4. Supervised Learning

While recognizing the convenience of anomaly detection techniques when treating money laundering actions as deviations from normal financial behavior patterns; it should be noted that a thorough description of normal is essential in this paradigm.

Such a manner of handling the problem necessarily brings along the assumption that abnormal behavior is equivalent to suspicious behavior in financial content; nevertheless, the implication in question is quite presumptive for lots of up-to-date studies in AML to ground on. On the other hand, supervised learning methods step forth with their potential to provide improvement in accuracy and precision in company with disposal of normality assumptions [15]. Here, what can be perceived as a drawback is that supervised learning remains incapable of detecting unprecedented suspicious activities, as the classifiers are only trained using foreseen or flagged elements.

A combination of simple data mining-based approaches is suggested by the work of Le-Khac *et al.* [16] for analyzing money laundering patterns in an international investment bank. The authors choose to pair transaction information with account and institution information for customer identification and derivation of financial activity patterns, in a similar manner to previously described anomaly detection perspective. Such an enrichment in dimensionality of bank data is rewarding as opposed to traditional threshold setting evaluations of transactions in terms of their amounts and frequencies. For instance, the knowledge of ownership of multiple accounts or accounts that operates under one institution is integrated into mining of the transactions to be analyzed consecutively under two techniques: clustering and neural networks. In the first step, both the investors and investment funds are clustered using k-means clustering, outcomes of which are passed to the second step of the framework in order to train the neural network with suspicious and non-suspicious instances [16]. There is room for improvement in such neural network applications along with attributing risk factors derived from account information to clusters of transactions. It is asserted that classification performance of neural networks can be ameliorated in case historical course of transactions is considered more particularly than their static properties such as frequency or amount [12].

A more sophisticated application showing the practical value of radial basis function (RBF) neural network, defined as a three-layer feed-forward neural network, as opposed to time consuming support vector machines or outlier detection techniques in

combatting money laundering is proposed by Lv *et al.* [17]. To enhance the learning speed of the neural network, the authors implemented a type of one-pass clustering algorithm, called APC-III clustering, to specify the center parameters of RBF in the hidden layer, differently from conventional use of k-means clustering. Here, training set of the RBF neural network is composed of feature vectors sourced from bank database including customary account and transaction attributes such as client name, transaction amount, time or type. The study presents elevated detection rate allied with decreased false positive rates; so that RBF neural network is claimed to be a remedy for time and data related restrictions that antecedent supervised learning and anomaly detection techniques hardly tackled [17].

The computational power of artificial neural networks is accompanied by fuzzy logic algorithms for determining risk evaluation of the users in banking systems via a method called adaptive neuro-fuzzy inference system [18]. The feature vector of the users is constructed through bank transaction history and is fed into the network, similarly to the previous neural network applications. While advancing between epochs, the parameters are tuned according to least squares estimation and tuned using gradient descent while going backwards. Thusly, suggested fuzzy method is trained merely by error backpropagation, so that it grows the capability of detecting how risky the users are in the banking system. The authors argue that the system exploits the linguistic ability of fuzzy inference in the sense that the need for expertise in supervised learning is diluted [18]. Moreover, XGBoost, a rather recent supervised learning method, is chosen for binary classification of suspicious transactions in a real data acquired from a Norwegian bank by Jullum *et al.* [19]. The authors favor XGBoost library for its competence in large scaled data and its training speed. Distinctively from routine machine learning models, the training data constitutes of flagged transactions instead of users or clients, which naturally depends on the operational preference of the banks where suspicious activity reports are enlisted. Therefore this time, the predictive model takes the transactional attributes that are enlarged with background information belonging to sender and receiver such as age group, nationality, the time passed since the last bankruptcy and so forth as inputs to predict the probabilities with which transactions

need to be reported to the analysts. The study enunciates the apparent need for including known non-suspicious transactions during training phase of the model, which is critical for the model's discriminative ability between non-suspicious and suspicious transactions. For the very reason, this research also has benefited from both clear labels as it is going to be detailed further in the upcoming sections. Aside from the efficiency and favorable performance of proposed supervised learning techniques, the computational effort that needs to be put into calculation of explanatory variables to feed these models stands out as a drawback summoning new habits for storing data [19].

2.2. Graph-Based Methods for Anti-Money Laundering

In aforementioned studies, the general idea behind inference of suspiciousness of bank customers from their individual transaction behaviors bear resemblance to one another; whereas graph representation stands out as a powerful tool for reflecting the inter-dependent essence of relational data. Taking into account the relational information embedded in financial data is certainly more practical compared to conventional anomaly detection techniques in anti-money laundering. Furthermore, developing detection methods operating in relational data structures puts money launderers into inconvenience for they can shelter themselves from individual investigations through organized schemes. This section is dedicated to graph-based approaches in anti-money laundering.

2.2.1. Graph-Based Anomaly Detection

Going back to anomaly detection approach, this once in relational data, Akoğlu *et al.* [20] state that the problem of spotting anomalies in graphs can be formulated as a fully supervised classification problem in case labeled data is available; on top of that, if the available labelled data is rather scarce, semi-supervised classification while learning from both labeled and unlabeled instances is possible. The tendency of classical anomaly detection or machine learning methods to neglect the information carried by relationships among entities under the assumption that the data instances are in-

dependent from each other is cast aside using relational classification technique [20]. On the contrary, relational classification exploits the dependencies between data instances while suggesting that connected ones are prone to belong to same class or carry the same label. These methods do not turn their back on the customary node and edge attributes; instead, the model expands with neighborhood properties so as to embody both individual and relational knowledge. Simply, relational classification can be implemented in two phases in a fully or partially labeled network. Utilization of a local classifier or predictive model such as Naïve Bayes, logistic regression and so forth, for labeling a node is followed by iterative relational classification based on connections between nodes and finally a collective inference such as relaxation labeling for simultaneously classifying unlabeled nodes [21]. At this point, it makes sense to point out that the term of relational classification is synonymous with network-based collective classification and that collective inference is advantageous in the way it addresses frequently observed autocorrelation between neighbor attributes. In a sense, simultaneous inference is capable of enhancing classification performance impoverished because of omitted dependencies. Despite the inherent challenges of designation of a meaningful neighborhood structure along with linearly increasing computational expense of network feature calculations as the network grows, consideration connectivity is promising for fraud detection performance.

While giving statistical relational methods credit for their superiority over traditional methods, Bayer *et al.* [22] challenge them in terms of their inferential complexity by introducing network-based features into standard machine learning methods. The claim is also supported by alleviating the dependence on intragroup class homogeneity assumption, which translates into assignment of same class labels within neighborhoods. Bayer *et al.* [22] present the competitive advantage of transferring the relational features derived from the available network data as they were classical features in a standard machine learning model. The authors show that collective inference methods can be outperformed by their proposition of integrating direct and indirect relational features into classical models by comparing two approaches on benchmark network data. The predictive ability of proposed models proves to be competitive while nour-

ishing greatly from newly introduced indirect neighbor features. Besides, the findings of this work are particularly important, since this research has sprouted up from a similar position on the potential value of integrating network-based features in computational studies for anti-money laundering.

2.2.2. Graph Mining

Taking a step back to see the big picture, the distinction between legal and illegal financial activities would have been downhill all the way if they had subsisted separately, yet the criminal agents insinuate their way into mainstream financial system by pretending to be regular agents. Such intruders prefer to minimize the risk of being caught by blending in with the innocent [20]. The first one among the fundamental laws of money laundering specified by United Nations Office of Drug Control and Crime Prevention states that "The more successfully a money laundering apparatus is in imitating the patterns and behavior of legitimate transactions, the less the likelihood of it being exposed." [23]. Accordingly, treatment of money laundering search as an issue of anomaly detection can be upgraded by taking structural conjectures into account; such that structurally rare subgroups are identified as anomalies which disobey frequent patterns [20]. Specifically, the pursuit of such substructures can be canalized to match known money laundering typologies.

Subgraph similarity search is often perceived as the problem of searching similar subgraphs to a given query graph in a target graph, where the target graph can be massive, and the concept of graph similarity is vaguely defined. Samanvi and Sivadasan [24] approach this problem from a topological point of view and use graphlet kernels to measure similarity between two undirected and unlabeled graphs. Basically, given a query graph, the authors map the local neighborhood structure of each vertex to a vector, then define those vectors as vertex labels. Here, it should be paid attention that the term label is used in a different meaning than the rest of referenced studies. The closeness between those vertex labels is the fundamental metric used for measuring graph similarity. The general idea is that two graphs are similar to the extent the correspond-

ing neighborhood topologies of the vertices they contain are similar. The framework suggests creating a data structure by combining mapped vectors using nearest neighbor queries to identify the vertices in the large target graph that are closest to the ones in the given graph. Based this subset of seed vertices, Samanvi and Sivadasan [24] present a final match for the query graph. As much as the proposed method is open for efficiency, it still requires a search throughout an entire undirected and unlabeled graph, which hardly serves the purpose of anti-money laundering. Besides, designation of a single generic query money laundering network leaves many suspicious activities uncovered because of structurally and dimensionally diverse typologies.

Based on a similar embedding perspective, graph neural networks are utilized for the problem of graph similarity learning with benefits of topological knowledge by Li *et al.* [25]. In their work, the authors come up with two graph neural network models that can be trained so that graphs can be embedded in vector spaces, to be compared against each other using a similarity metric in vector space. The first suggestion comprises of a standard graph embedding model, where the encoder maps local neighborhood information of vertices in vectors, that are accumulated to finalize the representation of vertices through multiple layers of propagation, then accumulated by an aggregator as an embedding of the whole graph in a vector representation. When these steps are executed for both graphs, a classical similarity metric like Euclidean or cosine similarity is utilized for computing the similarity score between them. The latter method called graph matching networks functions under a similar logic except that the similarity is computed collectively during the propagation phase as opposed to independently mapping two graphs and then comparing them in standard embedding models. Graph matching networks come with a higher performance and the possibility to adjust the graph representations reciprocally in exchange for efficiency since it performs comparisons at each step instead of doing it once in the end [25]. The springboard of these solutions tackles the problem of graph similarity learning as calculation of similarity between two graphs, or given one reference graph, retrieval of its close match. In the context of anti-money laundering, these premises are not usually available for reconciling the problem with these methods. Namely, it is not quite real-

istic to appoint one structure as the ultimate money laundering network and scan an entire graph for its counterparts. If such an accurately filtered subgraph, including or sometimes centering money launderers, could have been extracted, it would have been used to find similarly structed subgraphs embedded even in the undiscovered parts of the target graph using a graph similarity metric.

In order to detect complex organized schemes, in other words money laundering networks, graph mining methods draw attention. An exemplary practice by Michalak and Korczak [26] proposes learning from graph structures while building up subgraph prototypes that possibly contain suspicious transactions. Suggested machine learning model is trained using subgraphs of labeled transactions so that subgraphs including unlabeled transactions can be matched against them; in a sense, this study incorporates subgraph matching into supervised learning. The objective is again to reduce the false positives as much as possible among the transactions reported for further investigation to the experts. Proposed graph structure learning approach is designed with adaptive ability to varying suspicious transaction structures by means of parameterization through polygonal fuzzy numbers. The authors express that the model seeks structures that resemble the graph presented in Figure 2.1 below, which can be classified as one specific money laundering typology known as smurfing or structuring. To avoid rule-based detection, dividing the money into lesser quantities to be added up later is a preferred scheme, which appears as a flow of transactions departing from source nodes proceeding towards sink nodes [27]. Since the transactional or connectivity-wise

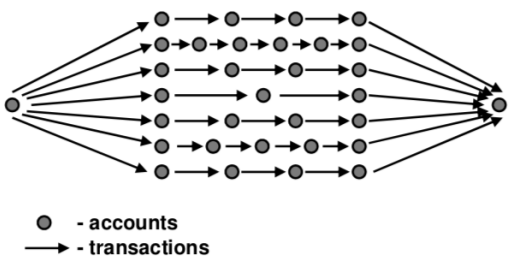


Figure 2.1. Representation of Smurfing Behavior in a Transaction Network.

attributes can vary in real money laundering networks, their fuzzy correspondence as parameters of the learning model proves to be a more realistic handling of the problem as opposed to simple motif search. Michalak and Korczak [26] highlight that fuzzy matching outweighs graph isomorphism; then again, their experimental results obtained through artificial data seem promising for applications on real bank data.

2.3. Accessibility of Financial Data

In this section, we present previous work related to possible replacement methods of inaccessible financial data.

2.3.1. Synthetic Data for Fraud Detection

One of the main and most common challenges that researchers working in financial domain face is the inaccessibility of real data sets usually due to regulations related to privacy violation of customer information. Even if banks enable data availability, there are requirements of anonymizing some features that can lead to camouflage of variable attributes. The limitation of data accessibility is by all means discouraging especially for those working in fraud detection. Curiously, the data related difficulty in question is addressed making use of agent-based simulation while avoiding legal issues [28]. Generating the needed data synthetically via simulation techniques paves the way for fraud detection research for those who do not have access to relevant data.

A simulator, called BankSim, based on multi agent-based simulation of financial payments is developed by Lopez-Rojas and Axelsson [29] using both statistical and social network analysis. Using the aggregated financial information of 6-months financial payments provided by a Spanish bank, BankSim is able to produce synthetic financial transactions along with encoded fraudulent activity generator. Breach of customer information is avoided thanks to submission of real data in aggregated form by the bank.

Lopez-Rojas and Axelsson [30] introduced another financial simulator of mobile transactions based on a real data set under the name of PaySim. They make use of multi agent-based simulation techniques as well as statistical properties of the original data set on hand. The simulator is built for fabricating transactions that mimic the authentic ones so that the synthetic data can be just as reliable for researchers. PaySim generates several different frequently observed types of transactions such as cash-in, cash-out, or money transfers. Scarcely encountered transaction types are omitted, which makes sense for the goal of this simulator is for the output to bear statistical resemblance to reality. The agent-based simulation model in the background functions with estimated variables based on acquired real data. In a way, the probabilities that an agent in the simulation takes action, that stands for carrying out a transaction in this case, are computed via existing real data. Presence of synthetic data generated via PaySim is surely practical in case of inaccessibility of real data; however, it is not sufficient for supervised learning methods since it lacks realistic labeling as fraud or normal. Therefore, in this study, PaySim is put to use confidingly for estimating parameters related to transaction amounts, which is elaborated in the following sections.

Furthermore, the problem of money laundering detection in mobile money services is once again formulated as a binary classification of transactions as either suspicious or normal in an earlier work of Lopez-Rojas and Axelsson [31]. Utilization of synthetically generated financial data to train machine learning algorithms for this purpose is thoroughly discussed. Before diving into the advantages and drawbacks of training a classifier with synthetic data, the authors do not skip mentioning that the outcomes of these classifiers should not be anticipated the same as the ones trained using real data sets. With synthetic data, the essential objective is to be able produce appropriate environments under different scenarios regarding customer profiles, rather than correctly sorting out the instances. For example, the transaction history of users from a low-income segment would apparently differ from transactions of wealthier people. Here, flexibility of simulation techniques allows testing for model considerations under different realistic scenarios, which can be counted among the benefits of using synthetic data in machine learning.

Moreover, generating the data permits control over complexity of the data, for instance by only including necessary attributes. While working with real transaction data sets for fraud detection using machine learning, the class imbalance problem steps forth as a major challenge since the fraudulent instances are relatively rare. Lopez-Rojas and Axelsson [31] suggest that simulating just enough instances belonging to both classes can cure imbalance problem. Adding synthetically generated anomalous instances in real data sets is an alternative remedy to class imbalance issue. These propositions are no doubt arguable because of their questionable validity in real world implementations. The matter relates to deciding whether the natural class distribution should be preserved or arbitrarily adjusted while sampling from a population. Although, properly modeled simulators have the capacity to avoid class mislabeling thanks to their accurate encapsulating of anomalous financial activities. In addition to all these, protection of customer privacy favors using synthetic data.

On the flipside, a classifier trained with synthetic data could lead to biased results because of their occasionally unrealistic typification. Therefore, it would be daring to rely upon the performance of algorithms trained solely with synthetic data. For amelioration of accuracy and precision scores of classifiers, real data is inevitable. Besides, prospective success of transferring the learning from synthetic data to classify newcomer real instances remains uncertain [31]. Weighing the aforementioned advantages and drawbacks ends in a stance where using synthetic data in anti-money laundering is attractive for construction of a parameterized and robust detection framework under various scenarios. When it comes to evaluation of detection performance, access to real data, even from outside financial context, seems rather preferable than simulated data.

2.3.2. Spam Detection in Online Social Networks

The aforementioned challenge regarding findability of real data for anti-money laundering pushes researchers towards different alternatives, among which social network data sets including spammer accounts shine out. As opposed to financial data, social network websites are more open for making their data available to the public.

In fact, in the field of social network analysis, detection of malicious content as well as spammer accounts is an ongoing challenge. The persistent problem is alike in many ways for both social network and financial domains; therefore, research on social networks where the availability of pertinent data is much less compelling has the potential to pave the way for anti-money laundering studies. In this study, customization of an available social network data collected for spam detection is favored over usage of synthetic data for performing supervised classification.

Fakhraei *et al.* [32] collected a data set from the social network website Tagged.com, which is designed for people to meet each other where different types of interactions between two people are enabled. Just as traditional methods for flagging suspicious accounts in banks, Tagged.com executes traditional filtering operations like responding to user reports or manual scanning for identifying spammers or malicious content. Accordingly, the challenging aspects of current situation for spam detection in social networks seem to be quite parallel with the ones faced by suspicious activity detection in financial networks. Aside from status quo resemblance, the circumstances regarding individual behavior align as well. Users of a social network join the website with various purposes leading to different behavior patterns that makes visibility of spammers ambiguous [32]. In short, given analogies between spam detection and anti-money laundering motivates this study for adaptation of an available Tagged.com website data as a transaction network.

Along with their usefulness in filling the shoes of lacking real data, spam detection research in online social networks also inspire this study in terms of their choice of graph-based features. Zheng *et al.* [33] collected data from Sina Weibo, China's most popular social network, for performing supervised learning to detect spammers in the network. As a result of applying support vector machine-based classification to Sina Weibo data, the authors provide the features they considered ranked according to their weights in the model. The highlights of the attribute analysis are average number of comments precedes average number of followees per followers which is followed by average number of users mentioned. These findings can be the guideway for determining

features indicating suspicious behavior in financial networks.

2.4. Network Reduction

Complexity reduction is necessary for rendering large transaction graphs suitable to analyze. The reduction techniques of these graphs are investigated with node and edge pruning as well as graph partitioning. Naturally, these techniques should be fine-tuned in such a way that the reduced graph preserves sufficiently valuable information for detection of money laundering groups inside. Lindner *et al.* [34] state that it is possible to protect prominent statistical and structural characteristics of graphs while reducing their size. The authors elaborate their claim such that screening out up to 80% of the complete set of edges allows for preserving network characteristics as well as enables acceleration of data mining algorithms. Network reduction techniques are mostly carried out in large social network data sets such as Facebook and they are referred as sparsification techniques, which divaricates as node and edge sparsification by filtering basis [34].

Within the context of anti-money laundering, ideally available transaction network data would involve accounts as nodes that are labeled upon suspiciousness. Under the circumstances, filtering nodes through node sparsification would imply an unpreferable loss of worthwhile information; hence, this section deliberately concentrates on edge sparsification techniques which preserve entire node set. Several solutions for properly eliminating edges in a network exist in the pertinent literature. The common ground of various edge sparsification techniques is primarily putting the edges in decreasing order of importance, then filtering them out accordingly. These techniques can vary in ways of calculating importance scores for each edge, for significance edges can differ upon the network property of interest. Global sparsification randomly, or by triangle counts is prone to discard local structures for instance by ranking the edges according to the number of triangles in the network they belong to. Satuluri *et al.* [35] address this issue by holding on to at least one edge incident to each node while assigning edge scores according to similarities of their end nodes. Jaccard similarity measure

is utilized for evaluating neighborhood overlap between two nodes, correspondingly the edges leading to preserving local structures are kept in the sparsified network. The authors target a better community detection performance refer to their sparsification approach as Local Similarity. Furthermore, Simmelian Backbones method is proposed by Nick *et al.* [36] especially for making a distinction between edges belonging to dense subgraphs and edges situated in between such hubs. The term backbone of a network stands for the remainder network after sparsification as defined by Lindner *et al.* [34]. Again, based on triangle count, Nick *et al.* [36] produce ordered neighborhoods of all nodes to be filtered later using Jaccard measure similarly to Local Similarity sparsification.

Lindner *et al.* [34] adapt the logic behind the Forest Fire node sampling algorithm to edge sampling. Starting from a similar reasoning to random walk, the way that a fire originating from a random tree in a forest spreads is the underlying hypothesis for calculating edge scores of importance, where the trees are represented by the nodes whereas the network stands for the forest. In this version of the algorithm, it is assumed that the trees that have burnt down are not able to catch fire again, and the instant diffusion to multiple trees is viable, in contrast with random walks [37]. The edges that are visited the most tend to climb up in terms of their importance in the network, therefore, to survive filtering operations. Another suggestion by Lindner *et al.* [34] to the problem of edge sparsification is the Local Degree method. Local Degree sparsification aims to keep edges that are linked to high degree nodes in the resulted network; in a way, it favors the edges connected to hubs of the network. The neighborhoods are ranked according to node degrees, i.e. the number adjacent edges to a node, this once. The suggested method is claimed to well preserve network characteristics such as diameter and connected components, since it tends to hold on to edges between hubs of the network [34]. Intuitively, the expectations that structural properties would be conserved is plausible. However, before implementing sparsification in a real network, it is crucial to make sure that the preferred method aligns with the purpose of the study. For this study, local structures are worthy of attention, as well as the interior edges, representing the transactions that took place; so that insider edges might be

just as significative as the edges residing among dense groups. After all, anti-money laundering is a domain where loss of meaningful information is rather undesirable.

2.5. Extraction of Substructures

Primarily, it is essential to reestablish the idea that money launderers tend to be each other's neighbors in transactional networks; namely, suspiciousness is associated with connectedness. Hypothetically, they are connected by links that represent the transfer of the proceeds of crime between them; which reinforces the expectation that suspicious nodes may reside in the same communities in a transactional network. Since in AML, both edge and node reduction are necessary, a hybrid system utilizing a node and an edge approach is utilized. Following the initial network reduction, identifying suspicious subgraphs within the reduced network still remains as a critical part of the solution methodology.

2.5.1. Top-to-Bottom Community Discovery

In real networks such as the transactional networks that this research deals with, the distribution of links is both globally and locally inhomogeneous, with high concentrations of links within groups of nodes, and low concentrations between these groups. This structural property of real networks indicating a high degree of organization is referred as community structure by Fortunato [38]. Revealing the community structure of the transactional networks based on high average degree partitions can be the departure point for both network reduction and then suspicious subgraph detection. In terms of network reduction, partitioning the network into its densely connected components can facilitate coping with the massive network by dealing with each partition separately. For the latter, community detection can also reveal the hierarchical organization within the partitions, in the sense that these will include the suspicious nodes along with the nodes strongly connected to them which are not necessarily money launderers. This particular analysis is promising for the purpose of reducing false positives.

Newman [39] mentions that previously studied algorithms from graph theory literature proposed as a solution to graph partitioning problems such as the well-known minimum-cut problem, can be employed for community detection purposes. A cut-based graph partitioning method pointed by Rosvall *et al.* [40] provides a good separation of balanced groups in line with the purpose of network reduction. From cut-based perspective, Bansal *et al.* [41] claim that it is also possible to handle the issue of graph partitioning with minimal cost multicut problem as known as correlation clustering. As keynoted by Kappes *et al.* [42], the core of the correlation clustering problem, that is actually a node labelling problem, is to find a partition of the graph while minimizing the cost of intra cluster edges. This method eliminates a limitation of traditional graph partitioning methods, which is the necessity of specifying the number of communities ahead in an unknown network structure. Even so, utilization of traditional graph partitioning algorithms for community detection bring along the assumption that each node belongs to a single community. Fortunato [38] remarks that neglecting the possibility of overlapping communities in real transactional networks can result in undesirable loss of relevant information. That is to say, nodes belonging to multiple communities may be playing an intermediary role between subgraphs. Especially in the domain of AML where a major part of the network remains undiscovered, deriving such an information from network structure can be crucial in bringing new suspicious networks to light.

Considering both the structural and semantic characteristics of real networks along with the concern of computational efficiency, Newman [43] proposes utilization of more recent methods including the algorithm of Newman and modularity maximization. Suggested methodology is based on a divisive approach consisting of iterative removal of the links between nodes. Removal of the links start from the link with the highest betweenness value, which stands for the number of shortest paths between node pairs that runs along the specific link. The link with the highest traffic flow tends to be located between different communities the most [43]. On the other hand, maximization of the quality metric called modularity from an agglomerative perspective is proposed again by Newman [44]. The suggested methodology consists of joining the links between communities such that the modularity of the network is maximized.

For this study, conventional community detection algorithms can be troubling because of their computational cost and limited representative capacity of actual state of affairs. Aforementioned methods and their counterparts partition the network into disjoint communities, which implies the unrealistic prerequisite of belonging exclusively to one community for each node. Depending on the context, application of the overlapping detection algorithms captures any valuable information that a disjoint algorithm would leave out. Ideally, research in anti-money laundering requires working with algorithms suitable for directed networks that allow overlapping. Yang and Leskovec [45] introduce a community detection method, BigClam, applicable to large networks for determining node-cluster affiliations from a non-negative matrix factorization perspective. The authors object to well accepted conjecture that the overlap regions situated between communities are usually sparsely connected. On the contrary, they suggest that nodes in the overlapping regions have a higher probability of being linked because of their affiliation to more than one group [45]. BigClam formulates the problem of overlapping node-cluster affiliations as an optimization problem; where the objective is to find most probable affiliation matrix while maximizing the likelihood function of the underlying large network; and follows a gradient ascent algorithm to solve it. Despite its applicability to large networks, BigClam seems inadequate for the purpose of this research for it is designed for undirected, even unlabeled networks.

For finding overlapping communities in social communication networks, Baumes *et al.* [46] present an algorithm in two phases: At first, initial clusters are determined with List Aggregate (LA) algorithm, then they are iteratively improved using IS2 procedure with respect to cluster density function as the quality metric. For this approach, the density function plays an important role in both defining a community and improving its quality. The authors seek for local optimum of density function while expressing that what makes an ensemble of nodes a community is maximization of density among highly similar ensembles, differing only in terms of one single node [46]. LA algorithm ranks the nodes according to a criterion like PageRank, introduced by Page *et al.* [47], their score before producing initial clusters, improving the overall efficiency. Then, like modularity maximization methods, nodes are added to or removed

from clusters if doing so betters cluster density.

Li *et al.* [48] propose a method under the name LEMON having the capacity to extract overlapping small communities from large networks when a couple of known members of ground truth communities are fed into the algorithm. Among the drawbacks of structure-based community detection algorithms, excessive computational cost proportional to size of the entire network is presumably the most prominent one. For the very reason, LEMON comes out as particularly advantageous and eligible for large networks since its running time scales to size of the target community, instead of the entire network. Furthermore, LEMON wipes out the necessity of methods spectral clustering to compute as many singular vectors as the number of communities by deducing those via random walks around the seed nodes [48]. The authors lean their algorithm on intuition that complementary nodes are located most likely around given seeds. The objective regarding clustering quality in this work is minimizing conductance; namely, the better a graph is connected, the higher its conductance is. Here, selection of these seeds is critical for which the authors suggest various approaches such as random, high degree or low degree seeding method [48]. Inclusion of actually unrelated nodes in the seed set of a target community would inevitably lead to increased conductance.

The work of Leskovec *et al.* [49] exhibits common statistical characteristics of communities found in large social and information networks by empirical analysis. The findings of the work regarding the relationship between community sizes and qualities indicate that as the scale of communities increases, it is highly likely that communities mingle in the rest of the network and become less separated than other communities. Also, the authors explore the geometric settlement patterns of communities in social networks where majority of the nodes belong to large core of the network, that is faintly connected to smaller whiskers. Upon these observations, the classical definition of the term communities, having sparse connections with outside, entail an optimum range for meaningful community sizes, which is empirically found around 100 by Leskovec *et al.* [49].

2.5.2. Bottom-to-Top Community Discovery

In this study, we will also be working on a bottom to top approach for forming subgraphs. Both top to bottom and bottom to top approaches used for forming subgraphs are part of the search efforts within the massive financial network. These subgraphs built or extracted by partition represent the possible clusters where money laundering activities may have taken place. On top of this step, the subgraphs need to be scored in terms of their suspiciousness using machine learning techniques, which will be explained in the upcoming sections. In light of the research in Social Networks, ego-centric approaches for uncovering social circles, in other words, densely connected communities have captured attention. Especially in massive social networks, where the information held by both users and their connections is growing day by day, the analysis of the entire network becomes extremely challenging. Researchers have been focusing on methods based on constructing ego networks for extracting knowledge from massive relational data [50].

Building subgraphs from bottom to top using an ego-centric approach means that each subgraph formed consists of the n-step-neighborhood along with the connections between the neighbors of the central node, which is called the ego network of the central node. Briefly, the ego network of the i^{th} node is defined as the subgraph spanned by all neighbors of the i^{th} node [51]. In contrast with conventional community detection methods where deduced communities are disjoint, extracting communities based on ego networks is favored in some relevant works. Xie *et al.* [52] show that utilization of ego networks for community extraction allows overlapping communities while overweighing the sensitivity of those to the sparsity of real-life networks. Another work which exploits the benefits of ego-centric approach in community detection tasks is suggested by Rees and Gallagher [53]. From a similar Social Networks standpoint, the authors state that each node in the network owns information about its "friends-groups" by means of its ego network; thus, aggregating individual friends-groups lead to detection of overlapping communities, through a process called EgoClustering. Departing from construction of ego networks of each node, the algorithm developed by Rees and

Gallagher [53] advances iteratively by merging these ego networks into communities.

Another node-centric approach, ANGEL, allowing for detection of overlapping communities based on ego network structures is suggested by Rossetti [54]. The algorithm is assertive in terms of reducing computational complexity while tailoring the building principle of ego networks to yield fair-sized communities. Rossetti [54] opts for removal of the centroid from ego networks in accordance with the idea that centroids create noise since they are connected to everyone in the subgroup. The author calls resulting local groups ego-minus-ego, that constitute a basis for the algorithm. ANGEL agglomerates those with respect to a merging threshold parameter to form mesoscale communities. The reasoning agrees with the indispensable switch in scanning practice from macroscopic to microscopic scale in large networks as evoked by Li *et al.* [48]. Nevertheless, ANGEL and anti-money laundering might be at odds about inclusion of ego nodes in communities, because ignoring the connections of a supposedly suspicious center contradicts with the aim of discovering hidden money laundering patterns.

From an Anti-Money Laundering perspective, ego networks would be beneficial by while exposing the trained machine learning algorithm to various different aspects of a suspicious account or account holder in different ego networks in which it resides. Furthermore, Sengupta [55] has shown the usefulness of ego networks in the context of anomaly detection in networks. Here, the author proposes a computationally efficient egonet heuristic to the so-called planted clique problem, which focuses on revealing anomalous cliques, in other words fully connected subgraphs, within a large network. The difficulty of differentiating between non-anomalous networks and anomalous networks using network level metrics is overcome by detecting such anomalies by looking at subgraph level metrics instead, while these subgraphs are selected as ego networks. Yet another utilization of ego networks in anomaly detection tasks in weighted networks is proposed by Akoglu *et al.* [56]. An "Oddball" algorithm, which takes its name from the sphere-like structure resulting from induced subgraph of a node and its neighboring nodes, is proposed for extracting ego network of each node. Then, the features of these ego networks are calculated in order to be used in the unsupervised OddBall

algorithm to reveal outliers, namely anomalies.

In addition, it is acknowledged that ego-centric approaches might be even more efficient in finding groups that are appropriate in size for money laundering networks [15]. It should be kept in mind that the financial relationship network is a massive collection of accounts or account holders and their numerous transactions. The majority of financial interactions take place in the core of this massive network, whereas they are observed to be sparser in the extremities. Accordingly, community detection methods function properly when it comes to filtering the communities at the extremes of the network; however, they are not as satisfying in dividing denser parts in the core of the network. At this point, the author underlines the advantages of ego-centric approach as opposed to the limitations of community detection methods for extraction of such subgraphs to be scored later. Here, selection of the steps for including neighborhoods depends on the scope of the Money Laundering problem along with the content of initial financial network. Namely, if Money Launderers are investigated in a financial relationship network formed by the data provided by a higher authority, it means that the network holds information about international transfers or transfers between different banks. In such cases, further neighborhood can also be meaningful especially for suspicious central nodes and they should be considered as part of the subgraph. On the other hand, if the financial relationships are derived from accounts belonging to the same bank, closer neighborhoods carry most of the information about suspicious activities. For this reason, fine tuning of the parameters is necessary and will be done according to the context and content of the data.

2.5.3. Removal of Overlapping Subgraphs

Both bottom-to-top and top-to-bottom approaches used for forming subgraphs can result in a significant degree of overlap between those subgraphs. Considering the ultimate motivations of this research in terms of decreasing the manual efforts put in investigation of money launderers, thus reducing false positive alerts; presence of excessive overlap between communities should be avoided or eliminated when possible.

Specification of multiple highly similar subgraphs as suspicious separately will increase the necessary labor contrary to expectations. Removing overlapping subgraphs is by nature a challenging step. Before transferring the subgraphs to the analysts, a post-processing step should be carried out in order to decrease undesirable amount of work.

Most of the relevant work in the area focus on unionizing the resulted subgraphs if they are sufficiently similar under the condition that if the degree of overlap between communities is not worthy of note, they can be scored distinctly in the next steps. Nonetheless, if the overlap happens to be substantial, these similar subgraphs can be combined. Passing the union of overlapping subgraphs to the next steps is effective in decreasing redundant manual screening for the analysts. A similar method for reduction of overlapping subgraphs is proposed in the work of Rees and Gallagher [53] as part of their EgoClustering algorithm. A high degree of overlap between extracted subgraphs is naturally inevitable in case of ego networks; therefore, remitting those overlaps by merging all exact matches among those subgraphs is straightforward in the first place. The subgraphs which are exact copies of each other or one being a proper subset of another are merged without condition. On top of that, the method handles the subgraphs that are quite similar but not exact matches based on a rule that gauges the size of the intersection in proportion to the size of the smaller subgraph. Merging of subgraphs based on their calculated differences contributes in terms of constructing more meaningful communities departing from ego networks; thus, needs to be combined with aforementioned methods.

Some optimization-based community detection techniques, such as modularity optimization, can be inspirational in merging of overlapping subgraphs. Yan and Gregory [57] carry out reduction of the number of communities can be carried out by repeatedly merging pairs of communities, where increase in modularity is maximized at each union realized.

3. METHODOLOGY

Before diving into the suggested system step by step, it can be useful to look back on the ultimate goal of this study: It aims to achieve a significantly improved performance of suspicious activity detection in terms of both true and false positives by making use of network-based analysis this once in comparison with existing individual analysis approach. Recalling the challenges proposed by the very large network and scarce presence of fraudulent activity; research on how to reduce it down to a manageable scale and later how to filter meaningful subgraphs from that needs has been conducted as it is thoroughly mentioned in Section 2.4. Besides, benefits of artificial intelligence in reducing human effort are undeniable and the global trend in combining artificial intelligence with anti-money laundering should not be neglected. Therefore, the task of suspicious activity detection is formulated as a binary classification problem that is carried out via supervised learning methodology. An overview of our methodology is given in Figure 3.1.

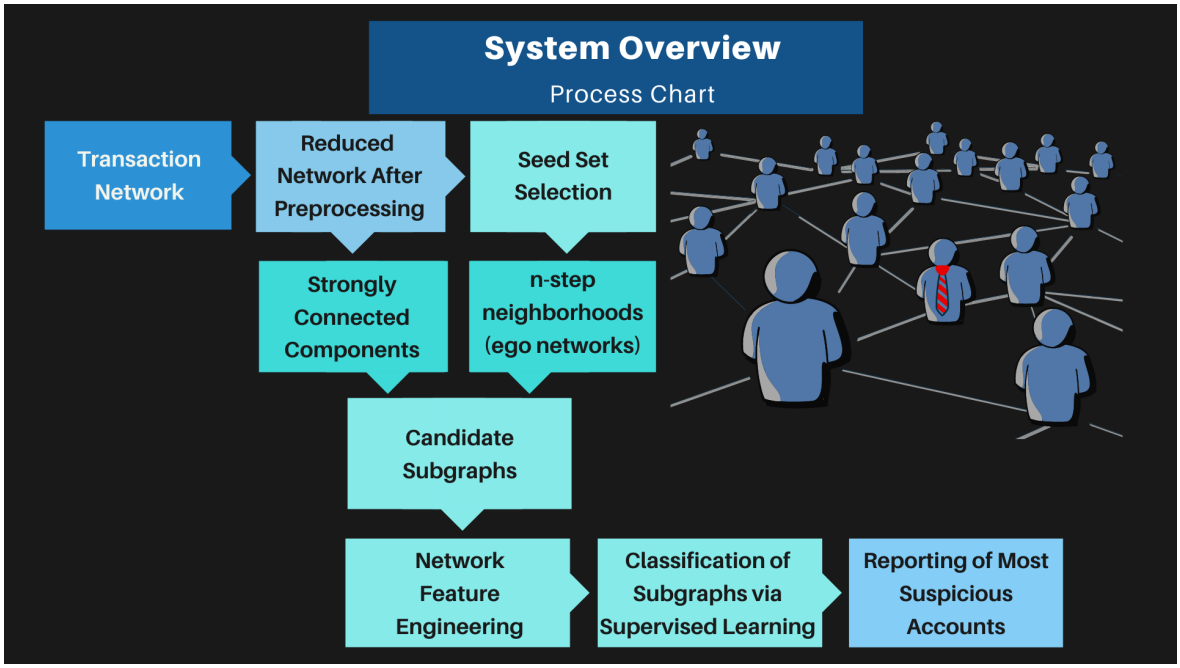


Figure 3.1. Overview of Our Methodology.

3.1. Network Preprocessing

At first, we aim to reduce complexity of large transaction networks that we are dealing with in order to simplify them down to a manageable size. For the sake of suspicious activity detection, we strive for conserving valuable components in the network.

3.1.1. Transaction Network

The system is principally designed for reducing the extensive load of manual analysis of labor force; in other words, improved efficiency. Ideally, the system feeds on historical transaction information provided by the banks along with previously tagged SAR accounts; and yields groups of accounts classified into two classes: Suspicious or innocent. The data provided by banks shall include transaction amounts with their source and target account identities, sometimes anonymously, categorical client information such as profession, gender and numerical ones as well such as account age. The availability and dimensionality of data merely depend on the banks' preferences regarding sharing their data under the legal restrictions and their code of conduct about protecting their clients' privacy. For the purposes of this research, anonymity of client or account identities does not count as a downside as long as their anonymization is done bijectively. All aside, this study expects the given data to be authentically labeled to be able to obtain interpretable results. A transaction network, preferably weighted and directed, is constructed based on the extent of given data. A symbolic representation of relationships in a bank transaction network is given in Figure 3.1.

Unfortunately, throughout this research access to real bank data could not be sustained. However, the expectations are still on the table since this study paves the way for further projects awaiting to be funded. When we evaluated alternative data sourcing mechanisms, such as synthetically generated financial networks or real-world social networks as mentioned in Section 2.3, with their pluses and minuses; we decided to prioritize originally labeled data sets over unlabeled financial ones within reach.

In a sense, we placed a premium on factuality of class difference over characteristic representation of financial networks.

Briefly, for the purposes of this study, a figurative network is created based on directed relationships derived from a public social network data set, collected by Fakhraei *et al.* [32]. The advantage of this data set from Tagged.com among its counterparts is presence of spam and normal account knowledge as binary labels; after all, it has been collected to be used in a spam detection research by Fakhraei *et al.* [32]. In the data, spammers are labeled as positive observations which is left intact for the sake of training supervised classifiers in subsequent phases. These positive instances typify SAR parties as if they were bank accounts. Naturally, adaptation of this social network as an exemplary raw data for anti-money laundering purposes is not limited to semantically matching. As regards to financial analysis, a critical shortfall property of social networks is lack of edge weights, namely transaction amounts. Social networks usually include relationships as present or not, based on their occurrences. However, even primitive rule-based detection systems, explained in Section 2.1.1, attribute great importance to money amounts in terms of pointing suspicious activity. This deficiency is, so to say, patched using estimated parameters from a synthetic financial data set generated via PaySim [30]. Further details regarding data preparation, network construction and its relevant properties can be found in Section 4.1.

3.1.2. Network Reduction

Considering the particularly challenging aspects of very large financial networks and scarcity of known suspicious activities inside, jumping directly to the utilization of subgraph extraction and supervised learning is not viable in anti-money laundering research. Preprocessing of the large network inevitably lies ahead of scanning the entire transaction network. There are some nodes that have connections with most of the nodes, whose existence shadow an important extent of the information, their removal is required as a part of preprocessing. As we mentioned in Section 2.4, previous works devoted to network reduction revolve around the notion of sparsification.

Although sparsification methods seem promising for their efficient implementations in large networks, they tend to disregard almost 80% of the relations that constitute the network. For the purposes of this study, the information carried by direct relations between accounts is irreplaceable; hence, we decided to place sparsification methods aside and carry on with thresholding assumptions supported by domain expertise.

In the first place, our system performs edge filtering with respect to transaction amounts. Such a removal procedure is supported by the fact that in real life, transactions under a certain lower bound hardly imply suspicious financial activity. This claim is supported by expert opinion acquired from H3M [58]. The lower bound for edge filtering is estimated using the same sample from the transaction data synthetically generated via PaySim, as the one used for producing artificial edge weights [30]. For the relevant sample, specified lower bound is 10 in whichever local currency is used in PaySim's original data. As far as it is known, the actual mobile money transaction sample is acquired from an African country to constitute the base for the simulator. The authors choose to cover further information for plausible privacy related reasons. Setting the lower bound of transaction amount to 10 leads to elimination of approximately 35% of the edges while preserving all the nodes in the network. The ratio seems sensible for the sake of reducing complexity and guarding significant transactions in the meantime. Additionally, an analogous procedure can be implemented for recurrence of transactions guided by the claim that transactions that have taken place only once are relatively insignificant. In the circumstances, we preferred to bypass our latter suggestion for edge filtering since even the synthetically generated sample via PaySim does not include any information about number of transaction occurrences but rather the total amount of transactions that may have taken place several times during a period of time. The only feasible and rather sensible way to involve the procedure on table would be assigning completely random repetition attributes to edges. We inferred that random assignment would not contribute considerably in suchlike applications to adapted data. In case a real transaction network becomes accessible, execution of the procedure in question would be of help to rule out the crowd in the network.

When it comes to node filtering, our system concentrates on instances located at the extremes of node degree distribution. At first, Freeman [59] defines the degree of a node as the number of neighbor nodes that it is adjacent to, in other words the number of nodes that it is linked with. In a transaction graph, some nodes attract notice because of their ultrahigh degrees, causing distortion of the algorithms seeking suspicious activity in the graph. In general, these nodes represent administrative or official accounts such as tax authorities, electrical distribution companies or water administration; therefore, their edges stand for billions of transactions of bill or tax payments made by millions of semantically irrelevant accounts. Naturally, such periodic transactions linked to nearly each node in the graph increase degrees of official accounts dramatically while bringing them to the top in terms of connectedness. The task of eliminating these knowingly clear nodes is done by thresholding like edge filtering steps. Here, the value of degree upper bound is identified as the 99.9th percentile in the node degree distribution. Surely percentile cut can be positioned according to sample distributions or adjusted by expert opinion.

In the end, network reduction is finalized by ruling out isolated nodes, in other words nodes having no connection with outside. Before preprocessing, the raw network had been built based on an edge list, including only the list of directed relationships with source and target parties. Although initially the raw network does not contain any isolated nodes, their appearance is eventually inescapable because of removal of a portion of edges from the network through reduction phases up to this point. Emergence of isolated nodes show that in fact, they were only connected to the rest of the network through insignificant relations; so that, their loss would not deprive us of essential knowledge.

3.2. Extraction of Candidate Subgraphs

This study aims to utilize subgraph extraction methods that can provide the capability to scan the entire transaction network in order to detect the well-hidden money launderers. Scarce presence of money launderers inside very large financial net-

works implies that presence of criminals who are not connected to any known suspects is highly likely. Here, the spatial reflection of money launderer behaviors gains importance. If an accurately filtered subgraph, including or sometimes centering money launderers, can be extracted, they can be useful for training a supervised classifier to identify suspiciousness of newcomer subgraphs embedded even in the undiscovered parts of the network.

3.2.1. Search for Typological Structures

In the first place, our system conducts a search based on structural hypothesis about money laundering typologies. Among those typologies, circulation of funds to disguise the ownership of dirty money draws our attention. A group of individuals exhibiting circling behavior; namely crowding the subgraph with too many unnecessary transactions without a net change of value at total, raises doubts. Therefore, the search for candidate subgraphs within the reduced network initiates by uncovering circling groups, shown in Figure 3.2. Here, we choose the adjective candidate as a hint for relative possibility of the extracted subgraphs to embody suspicious financial activity. In other words, candidacy means that the subgraph is worth exploring and has the potential to contribute in training of the supervised classifier in the following phases. For the sake of computational efficiency, we reformulated the problem as discovery of strongly connected components that contain more than one node. What makes a directed network strongly connected is existence of a directed path between distinct pair of nodes. Strong connectedness of a network implies its overall connectedness, thus reciprocal reachability of each node from every other node [21]. Therefore, the term strongly connected component stands for a maximal subset of nodes in a directed network that are strongly connected to one another. Lastly, strongly connected component subgraph is a graph created using all members of the strongly connected component along with all the relations between them. It is not necessary for each member of a strongly connected components to be each other's neighbors; however, existence of a directed path between each pair of nodes is essential [60].

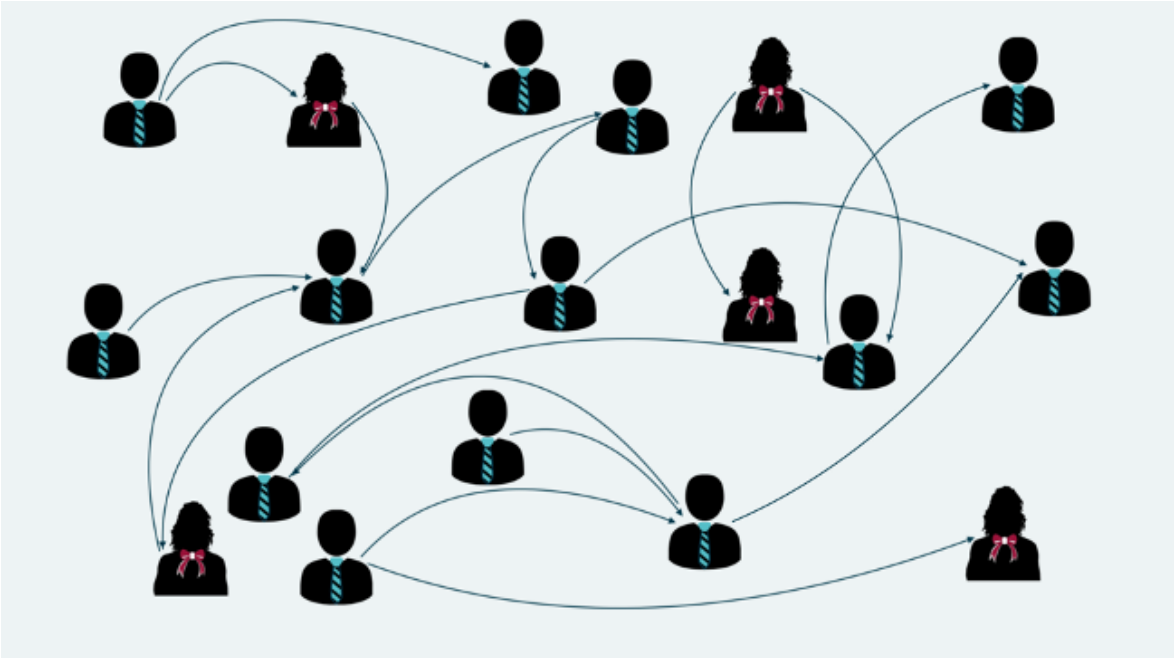


Figure 3.2. Representation of Circulating Groups in Financial Networks.

We can infer that if two member nodes u and v of a directed graph G reside in the same strongly connected component C extracted from G, u should be reachable from v and there should exist a directed path from v to u as well. This means that a directed cycle passing from u and v can be found in any pair of members in C. This deduction can be translated to discovery of circulating behavior in a subgroup under structural considerations related to money laundering typologies. We prefer not to overlook such structures clearly matching money laundering typologies and record them as subgraphs to be processed in further steps.

Search for strongly connected components is carried out using a fundamental graph traversal algorithm: Depth First Search (DFS) [60]. DFS starts from one node, goes on by choosing one of its neighbors that have not been visited yet and continues until the set of unvisited nodes is exhausted unless it encounters one of the previously visited nodes through the journey. Then the algorithm terminates and declares presence of a cycle in the graph. In our approach, we discover all the strongly connected components including more than one member and construct implied subgraphs from

those. Meaning that we copy the labeled members along with their adjacent edges with one end belonging to one of other members of the component.

Components having a single member are not meaningful for the purposes of this study since they lack valuable information brought along by connections between pairs of nodes. The resulting subgraph, much smaller compared to the original network, is enlisted as one of the candidate subgraphs. Then again, we think it is remarkable that extracted subgraphs are not always dense subsets of the large network; they may even be sparsely connected compared to the rest. In a way, we relax the requirements regarding high density coming from the traditional community extraction approaches, mentioned in Section 2.5.1, for a better catching capacity. Money laundering behavior is not necessarily embedded in high number of connections, instead focusing on the structure of those connections seems more sensible. The second method we propose for extraction of candidate subgraphs, explained in Section 3.2.2 below, bears the trace of this argument as well.

3.2.2. Subgraph Construction from Ego Networks

From an Anti-Money Laundering perspective, bottom-to-top construction of substructures would be beneficial by while exposing the trained machine learning algorithm to various different aspects of a suspicious account or account holder in different subgraphs in which it resides. Also, the drawbacks of existing top-to-bottom community detection methods support the tendency towards bottom-to-top community extraction. First of all, the computational cost of top-to-bottom community detection is heavily disadvantageous for this study since the screening out an entire transaction network takes a very long time because of its massive size. Then again, restriction of overlapping communities with the assumption that each node can have a membership of only one community is not realistic for transaction networks. One account may send or receive money under many different circumstances; namely, money launderers may also be paying their bills from those accounts which makes them a member of both normal and suspicious activities. We want our system to bear witness to many

aspects of such accounts to develop a better discriminative capability between what is normal and what is suspicious. For this reason, our system suggests building ego networks as candidate subgraphs. This means that each subgraph formed consists of the $n - step - neighborhood$ along with the connections between the neighbors of the central node, which is called the ego network of the central node. Basically, the ego

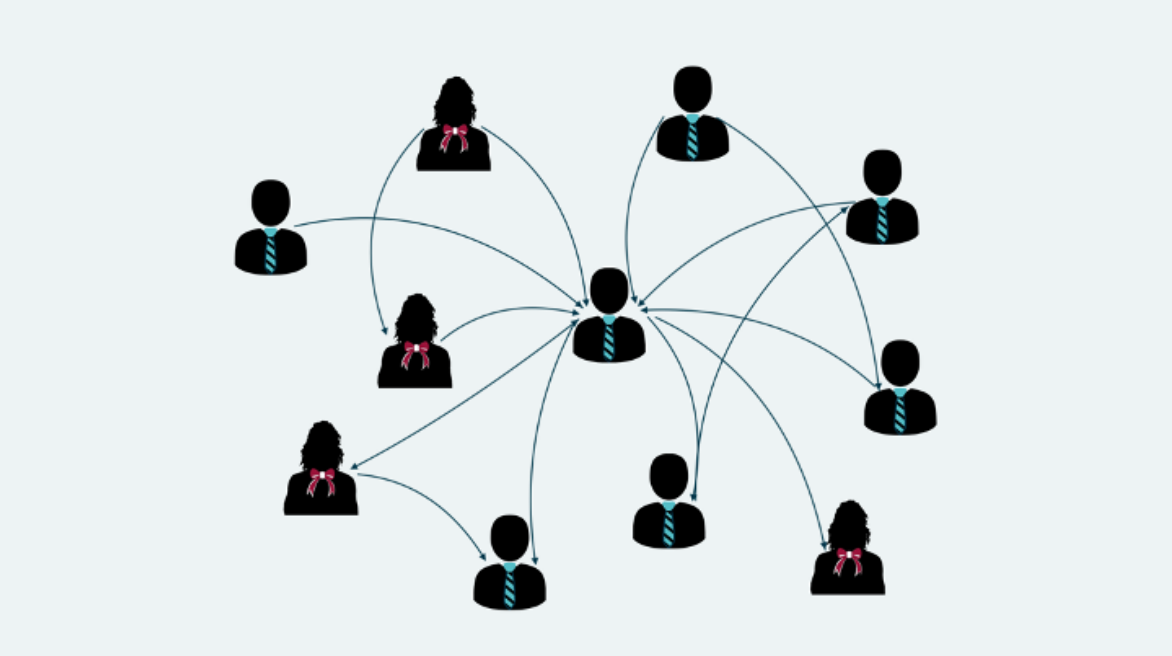


Figure 3.3. Representation of 1-step-neighborhood in a Financial Network.

network of the i^{th} node is defined as the subgraph spanned by all neighbors of the i^{th} node [51]. The set of $1 - step - neighbors$ of a node i consists of all nodes j such that there is a directed edge from i to j or there is a directed edge from j to i. Departing from this definition, the smaller ego network that can be built is the subgraph of $1 - step - neighborhood$ of the central node, itself included. When it comes to $2 - step - neighborhood$, the radius of ego network is incremented by one such that the neighbors of $1 - stepneighbors$, that were not in direct relationship with the central node, are included in the node set. In our approach, we parameterize the number of neighborhood layers for constructing ego networks, which translates to the radius of the subgraph. It is critical to customize ego network construction algorithm according to the substance of the problem. For instance, when illegal gambling is aimed

to detect, it makes more sense to consider $2-step-neighborhood$ of the suspects since in $1-step-neighborhood$, the gamblers shall reside, being connected to other suspect via their bets. On top of that, the structural properties of the sample data

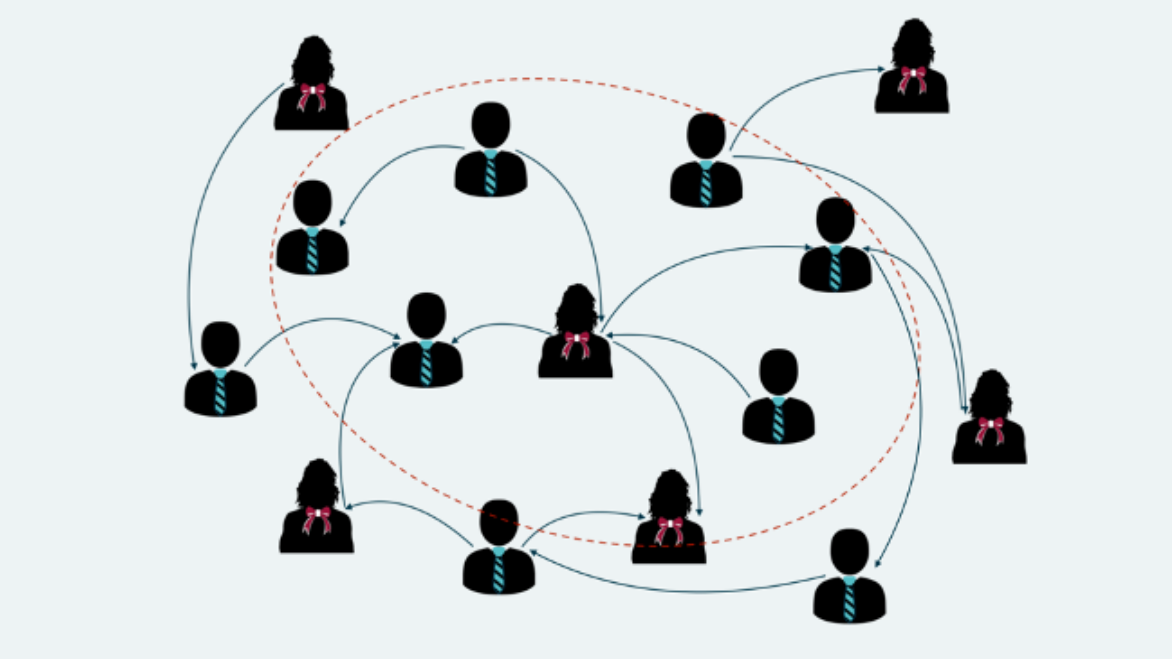


Figure 3.4. Representation of 2-step-neighborhood in a Financial Network.

matter a lot while setting up the algorithm. In some cases, increasing the radius of ego networks would lead to a very quick growth of the subgraph, which is unfavorable since meaningful communities in real-life networks are expected to contain around 100 members [49]. The more the coverage of the extraction method expands, the less indicative the resulted subgraph becomes. In this study, we extracted ego networks within $2-step-neighborhoods$ of the selected central nodes, as known as centroids.

Here, another interrogation point arises about the seed selection method. Clearly, the ideal setting for construction of ego networks would be to cover the entire network, leaving no potential insight ignored. However, understandably, such a plan carries with it an extraordinary need for computational capacity. Therefore, we proposed a high degree seed selection method inspired from the work of Li *et al.* [48]. The selection procedure requires ordering of all the nodes in the network according to their

degrees, i.e. number of connections, and samples a portion from the top as centroids of prospective ego networks. Naturally, the size of the sample also depends on the available computational capacity. High degree seed selection method is promising for carrying more information along with the connections it brings into the sample as opposed to other seed selection methods such as low degree or random seed selection. However, we do not settle for one approach and we also make use of random seed selection for the sake of preserving the original class distribution. Experimental results obtained from both seed selection methods are given in detail in Section 4.3.

Finally, before diving into the calculation of network features, the candidate subgraphs need to be labeled as suspicious or clear, under several scenarios regarding risk aversion behaviors of the banks. In the first place, we created a tolerance upper bound parameter reflecting their tendency to take risk and carried out the experiments according to three different levels of tolerance upper bound: 0%, 25% and finally 50%. Tolerance upper bound stands for the maximum percentage of suspicious accounts in a subgraph that banks would tolerate and bypass. Accordingly, low level of tolerance upper bound indicates high risk aversion behavior, such that presence of even one suspicious account in a subgraph is necessary for doubting that particular group. Under high risk aversion scenario, ego networks are identified with respect to their centroids; namely, if the central node has a positive label, its ego network is also labeled suspicious and vice versa. Furthermore, moderate and low risk aversion scenarios where the tolerance upper bound parameter is set to 25% and 50% respectively, labeling of subgraphs is done according to the percentage of suspicious nodes in the network. This means that if a higher percentage of the nodes in a candidate subgraph than the tolerance upper bound are suspicious; then we claim there is enough evidence that the subgraph in question exhibits suspicious behavior. The tolerance upper bound of the scenario where risk is most tolerated is set to 50% since an upper bound higher than 50% already would not make sense to tolerate from an anti-money laundering perspective. Labeling of both strongly connected components and ego networks are performed with the same procedure.

3.3. Selection of Network Features

Completion of candidate subgraph extraction is the first step in preparation of a data frame in our system. Each candidate subgraph is a data instance, whose features need to be computed for training a supervised classifier in the upcoming phase. For the purposes of anti-money laundering, we favor transfer of the relational insights embedded in the network into classical features over relational learning methods mentioned in Section 2.2.1. The features considered in our system are selected according to their representative ability of money laundering behavior and their capacity to characterize a graph. In a sense, significant features should help the model to differentiate between normal and suspicious subgraphs both semantically and structurally. Proper feature engineering is critical for expecting a satisfactory performance from classifiers. Network features of choice are listed below:

- Density: The density measures connectedness of a graph by dividing the number of actual connections by the number of maximal connections, i.e. total possible number of edges [21]. Suppose V is the number of nodes and E the number of edges present in a graph, the density D of a directed graph can be found using the formula:

$$D = \frac{E}{V(V-1)}$$

  The density measuring the pairwise connections in a subgraph can be a meaningful metric for the purposes of anti-money laundering. The intensity of transactions can help discriminating between innocent and suspicious financial activities.
- Number of Nodes: The total number of member nodes in a subgraph is the simplest metric for its magnitude. It is clearly peculiar to each subgraph.
- Number of Edges: The total number of edges in a subgraph corresponds to the number of transactions that took place inside the subgraph in the context of this system. In brief, it is referred as size of a network. Just as number of nodes, it is clearly peculiar to each subgraph.

- Weighted Sum of Edges: In a financial network, this feature refers to the total transaction amount within a group of accounts. Even out of date rule-based detection systems would assign this feature as an indicator of suspicious activity. However, this once, we do not set a threshold by assuming a positive correlation between total transaction amount and suspiciousness, instead we attribute importance to this quantity and let the classifier decide on its impact in the responses.
- Number of Incoming Transactions: This feature is also related to search for money laundering schemes and it stands for the number of transactions that are flowing from outside to inside of the subgraph. It means that this metric counts the number of edges that have their source nodes outside the subgraph in question, but their target ends inside. It serves an indicator of blockhole-like behavior, namely a sink where money accumulates.
- Volume of Incoming Transactions: Alongside the number, total volume of incoming transactions to a subgraph is also a significant feature for discovering financial anomalies. This is an extension to the feature above and calculated by summing up the weights, i.e. transaction amounts of all the inflowing edges situated on the boundary.
- Number of Outgoing Transactions: The other way around, number of outgoing transactions is the count of edges that are, so to speak, born inside the subgraph and ends outside of it. From a broader perspective, this is an indicator of volcano-like behavior of a subgraph and has the potential to imply financial anomaly.
- Volume of Outgoing Transactions: The volume on outgoing edges situated on the boundary is at least as significant as their numbers. This feature is calculated by summing their weights, just like it is done for the incoming edges.
- Average Degree: Average degree of the network is another indicator of network connectivity. In a directed simple graph, in-degree of a node n is the number of in-neighbors it has, which is the same thing as the number of edges whose target is n. Similarly, out-degree of a node n stands for the number of edges whose source node is n. In a directed network, average in-degree and average out-degree have the same numerical value since for every tie, there is a source and a target [21].

Let n be a node in a network with node set V, k^{in} be the in-degree and k^{out} the out-degree of n; average in-degree and out-degree of the network is calculated using the formula:

$$\bar{k}^{in} = \bar{k}^{out} = \frac{1}{|V|}\sum_{n \in V} k_n^{in} = \frac{1}{|V|}\sum_{n \in V} k_n^{out}$$

- Average Clustering Coefficient: Clustering coefficient is a structural metric which measures the degree to which nodes in a network are prone to cluster together. In other words, it shows how close the neighbors of a node are to forming a complete graph with unit density [61]. Let $|e_n|$ be the number of edges with both endpoints in the $1 - step - neighborhood$ of node n, and k_n its degree; clustering coefficient CC_n of a node n is found by:

$$CC_n = \frac{2|e_n|}{k_n(k_n - 1)}$$

Accordingly, average clustering coefficient CC of a network, is found by averaging clustering coefficients of all nodes in the network where V is the set of nodes:

$$CC = \frac{1}{|V|}\sum_{n \in V} CC_n$$

Clustering coefficient measures how well the neighbors of the nodes in a network are connected. Therefore, a small value of clustering coefficient indicates that the number of triangular relationships in the network are rare [21].

- Average Weighted Clustering Coefficient: This feature is an extended version of average clustering coefficient by means of edge weights. Here, the only difference is taking the edge weights into account while calculating local clustering coefficients of each node, whereas averaging operation at the second step is done the same way.
- Average Shortest Path Length: The shortest path length between two nodes u and v is also called the geodesic distance, that is equal to the minimum distance value among all walks going from u to v [62]. Therefore, average shortest path

length is calculated by average number of steps along the shortest paths for all possible node pairs in each candidate subgraph.

- Average Weighted Shortest Path Length: In weighted networks, the meaning of shortest path transforms into least costly path, which can yield different paths than the accustomed computation [63]. Weights can have a significant impact on strongness or weakness of the connections between two nodes, thus their consideration would alter the value of average shortest path length. For this reason, we preferred to compute the weighted variation of average shortest path length of the candidate subgraphs as well.
- Diameter: Diameter is classified among the distance and path measures in networks [21]. It is also referred as the longest geodesic or the maximum eccentricity of a network where eccentricity of a node u is the largest distance from u to any other node v in the network such that $v \in V$. Then, diameter stands for the longest distance between two nodes when all possible pairs are considered. Let d_{uv} be the distance of a walk from u to v, and V the set of nodes of a network, the diameter is given by $\max_{u,v \in V} d_{uv}$.
- Radius: This feature reflects the minimum eccentricity value among all nodes of a network [21]. It translates to the minimum of all the longest pairwise paths present in the network.
- Number of Triangles: As its very name signifies, this feature is the number of triangles present in the subgraph where all three nodes and edges should be a member of the same subgraph. First, we calculate the number of triangles that they are a part of for each node, then we divide their sum with three since each triangle is counted three times.
- Transitivity: Finally, transitivity is equal to the ratio of all possible triangles in a network. This feature has a similar logic to density measure, we can interpret that it stands for the realization ratio of triangles. Basically, the number of triangles present in the candidate subgraphs is divided by the maximal number of triangles, i.e. all possible combinations including three nodes. True to its name, this feature shows transmittance capacity in a network. From an anti-money laundering perspective, transitivity might indicate financial anomalies such as

flow of money between three accounts, or a unique ownership for three separate accounts, and so forth.

3.4. Machine Learning

We formulated the problem of identifying suspicious financial activities in transaction networks as a supervised binary classification problem where the positive observations are the suspicious candidate subgraphs and negative instances are clear candidate subgraphs. Principally, classification models are designed to output probabilities for each observation regarding their ownership to each class, then assign those observations to discrete classes with respect to these probabilities compared to a certain threshold [64]. Binary classification problem is modeled under the assumption that each observation pertains to either one or the other classes, in our case either suspicious or clear. The relational information carried by network features are tackled as classical features to be fed into a supervised classifier. Recalling the objective of this study to be as efficient as possible, we propose utilization of three different types of novel methods for binary classification task: Random Forest, XGBoost and Light Gradient Boosting. All three methods are examples of ensemble learning, where multiple learners are trained to obtain a better predictive performance than decision trees with lower variance [64].

In the first place, we split our candidate subgraphs data into two sets for the sake of supervised learning: Train and test sets. Test set comprises 20% of the entire observations where the splitting operation is done randomly. We set the test set aside, perform all the training operations using the train set, then evaluate the optimized classifiers on the intact test set. At this juncture, it would appropriate to speak of a frequently observed data related issue in practical applications of such models: Imbalanced class distribution. The issue refers to the problem of skewed distribution of the observations in training set with respect to their class labels such that one of the classes steps forward as majority whereas the other one remains minority. In some real-world problems like fraud detection, a severe imbalance between classes, such that

there are hundreds or more examples belonging to the majority class against one in minority, happens to be a common challenge [65]. Class imbalance can cause insufficient learning of the minority class understandably; hence, evaluating the models built upon imbalanced class distributions the same way as the ones learning from balanced classes can be deceptive. Subsistence of class imbalance is entirely dependent on a particular data set or sampling procedure [66]. Bearing this in mind, the evaluation metrics are chosen properly considering class distributions of the experimental samples in this study (Section 4.3.1).

Random forest is a classification algorithm consisting of a large number of individual decision trees with branches based on features and function values in the leaf nodes [67]. Each individual tree in the random forest yields a class prediction and the class with the most votes serve as the model's final prediction. Random forest guarantees that the behaviors of individual decision trees do not correlate with each other by bagging. This means that each decision tree samples data instances randomly with replacement. The reason behind this is the aim to decorrelate the trees in the forest [64]. The important thing here is that the random forest classifier should be fed with discriminative power between two classes.

To contribute in the efforts of the model for obtaining low correlation between trees, we performed hyper parameter tuning to optimize random forest classifier over a parameter space using a grid search 5-fold cross validation procedure. The selected parameters to be tuned are number of estimators, corresponding to the number of trees in the forest, and max depth of the trees. Number of trees naturally increases the predictive capability of the model while also increasing the computational cost, whereas max depth should be fine-tuned in order to avoid overfitting with unnecessarily deep forests. Principally in 5-fold cross validation, the train set is divided into 5 folds, out of which 4 are used for training while the remaining fold is saved for evaluation. The classifier is trained and tested this way 5 times to finally yield an average of the results from each set. Grid search on the other hand, is a procedure introduced for hyper parameter tuning, meaning that for choosing the best model parameters that

optimizes the model's performance. While the process of selecting hyper parameters could be carried out manually, it would be a long and complex chain of trials. Grid Search facilitates these efforts by automatically comparing each model while cross-checking their performances and pointing out the best parameter combination for a given train set, obviously.

Next, we decided to implement a novel gradient boosting algorithm introduced by Chen and Guestrin [68], XGBoost, which can be implemented to large and scarce data. Just like bagging, boosting methods are also part of ensemble learning which combines multiple learners in order to achieve a higher fitting performance with lower variance [64]. As opposed to bagging which independently takes simple averages of responses, boosting functions sequentially. This means that new models are iteratively added to the ensemble while improving the poor fitting aspects of previous models. Gradient boosting is a prominent example of boosting algorithms which uses gradient descent for approximating the loss function [69]. Additionally, gradient boosting models seem beneficial owing to their intrinsic remediation for aforementioned imbalanced classification issue. While iteratively sampling training sets, these models aim to enhance the performance of the classifier in progress by giving more weight to selection probability of previously misclassified examples. XGBoost is a parallelized effective library created under the gradient boosting framework [68]. For the hyperparameter tuning of our XGBoost implementation, we chose to experiment on learning rate, number of estimators and max depth of the forest, using grid search with 5-fold cross validation, again. Learning rate can be understood as the step size of the gradient boosting algorithms while approximating the loss function.

Finally, we chose to apply a most recent algorithm under the gradient boosting framework, LightGBM, which is very promising with its excessive efficiency and ability to achieve accuracy comparable with its predecessors, i.e. XGBoost. The main change in perspective introduced by LightGBM is to prefer horizontal expansion in the forest rather than increasing the depth [70]. Accordingly, we added number of leaves this once to the parameter space for hyper parameter tuning.

4. EXPERIMENTAL STUDIES AND RESULTS

This section is dedicated to our experimental studies and discussing their results.

4.1. Data Preparation

The anonymized dataset was sampled from the dataset collected from Tagged.com social network website by Fakhraei *et al.* [32]. Original data provided by the authors include 7 different, again anonymized, types of relationships between users of the social network. We filtered these relationships to obtain homogeneity in the edges and sampled all the users that interacted through one specific relation type with each other. The sample includes 119230 users among which 41981 are known spammers and 106851 edges between them indication one specific type of action. Here, the positive instances constitute 35.2% of all the observations; however, such a fraud percentage is not realistic to expect from the banks in the domain of anti-money laundering since the contemporary methods are able to detect only around 0.2% of the suspicious financial activities [10]. We approach this relatively high ratio of positive observations in our authentically labeled sample data as an advantage for better training the classifiers with respect to both labels. A directed graph is created using these edges as the main object of analysis. The overview of the graph before preprocessing is given in Table 4.1.

Table 4.1. Overview of the Sample Graph before Preprocessing.

Type:	**DiGraph**
Number of nodes:	119230
Number of edges:	106851
Average in-degree:	0.8962
Average out-degree:	0.8962

Before reducing the network, weights are assigned to each edge to imitate a financial transaction between two parties. The gamma distribution of the financial transaction amounts is verified by means of the simulated transaction sample via PaySim and the shape and scale parameters of gamma distribution are estimated using the same sample [30]. The distribution of the synthetically generated transaction amounts using PaySim is given in Figure 4.1. Let μ be the sample mean of transaction amounts and σ^2 their variance, we estimated gamma distribution parameters shape α and scale β using the following formulas:

$$\alpha = \frac{\mu^2}{\sigma^2} \qquad \beta = \frac{\sigma^2}{\mu}$$

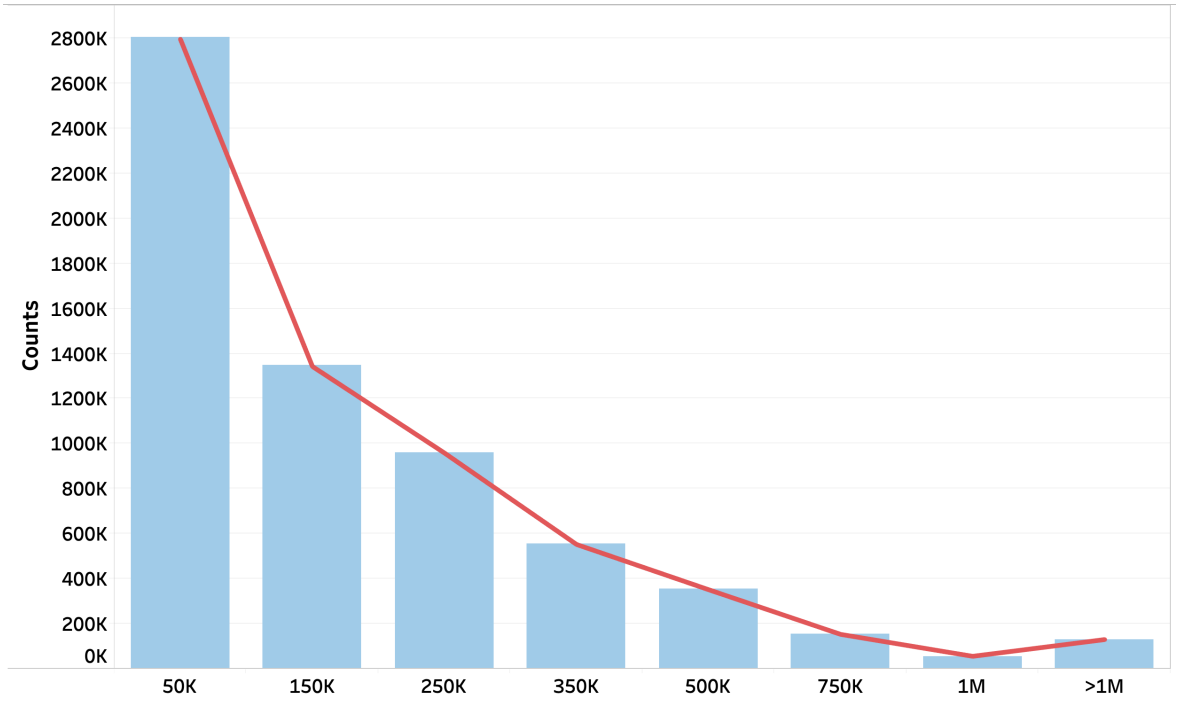


Figure 4.1. Distribution of Synthetically Generated Transaction Amounts via PaySim.

Using the estimated shape and scale parameters, we generated random gamma variates as transaction amounts and assigned them to each edge in the raw network, as

explained in Section. 3.1.2. Then, following the network reduction steps, we performed edge filtering and node filtering respectively to obtain the reduced network which is overviewed in Table 4.2. The length of the raw network, i.e. the number of nodes, has been reduced down to 66% of its original value whereas the size of the network corresponding to the number of edges are filtered down to 55.8% of the initial size.

Table 4.2. Overview of the Sample Graph after Preprocessing.

Type:	**DiGraph**
Number of nodes:	78674
Number of edges:	59618
Average in-degree:	0.7578
Average out-degree:	0.7578

Among the most important features according to information gain criterion, given in Table 4.14, the most prominent ones, except for volume of outgoing transactions and inside transaction volume, are plotted against the ratio of suspicious ones to all candidate subgraphs. The two features are excluded from this descriptive analysis since they are based on transaction amounts that are generated using estimated parameters and assigned randomly in the beginning. Therefore, we decided to analyze the discriminative potential of important features calculated based on the real network characteristics.

In Figure 4.2, suspicious subgraph ratio, which is the same thing as the number of positive observations divided by the number of all observations, is plotted versus the values of 6 different features for the random sampling case under high risk aversion scenario. The same analysis for the samples generated with random seed selection procedure under moderate risk aversion scenario is given in Appendix A and the one under low risk aversion scenario can be found in Appendix B.

It is clearly seen in Figure 4.2 that number of incoming transactions, average degree and diameter are the features that have a positive correlation with suspicious subgraph ratio. Meaning that the probability that a subgraph is suspicious gets higher for higher levels of each of these three features. The number of incoming transactions for a particular subgraph refers to the number of transactions of which the sender is outside the subgraph where the receiver resides inside of it. This implies that suspicious subgraphs tend to have a higher number of inflowing transactions into the group, which reminds of a blackhole-like structure.

When it comes to average degree, it stands for the average number of connections each member of the subgraph has. Observing a positive correlation between average degree and suspicious ratio is supported by the inference we made in Section 4.2 about the tendency of suspicious individuals towards having more connections. Here, we can interpret that suspicious subgraphs are more likely to include members who involve in more transactions.

Lastly, diameter changes in the same direction as suspicious subgraph ratio, which gives insight about structures of suspicious subgraphs. Among the subgraphs with diameter equaling to 1, almost 30% are suspicious whereas when diameter goes up to 3 or 4, 50% of them appear suspicious. This basically indicates that suspicious subgraphs are more likely to contain remote couples compared to clear ones. It should be noticed that diameter is not in fact informative about all paths between all node pairs in a subgraph; instead, it gives insight about the longest path between two most distant nodes.

As regards to remaining important features, number of outgoing transactions, average shortest path length and density are negatively correlated with suspicious subgraph ratio. Negative correlation with the number of outgoing transactions aligns with abovesaid resemblance to blackhole structure. Moreover, as the average of shortest path lengths between all pairs in a subgraph increases from 0.1 to 0.5, the subgraph is approximately 25% less likely to be suspicious. This means that in a suspicious

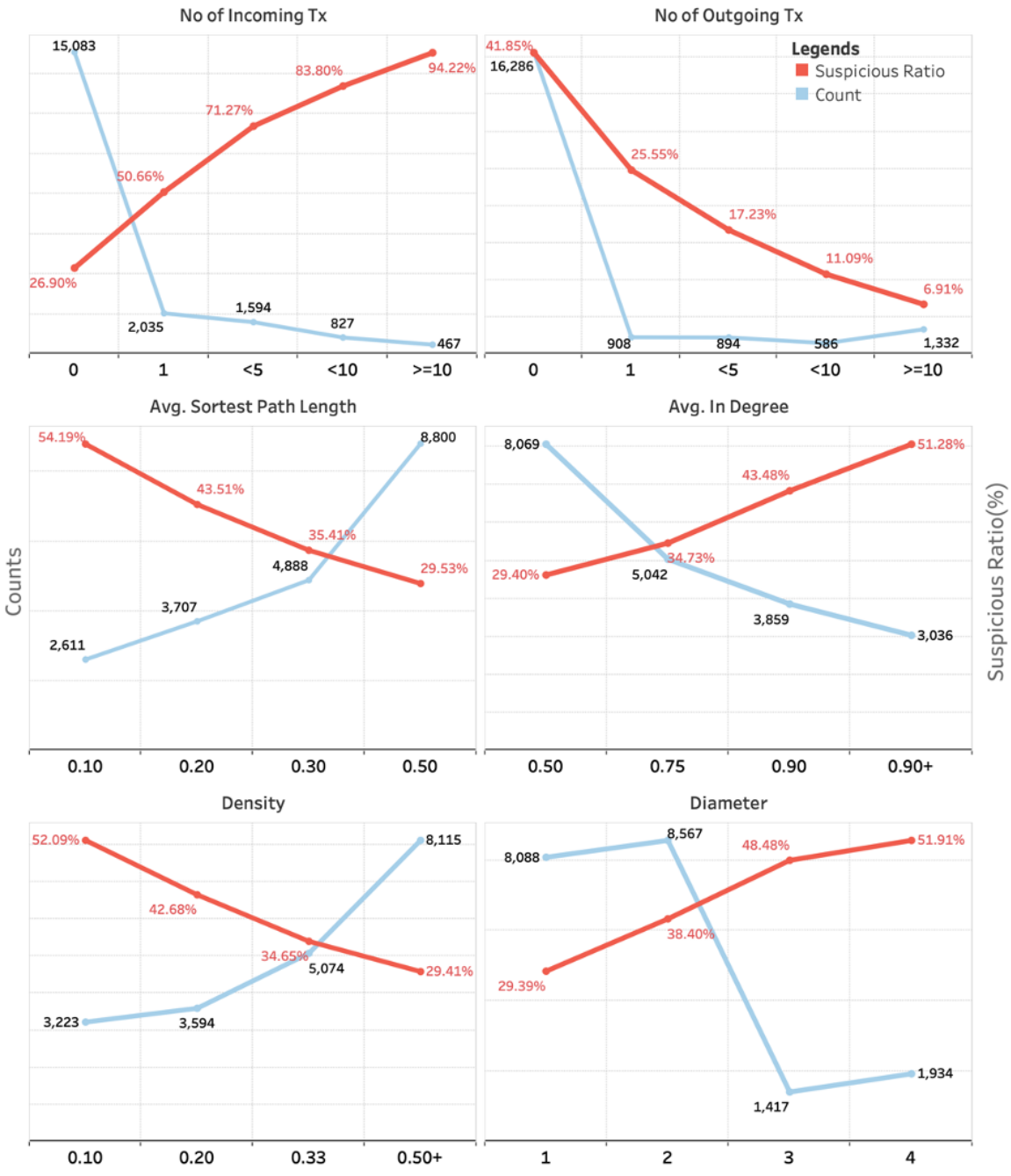


Figure 4.2. The Relationship between Suspicious Subgraph Ratio and Important Features for High Risk Aversion Scenario.

subgraph, average distances between node pairs tend to be shorter compared to a clear one. Combining this deduction with the positive correlation of average degree, it is understandable that member accounts of a suspicious subgraph are more inclined to make transactions with each other as well as outside, creating more directed paths in between thus shortening the distances.

Finally, density is negatively correlated with suspicious subgraph ratio, meaning that sparser subgraphs are prone to be suspicious. This is a counterintuitive observation and it can be explained when it is combined with suspicious ratio's trend against average degree, such that the transactions which cause the increase in average degree might be done with outsiders rather than co-members. Fortunately, this negative trend of suspicious ratio versus density consolidates our preferences for subgraph extraction methods since both ego networks and strongly connected components do not impose density by definition (Section 3.2). In other words, this relationship invalidates the intuition that meaningful communities would be dense subgroups and pave the way for novel subgraph extraction methods just like our suggested approach.

4.2. Experimental Setup

We designed an experimental setup to test our system according to three sampling preferences in terms of seed set selection while building strongly connected component subgraphs and ego networks under three different levels of risk aversion scenarios. The sampling preferences are introduced in our system because of deficiency of computational power and ideally all possible meaningful substructures would have been involved as candidate subgraphs.

First, we decided to select 20,000 centroids completely randomly from the reduced network and build their ego networks for analysis. Next, we ranked the node set in descending order of their degrees, meaning the number of connections they have, and sampled 20,000 from the top. High degree seed selection provides containment of more information. Finally, we extend the high degree selection method considering two

classes separately. We divide the node set into two parts with respect to their labels and then rank them in descending order of their degrees. We sample the top 10,000 nodes from each ranked list to be the centroids.

Each sampling procedure is repeated under three different risk aversion scenarios explained below:

(i) High Risk Aversion: The tolerance upper bound parameter is set to 0. This is the case when banks are not willing to take any risk of overlooking suspicious activity.

(ii) Moderate Risk Aversion: The tolerance upper bound parameter is set to 0.25. In this scenario, we assume banks conform with tolerating up to 25% presence of positive instances in designated groups.

(iii) Low Risk Aversion: The tolerance upper bound parameter is set to 0.50. This is the final case where we assume banks are most tolerant in the face of suspicious activities.

For sure, different sampling procedures that we suggest for seed selection lead to different samples corresponding to of diverse sets of candidate subgraphs. When it comes to distribution of class labels in each sample, we checked the ratio of positive observations over all observations in order to gain insight before evaluating the model outcomes. These ratios are provided in form of percentages in Table 4.3. Severity of the imbalance issue depends on the extent of unevenness between the counts of two class labels in particular samples. While slight imparities could be tolerated, in fact extreme differences between class ratios might provoke poor predictive performance and require appropriate treatment [71].

In our experimental setup, under each one of three scenarios, we see that ranking the nodes in descending order of their degrees boosts the selection probability of positive, namely suspicious, instances. Therefore, going from random selection to high degree, the percentages of positive observations increase in all scenarios.

For high risk aversion setup where tolerance upper bound equals to 0, we see that random sampling preserved the distribution of the original data, given in Section 4.1. On the other hand, sampling from degree ranked nodes result in pretty balanced samples in this setup. Recalling that in high risk aversion scenario we assign suspiciousness of ego networks with respect to their centroids, therefore they are identified by their seeds. So, we can deduce from the increase in percentages observed in Table 4.3 that suspicious nodes tend to be ranked high in decreasing order of node degrees. This is a valuable inference for discriminating between suspicious and innocent nodes on the individual basis depending on the number of connections they tend to have. In a nutshell, we can say that the sampling procedures we suggested alongside random sampling have the potential to straighten imbalance related problems.

Table 4.3. Percentage of Positive Observations in Experiments.

Seed Selection Approach	High Risk Aversion Scenario	Moderate Risk Aversion Scenario	Low Risk Aversion Scenario
Random Sampling	36.9%	63.1%	22.8%
High Degree Sampling	45.8%	73.8%	35.5%
High Degree Balanced Sampling	49.9%	75.6%	36.9%

Moderate risk aversion scenario is designed such that if the number of suspicious nodes present in a subgraph constitute more than 25% of the nodes, then the subgraph belongs in positive class. We see that for each of the three sampling procedures, the moderate risk aversion setup yields samples where the majority class is the suspicious one, again with an increasing rate from random sampling towards others. We can infer from these percentages in Table 4.3 and their comparison with the high risk aversion case that even though the centroids of ego networks are clear nodes, they tend to con-

nect with suspicious nodes within their 2-step-neighborhoods leading to a membership to a suspicious subgraph. Therefore, we see a significant increase in positive class percentages as opposed to zero tolerance case. Such an outcome contradicts with habitual data in real-life problems such as fraud detection where the positive class of interest is mostly the minority. Hence, the common problems of insufficient learning could subsist for the minority class of clear subgraphs in our case.

When a bank's attitude towards labeling of subgraphs revolves around moderate risk, their expectations should be thoroughly clarified because it is likely that the model can successfully develop predictive ability for the majority class only. However, if identification of the innocent class is equally awaited, possible class imbalance should be addressed through additional supply of computational capacity.

Finally, as the tolerance upper bound is set to 50% in low risk aversion setup, the threshold for deeming a candidate subgraph suspicious is pulled up. Unsurprisingly, the class of positive instances becomes the minority one in this scenario. As explained above, suggested sampling procedures can help augmenting number of instances in minority class up to rather tolerable levels of 35.5% or 36.9%. In this study we pay attention to these class distributions and introduce various evaluation metrics for proper interpretation of model performances in case of both balanced and imbalanced samples in Section 4.3.1.

4.3. Results

The evaluation metrics of our choice are thoroughly explained while beginning this section. Then we present the results of our experiments, followed by a discussion on those outcomes.

4.3.1. Evaluation Metrics

We consider customary machine learning metrics for evaluating the performance of our classifiers, including the area under receiver operating characteristic curve (AUC), area under precision-recall curve, accuracy, positive predictive rate (PPR), sensitivity and F1-score which is the weighted average of PPR and sensitivity.

First of all, accuracy is the most common evaluation metric for machine learning models, and it measures the ratio of correct predictions among all predictions in the test sample. Depending on the problem content, evaluating the model performance solely based on accuracy may be misleading since it serves as an average measure for correct predictions in both positive and negative classes. In the context of anti-money laundering, correct classification of fraud can be favored over correct classification of normal behavior in some cases. Therefore, we are making use of various evaluation metrics alongside accuracy.

Receiver operating characteristic (ROC) curve is a useful evaluation metric for binary classification tasks. It shows the false positive rate, namely positively predicted actual negative rate, on the horizontal axis against the true positive rate, that is accurately predicted positive value rate, on the vertical axis [72]. The true positive rate is the same as sensitivity and calculated as shown below:

$$\textbf{True positive rate} = \frac{\textbf{Correctly predicted actual positives}}{\textbf{All actual positives}}$$

False positive rate on the other hand, is an important metric that we aim to reduce in this study and shows incorrectly predicted actual negatives, so to speak, false alarms. Its calculation is given below:

$$\textbf{False positive rate} = \frac{\textbf{Incorrectly predicted actual negatives}}{\textbf{All actual negatives}}$$

The area under ROC curve (AUC) is actually a measure for discriminative ability of the classifier between two classes. It equals to 0.5 when the predictions are done completely randomly and increases as the performance of the classifier improves. We should keep in mind that in case of imbalanced class distribution in training data, utilization of ROC curve for evaluating classifier skill may turn out fallacious and hollowly provide satisfactory results [73]. That is why alternative ways to gauge performance of binary classifiers are taken into account in this study.

Precision, in other words positive predictive rate (PPR), is also a machine learning metric for evaluating performance of binary classification tasks. Basically, it indicates the model skill in predicting the positive labeled class and measures the classifier's exactness. High precision values indicate low false positive rate; hence imply that the classifier yields accurate positive predictions among all its predictions. It is calculated with the formula below:

$$\textbf{Precision} = \frac{\textbf{Correctly predicted positives}}{\textbf{All positive predictions}} = \frac{\textbf{True positives}}{\textbf{True positives} + \textbf{False positives}}$$

On the other hand, sensitivity, also referred as recall, is the measure of classifier's completeness. High recall values imply low false negative rates. This means that the classifier manages to correctly classify most of actually positive instances. The formula of recall or sensitivity is given below:

$$\textbf{Recall} = \frac{\textbf{Correctly predicted positives}}{\textbf{All actual positives}} = \frac{\textbf{True positives}}{\textbf{True positives} + \textbf{False negatives}}$$

There is also a Precision-Recall curve which is a better indicator of model skill for binary classification tasks with imbalanced class distributions [74]. Accuracy and ROC curve are not transparent in reflecting a model's skill in case of class imbalance due to consideration of true negatives while calculating false positive rate. In a sense, the more we are interested in the model's ability to predict the rare positive labels

correctly, the more precision and recall metrics matter. Larger values for area under Precision-Recall curve connote high levels of both precision and recall; indicating lower false positive and false negative rates respectively.

For the purposes of anti-money laundering, the positive class is intuitively selected as the suspicious class, therefore we are more interested in correct classification of suspicious subgraphs along with the discriminative ability of the model between innocent and suspicious. Accordingly, we are also interested in the area under Precision-Recall curve as a metric for the classifier's predictive ability for the positive class.

Finally, F1-score merges two metrics focusing on positive predictive skill of the classifier and it is found as the harmonic mean of precision and recall rates:

$$\textbf{F-1 Score} = \frac{\textbf{2} \times \textbf{Precision} \times \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

4.3.2. Experimental Results

We ran our suggested system under three levels of risk aversion scenarios, using three different seed selection approaches for centroids and three types of binary classifiers. We implemented grid search for hyper parameter tuning using 5-fold cross validation for evaluation for each classifier in each experiment. The experiments are carried out within a cloud environment via virtual machines. The features of the machine type we used were 8 vCPUs and 30 GB of memory. For each experiment setting, implementation of the system took approximately 8 hours. The results of our experiments show model performances obtained through the best parameter combination that grid search yields, in terms of selected evaluation metrics: Accuracy, precision, recall, F-1 score, and finally areas under receiver operating characteristic (ROC) and Precision-Recall curves. We present the results of each experimental case in the form of ROC and Precision-Recall curves as well.

Resulting values of considered evaluation metrics under high risk aversion scenario with random seed set selection are given below in Table 4.4 pursued by associated ROC and Precision-Recall curves in Figure 4.3.

Table 4.4. Experimental Results with Random Seeds and Tolerance = 0.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.755	0.79	0.44	0.56	0.76270	0.69548
XGBoost	0.754	0.79	0.44	0.56	0.76330	0.69697
LightGBM	0.754	0.79	0.44	0.56	0.76082	0.69640
Logistic Regression	0.727	0.83	0.31	0.45	0.72985	0.66710

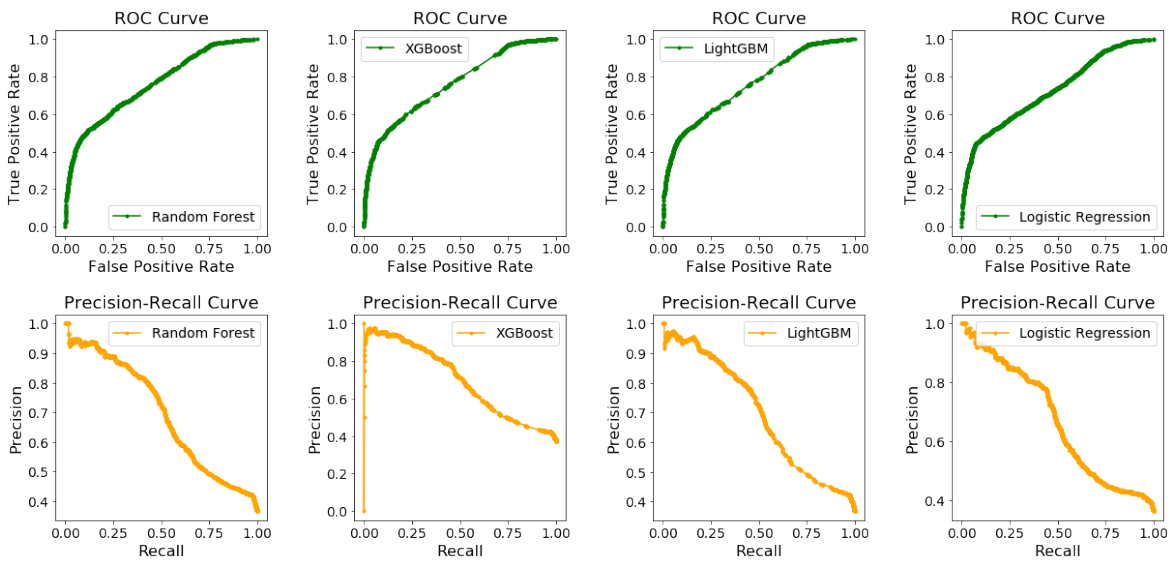


Figure 4.3. ROC and Precision-Recall Curves for Random Seed Sampling when Tolerance = 0.

Resulting values of considered evaluation metrics under high risk aversion scenario with high degree seed set selection are given below in Table 4.5 pursued by associated ROC and Precision-Recall curves in Figure 4.4.

Table 4.5. Experimental Results with High Degree Seeds and Tolerance = 0.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.793	0.90	0.63	0.74	0.86369	0.85803
XGBoost	0.791	0.89	0.63	0.74	0.86109	0.85877
LightGBM	0.790	0.89	0.63	0.74	0.86285	0.85973

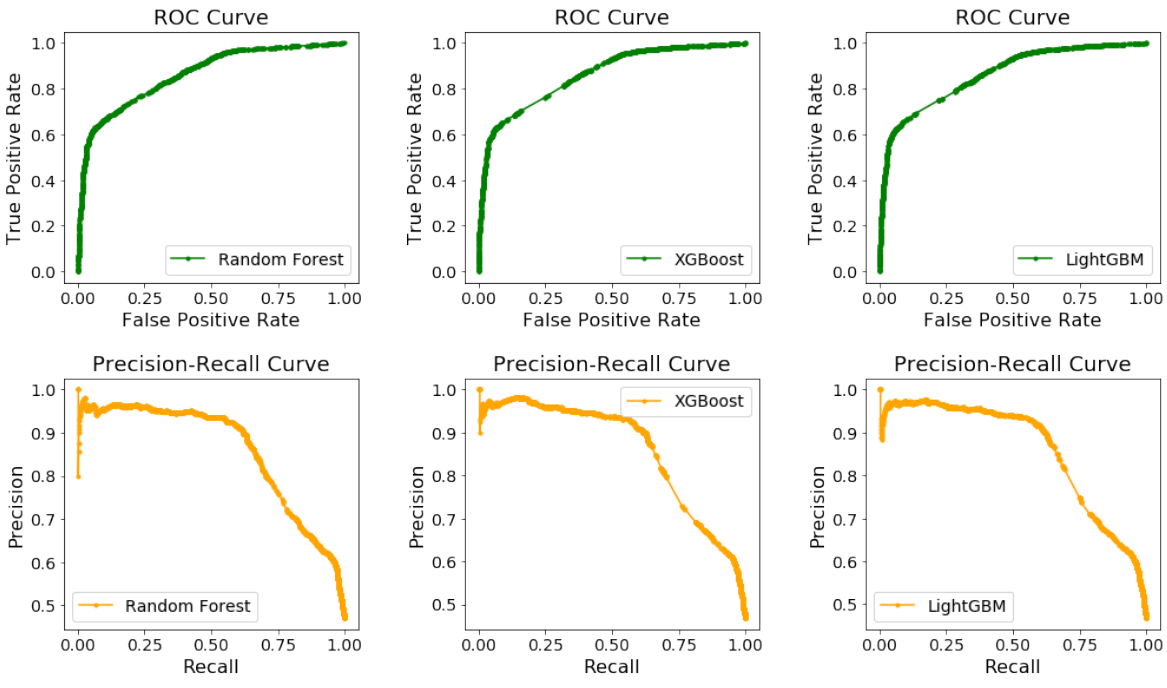


Figure 4.4. ROC and Precision-Recall Curves for High Degree Seed Sampling when Tolerance = 0.

Resulting values of considered evaluation metrics under high risk aversion scenario with equally sampled high degree seed set selection are given below in Table 4.6 pursued by associated ROC and Precision-Recall curves in Figure 4.5.

Table 4.6. Experimental Results with Equally Sampled High Degree Selected Seeds and Tolerance = 0.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.789	0.86	0.70	0.77	0.86429	0.86356
XGBoost	0.794	0.87	0.70	0.77	0.86615	0.86255
LightGBM	0.790	0.86	0.70	0.77	0.86555	0.86118

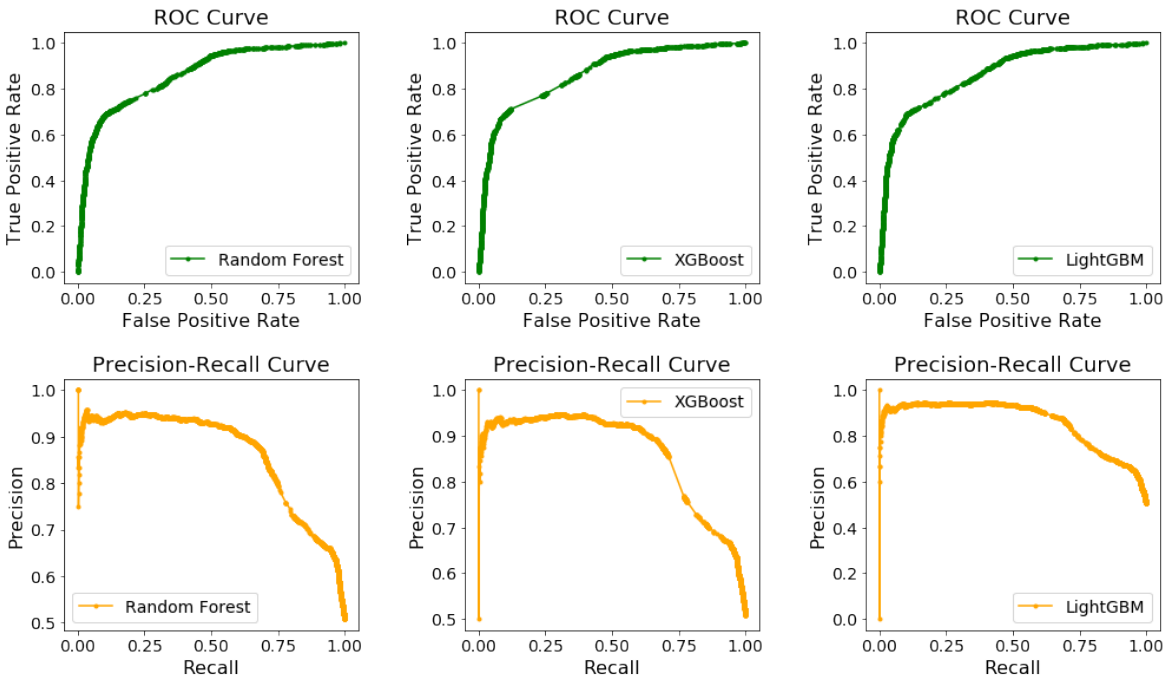


Figure 4.5. ROC and Precision-Recall Curves for High Degree and Balanced Seed Sampling when Tolerance = 0.

Resulting values of considered evaluation metrics under moderate risk aversion scenario with random seed set selection are given below in Table 4.7 pursued by associated ROC and Precision-Recall curves in Figure 4.6.

Table 4.7. Experimental Results with Randomly Selected Seeds and Tolerance = 0.25.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.694	0.69	0.94	0.79	0.74930	0.84456
XGBoost	0.726	0.75	0.86	0.80	0.78653	0.86387
LightGBM	0.705	0.71	0.91	0.80	0.76591	0.85645

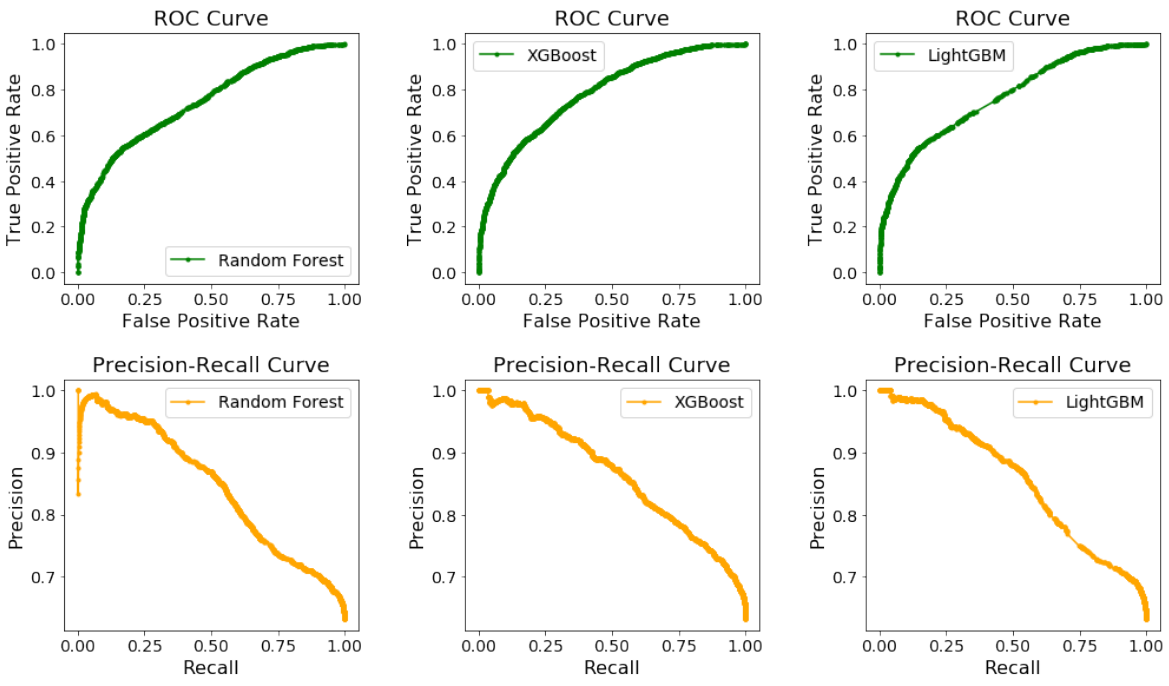


Figure 4.6. ROC and Precision-Recall Curves for Random Seed Sampling when Tolerance = 0.25.

Resulting values of considered evaluation metrics under moderate risk aversion scenario with high degree seed set selection are given below in Table 4.8 pursued by associated ROC and Precision-Recall curves in Figure 4.7.

Table 4.8. Experimental Results with High Degree Seeds and Tolerance = 0.25.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.756	0.76	0.97	0.85	0.79105	0.91155
XGBoost	0.783	0.80	0.94	0.86	0.81023	0.92069
LightGBM	0.777	0.79	0.95	0.86	0.80376	0.91895

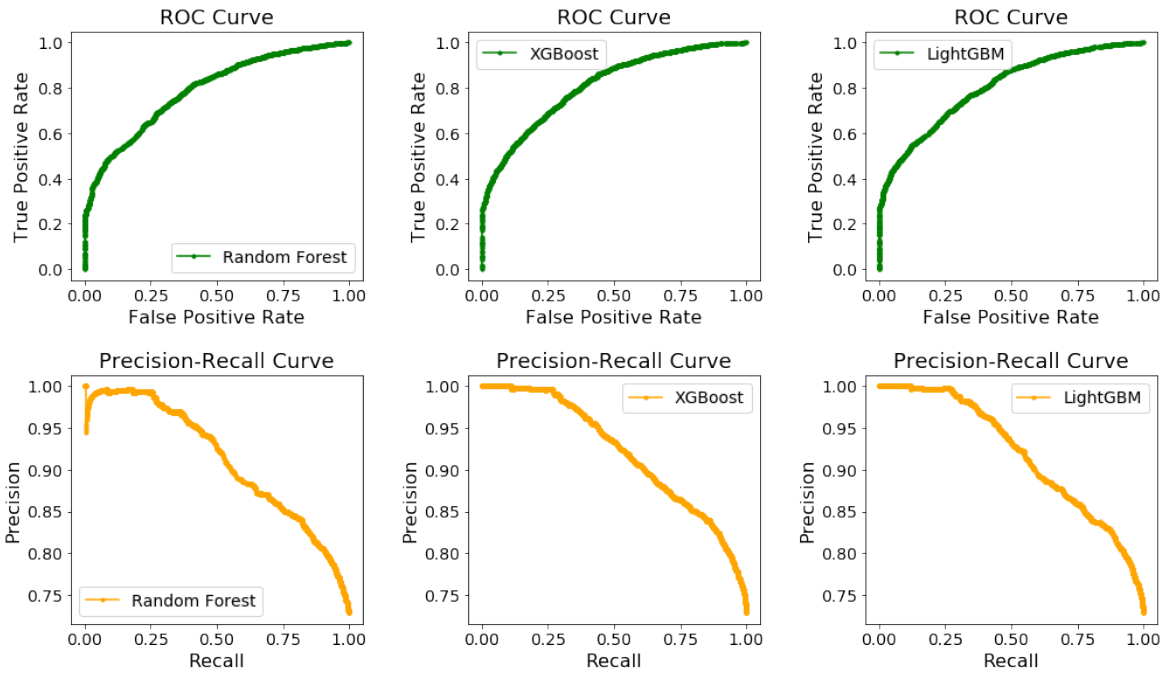


Figure 4.7. ROC and Precision-Recall Curves for High Degree Seed Sampling when Tolerance = 0.25.

Resulting values of considered evaluation metrics under moderate risk aversion scenario with equally sampled high degree seed set selection are given below in Table 4.9 pursued by associated ROC and Precision-Recall curves in Figure 4.8.

Table 4.9. Experimental Results with Equally Sampled High Degree Selected Seeds and Tolerance = 0.25.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.765	0.77	0.97	0.86	0.78322	0.92199
XGBoost	0.783	0.81	0.93	0.87	0.79948	0.92661
LightGBM	0.781	0.81	0.93	0.87	0.79087	0.92357

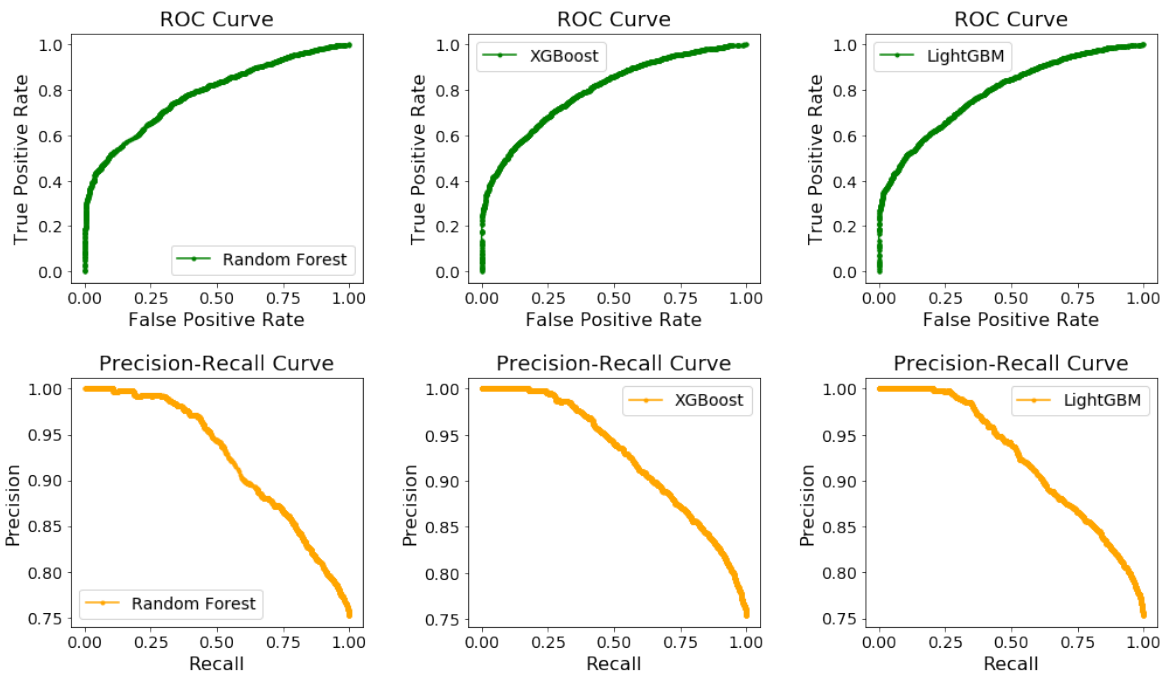


Figure 4.8. ROC and Precision-Recall Curves for High Degree and Balanced Seed Sampling when Tolerance = 0.25.

Resulting values of considered evaluation metrics under low risk aversion scenario with randomly sampled seeds are given below in Table 4.10 pursued by associated ROC and Precision-Recall curves in Figure 4.9.

Table 4.10. Experimental Results with Randomly Selected Seeds and Tolerance = 0.5.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.875	0.84	0.56	0.67	0.89908	0.79157
XGBoost	0.890	0.83	0.65	0.73	0.91249	0.83084
LightGBM	0.884	0.82	0.63	0.71	0.90752	0.81945

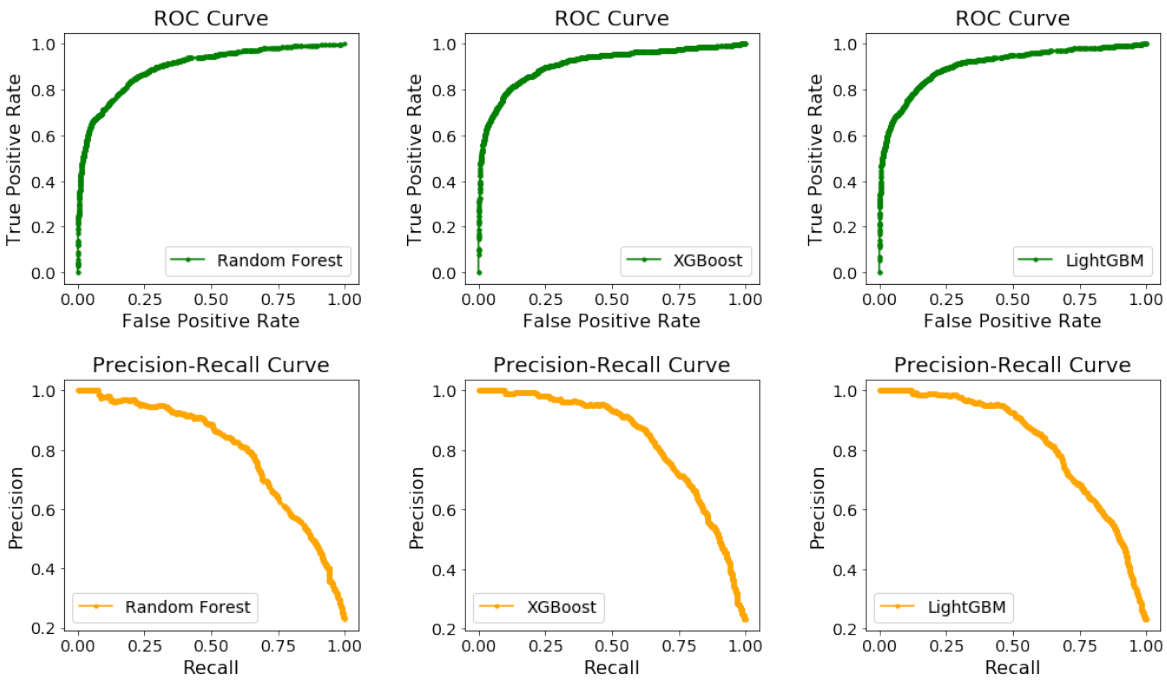


Figure 4.9. ROC and Precision-Recall Curves for Random Seed Sampling when Tolerance = 0.50.

Resulting values of considered evaluation metrics under low risk aversion scenario with high degree seed set selection are given below in Table 4.11 pursued by associated ROC and Precision-Recall curves in Figure 4.10.

Table 4.11. Experimental Results with High Degree Seeds and Tolerance = 0.5.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.826	0.85	0.62	0.72	0.86661	0.82905
XGBoost	0.830	0.84	0.64	0.73	0.87604	0.84456
LightGBM	0.830	0.85	0.63	0.72	0.87321	0.84060

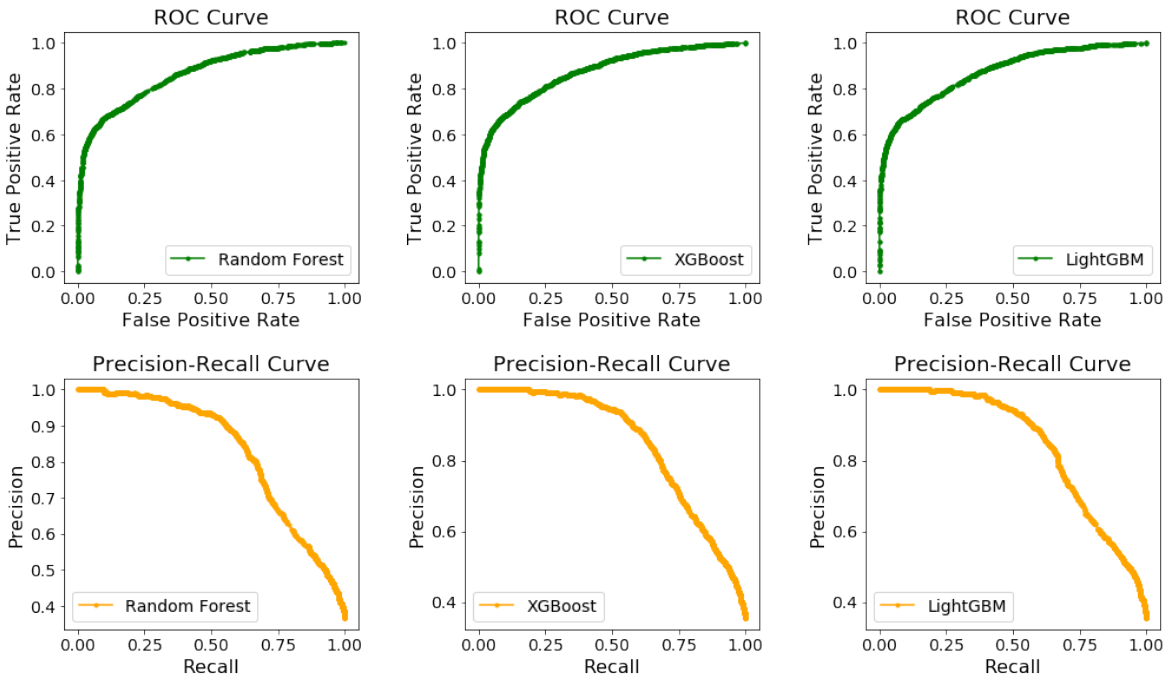


Figure 4.10. ROC and Precision-Recall Curves for High Degree Seed Sampling when Tolerance = 0.50.

Resulting values of considered evaluation metrics under low risk aversion scenario with equally sampled high degree seed set selection are given below in Table 4.12 pursued by associated ROC and Precision-Recall curves in Figure 4.11.

Table 4.12. Experimental Results with Equally Sampled High Degree Selected Seeds and Tolerance = 0.5.

Classifier	Accuracy	Precision	Recall	F-1 Score	Area under ROC Curve	Area under Precision-Recall Curve
Random Forest	0.828	0.86	0.65	0.74	0.87414	0.84269
XGBoost	0.833	0.86	0.66	0.75	0.87816	0.85234
LightGBM	0.833	0.86	0.66	0.75	0.87725	0.85157

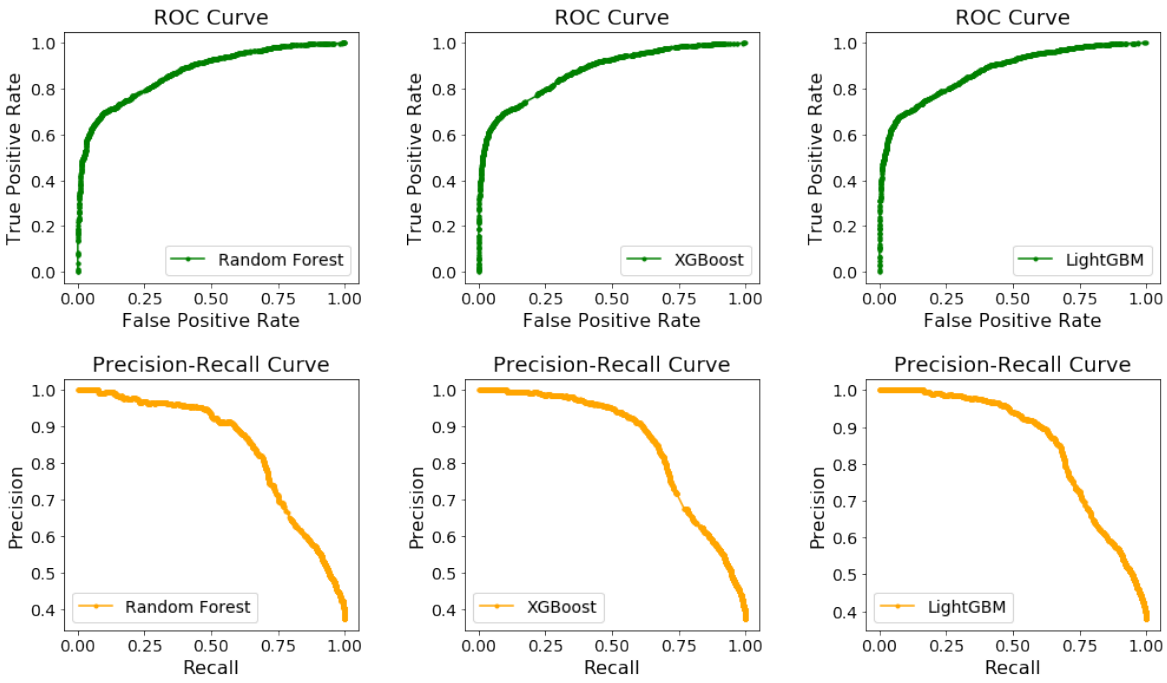


Figure 4.11. ROC and Precision-Recall Curves for High Degree and Balanced Seed Sampling when Tolerance = 0.50.

4.3.3. Discussion of Results

Evaluation of our experimental results with respect to various metrics are illuminating according to classifier choice and sampling procedure under specific risk-taking scenarios. In the beginning, we plan to make a comparison between the classifiers independently for each experimental case by means of meaningful evaluation metrics. This comparison is planned from a case-specific perspective since in each experiment candidate subgraphs created and labeled to train the classifiers are necessarily different. In each experiment, we set aside the test set beforehand, and tested three trained classifiers on the same test set.

First of all, the results of the high risk aversion scenario imply no clear performance related hierarchy among random forest, XGBoost and LightGBM classifiers whereas logistic regression falls behind apparently in terms of all considered evaluation metrics. From Table 4.4 and Figure 4.3, we can deduce that using randomly selected seeds in high risk aversion scenario, binary logistic regression is not able to compete other tree-based classifiers when both area under ROC and Precision-Recall curves are examined. Therefore, we decide upon excluding logistic regression from the comparative experiments to keep going with more sophisticated tree-based classifiers. In random sampling under this scenario, we compare the classifiers using precision and recall because of class imbalance as explained in 4.3.1. However, for other two sampling procedures, the class distributions shown in Table 4.3 are nearly even so that accuracy and area under ROC curve can reflect the model performances properly. In each sampling case, there are insignificant minor differences between three classifiers, shown in Table 4.4, Table 4.5 and Table 4.6 with respect to insightful metrics, which does not lead to any sufficient evidence about one specific classifier stepping forward.

When it comes to moderate risk aversion scenario, again considering uneven class distributions given in Table 4.3, we proceed with precision and recall for evaluating the classifiers. Accordingly, from Table 4.7, Table 4.8 and Table 4.9, we observe that XGBoost shines out among the classifiers under all three sampling procedures. For

random sampling, the interval between values of area under precision-recall curve are still trivial but relatively larger, where random forest comes out as the worst performer; however, in other sampling procedures, the differences seem even less significant. Even based on these minor differences, XGBoost can be appointed as the best performer in each one of the sampling procedures in the face of moderate risk aversion attitude.

This time in low risk aversion scenario, the class imbalance leads us to evaluate the classifiers using precision and recall where we observe that again XGBoost shines out as the best performer with an 83% area under precision-recall curve value, given in Table 4.10, for random sampling whereas random forest yields 79% for the same metric. Just like the previous scenario, XGBoost holds its premier rank in other two sampling methods as well while the gap between relevant metrics are narrower between LightGBM and XGBoost. Both result in almost 85% in high degree balanced sampling and 84% in high degree sampling whereas random forest ranks last with area under precision-recall curve levels of 84% and 83% respectively. At this junction, surely depending on risk-taking attitude, we can suggest XGBoost for a binary classification task of subgraphs, which has been advantageous with its performance in each experiment of this study. Proceeding from this declaration, we step into a broader comparison between suggested sampling procedures again independently under each risk-related scenario.

Recalling that in each experimental case, our classifiers learn from distinct candidate subgraph samples with different class ratios; it would be sane to reconcile, so to say, habitats of these models to set the ground for a comparison between sampling methods. In this case, overall evaluation metrics may be misleading because of varying target distributions in our experimental samples. Therefore, we choose to assess the positive predictive rate, i.e. precision, of XGBoost classifier for the top 20 percentile ranked according to prediction probabilities. This means that this once the comparison concentrates on the instances in top 20 percentile, which the classifier predicts suspicious with the highest probability. Such a metric for the comparison in question seems quite informative in terms of false positive rate of the subgraphs deemed most

probable suspects by trained model. Here, it is appropriate to remind of our goal of reducing the false positive rate to relieve labor's burden. Additionally, we settled a unique test sample for the sake of a plausible comparison between sampling methods. For that purpose, disjointly under each scenario, we tested both XGBoost classifiers trained with different samples obtained with two other seed selection methods on the test set of random sampling experiment. The precision scores for top 20 percentile produced as a result of these trials are given in Table 4.13:

Table 4.13. Positive Predictive Rates for Top 20 Percentiles.

Seed Selection Approach	High Risk Aversion Scenario	Moderate Risk Aversion Scenario	Low Risk Aversion Scenario
Random Sampling	78.75%	94.12%	79.00%
High Degree Sampling	78.00%	92.75%	75.12%
High Degree Balanced Sampling	79.12%	88.88%	75.88%

Primarily under high risk aversion scenario, scores in Table 4.13 show that an XGBoost classifier trained by a sample resulting from high degree balanced sampling yields a better prediction of 79.12%, thus a lower false positive rate when tested using the same sample as the other two cases. It is followed by random sampling, and lastly high degree sampling. This means that if a bank's presumed attitude towards taking risk falls under this scenario and the extracted subgraphs are labeled accordingly, ranking the nodes in descending order of their degrees and sampling from the top for training a classifier would be the most rewarding way as suggested by our system.

As regards to both moderate and low risk aversion scenarios, we see that classifiers score the highest in case of random sampling for seed selection. For a moderately risk-

averse attitude, we can sort sampling procedures from random scoring 94.12%, followed by high degree to high degree balanced with precision level of 88.88%. However, for low risk aversion scenario, we see that the order changes while the best performer remains random sampling with 79% precision. In a word, when preferences change towards tolerating risk, we observe that randomly sampling the seeds of subgraphs leads to lower false positive rates when we test suitably trained models with unfamiliar data. On the other hand, when the tolerance upper bound is set to zero, equally sampling from both classes of labeled nodes with a higher number of connections turns out preferable for a better predictive ability for truly suspicious subgraphs.

While navigating between assumed risk-related scenarios, we should consider the class distributions of target before jumping into any inferences. As mentioned above, the models are tested with a fixed data acquired via random sampling in each trial in Table 4.13. For random sampling, positive class constitutes 36.9% of all instances in high risk aversion scenario. In moderate and low risk aversion scenarios, this ratio becomes 63.1% and 22.8% respectively (Table 4.3). This means that without any machine learning practice, a simple predicting probability for a positive instance would have been 36.9% in the first scenario, 63.1% in the second and similarly 22.8% in the third. To assess the extent of contribution, we hold forth the best practices appointed above under each scenario and gauge their skill in proportion to a baseline prediction. As given in Table 4.13, in high risk aversion, our suggested system scales these estimated predicting probabilities up to 79.12%, nearly 2.1 times itself, whereas the improvement is 1.5 times in moderate case and lastly 3.5 times in low risk aversion. To put it another way, the system suggested in this study provides a predictive ability of 2.1 times the baseline for the first scenario, while improving it up to at most 1.5 times the baseline in second and showing off an increase to 3.5 times in the final case. Hence, we can declare that our suggested system fulfills its potential the most when banks or money laundering experts tend to tolerate risk more whereas it still makes a worthwhile contribution with the suitable sampling method when any risk is certainly avoided. When it comes to moderate willingness levels to take risk, despite a satisfactory prediction score of 94.12% for top 20 percentile, our system can raise the baseline

score at most by half itself since the labeling procedure boosts the positive labels in this scenario and suspicious subgraphs constitute the majority class.

Table 4.14. Important Features Ranked from Top to Bottom.

Features in Order of Importance
Number of Incoming Transactions
Volume of Outgoing Transactions
Number of Outgoing Transactions
Average Shortest Path Length
Inside Transaction Volume (Weighted Sum of Edges)
Average Degree
Density
Diameter
Radius
Number of Inside Transactions (Number of Edges)
Number of Nodes
Average Clustering Coefficient

Finally, the features fed into the models are ranked in descending order of their importance in Table 4.14 according to information gain, which translates to decrease in entropy. In this study, feature elimination is not prioritized, since all suggested classifiers are tree-based models in which calculation of feature importance is essential while building the tree. In principle, these models try to hold the most important features on the top [67]. The relations of top 6 features with suspicious subgraphs except for the ones based on volume are examined in Section 4.1. In the end, each one of these outweighing features happen to be positively or negatively correlated with the response, which validates their contribution in learning. Thus, ameliorative potential of network-based features in anti-money laundering, serving as the driving force behind this study, is sustained.

5. CONCLUSION

In this study, we propose an anti-money laundering framework applicable to bank transaction networks. At first, we achieved a considerable reduction in complexity while clinging to valuable, i.e. informative components of the network. Afterwards, we introduced a peculiar approach for extracting subgraphs in the domain of anti-money laundering. This way, we recovered the disposition of subgraphs from imperatively high density under domination of traditional community detection studies. We suggest seeking subgraphs through both local neighborhood-based search and typological quest specifically for circulation of funds inside a money laundering network.

Furthermore, since this study is motivated by potential benefits of network-based analysis in identification of suspiciousness, conceptually and structurally meaningful characteristics of extracted subgraphs are integrated into machine learning techniques. We formulated the problem as a supervised binary classification problem, for which we experimented with different tree-based classifiers under 3 scenarios reflecting possible demeanors of banks or experts in the face of money laundering risks. On account of computational restrictions in the forefront among the challenges encountered during this study, we developed three sampling procedures in order to filter samples as insightful as possible. Because building the subgraphs alongside the calculation of their network-based features are indeed computationally the most expensive stages in our system. We introduced appropriate evaluation metrics considering varying class ratios in our experimental cases and tested our results accordingly. Among the classifiers, XGBoost outperformed loosely random forest and LightGBM throughout our experiments. On top of that, we observed that different sampling procedures, which we introduced as a remedy for occasional computational deficiencies, may be preferred under different risk aversion scenarios. In a sense, prioritizing the accounts involving in more transactions in sampling is advantageous for our framework under high risk aversion scenario. On the other hand, randomly sampling seed accounts becomes favorable in other two scenarios among which, our system turns out the most promising

for an improvement in predictive ability up to 3.5 times the baseline under low risk aversion.

Overall, we suggest a system to ease the money laundering analysts' burden caused by high false positive rate, which proves to be promising by the look of our experimental results based on an accessible social network data including real spammers. Our framework realizes initially hypothesized potential of network-based analysis in anti-money laundering research while yielding satisfactory detection performance along with considerable decline of false positive rate.

By all means, this study faced the inherent challenges relevant to both the domain of anti-money laundering and working with massive networks. Inaccessibility of real bank transaction networks proves to be the most compelling aspect in anti-money laundering because of legal liabilities and privacy-related concerns. Such a challenge paves the way for further research in terms of the ways to generate synthetic networks as realistically as possible or the ways to fill missing data's shoes within the bounds of accessibility. After all, generation of plausibly realistic synthetic data forms a field of research on its own. Moreover, efforts in complexity reduction in networks without loss of valuable information provides a basis for further research on its own.

As mentioned above, we can deduce from the computationally expensive experience of construction of subgraphs followed by calculation of their network features that these stages are also open for further improvement. Computational challenges in extraction of substructures from a large network still needs to be addressed by algorithmic efficiency. Once again, the scope of this study comes up to identification of suspicious subgraphs via machine learning techniques. The manner of reporting these findings to the banks would preferably be on the basis of accounts, rather than subgraphs. At this point, developing a reporting procedure to distinguish the most probably suspicious accounts among all members of positive predictions while dealing with the overlap between appointed subgraphs for the sake of efficiency remains open research problem.

REFERENCES

1. Schott, P. A., *Reference guide to anti-money laundering and combating the financing of terrorism*, The World Bank, 2006.

2. United Nations Office on Drugs and Crime Global Programme Against Money Laundering, *Money Laundering and the Financing of Terrorism: The United Nations Response*, https://www.imolin.org/pdf/imolin/UNres03e.pdf, accessed in July 2020.

3. Yepes, C. V., *International Cooperation in the Fight against Terrorist Financing*, Ph.D. Thesis, Universitat de Barcelona, 2008.

4. Schneider, F. and R. Caruso, *The (hidden) financial flows of terrorist and transnational crime organizations: a literature review and some preliminary empirical results*, Tech. rep., Economics of Security Working Paper, 2011.

5. *Turkish Penal Code Article 282 Law No: 5237*, 2004.

6. Republic of Turkey Ministry of Treasury and Finance, *The Financial Crimes Investigation Board*, https://en.hmb.gov.tr/fcib-presentation, accessed in July 2020.

7. *Law on Prevention of Laundering Proceeds of Crime Law No: 5549*, 2016.

8. *Financial Action Task Force (FATF)*, http://www.fatf-gafi.org/about/whatwedo/, accessed in July 2020.

9. Financial Action Task Force On Money Laundering, *Report on Money Laundering Typologies*, 2004, https://www.fatf-gafi.org/media/fatf/documents/reports/2003_2004_ML_Typologies_ENG.pdf, accessed in July 2020.

10. United Nations Office on Drugs and Crime, *Estimating Illicit Financial Flows Resulting from Drug Trafficking and Other Transnational Organized Crimes*, 2011, `https://www.unodc.org/documents/data-and-analysis/Studies/Illicit_financial_flows_2011_web.pdf`, accessed in July 2020.

11. Senator, T. E., H. G. Goldberg, J. Wooton, M. A. Cottini, A. U. Khan, C. D. Klinger, W. M. Llamas, M. P. Marrone and R. W. Wong, "Financial crimes enforcement network AI system (FAIS) identifying potential money laundering from reports of large cash transactions", *AI Magazine*, Vol. 16, No. 4, p. 21, 1995.

12. Gao, Z. and M. Ye, "A framework for data mining-based anti-money laundering research", *Journal of Money Laundering Control*, Vol. 10, No. 2, pp. 170–179, 2007.

13. Watkins, R. C., K. M. Reynolds, R. Demara, M. Georgiopoulos, A. Gonzalez and R. Eaglin, "Tracking dirty proceeds: exploring data mining technologies as tools to investigate money laundering", *Police Practice and Research*, Vol. 4, No. 2, pp. 163–178, 2003.

14. Kingdon, J., "AI fights money laundering", *IEEE Intelligent Systems*, Vol. 19, No. 3, pp. 87–89, 2004.

15. Savage, D., Q. Wang, X. Zhang, P. Chou and X. Yu, "Detection of money laundering groups: Supervised learning on small networks", *The Workshops of the Thirty-First AAAI Conference on Artificial Intelligence: The AAAI-17 Workshop on AI and Operations Research for Social Good*, pp. 43–49, 2017.

16. Le-Khac, N.-A., S. Markos and M.-T. Kechadi, "Towards a new data mining-based approach for anti-money laundering in an international investment bank", *International Conference on Digital Forensics and Cyber Crime*, pp. 77–84, Springer, 2009.

17. Lv, L.-T., N. Ji and J.-L. Zhang, "A RBF neural network model for anti-money laundering", *2008 International Conference on Wavelet Analysis and Pattern Recognition*, Vol. 1, pp. 209–215, 2008.

18. Heidarinia, N., A. Harounabadi and M. Sadeghzadeh, "An Intelligent Anti-Money Laundering Method for Detecting Risky Users in the Banking Systems", *International Journal of Computer Applications*, Vol. 97, pp. 35–39, 2014.

19. Jullum, M., A. Løland, R. B. Huseby, G. Ånonsen and J. Lorentzen, "Detecting money laundering transactions with machine learning", *Journal of Money Laundering Control*, Vol. 23, No. 1, pp. 173–186, 2020.

20. Akoglu, L., H. Tong and D. Koutra, "Graph based anomaly detection and description: a survey", *Data mining and knowledge discovery*, Vol. 29, No. 3, pp. 626–688, 2015.

21. Silva, T. C. and L. Zhao, *Machine learning in complex networks*, Springer, 2016.

22. Bayer, I., U. Nagel and S. Rendle, "Graph based relational features for collective classification", *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 447–458, Springer, 2015.

23. Blum, J. A., M. Levi, R. T. Naylor and P. Williams, "Financial havens, banking secrecy and money laundering", *Trends in Organized Crime*, Vol. 4, No. 4, pp. 68–71, 1999.

24. Samanvi, K. and N. Sivadasan, "Subgraph Similarity Search in Large Graphs", *arXiv preprint arXiv:1512.05256*, 2015.

25. Li, Y., C. Gu, T. Dullien, O. Vinyals and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects", *arXiv preprint arXiv:1904.12787*, 2019.

26. Michalak, K. and J. Korczak, "Graph mining approach to suspicious transaction detection", *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 69–75, IEEE, 2011.

27. ACAMS, *AML Glossary of Terms*, `www.acams.org/aml-glossary/`, accessed in July 2020.

28. Lopez-Rojas, E. A. and S. Axelsson, "A review of computer simulation for fraud detection research in financial datasets", *2016 Future Technologies Conference (FTC)*, pp. 932–935, IEEE, 2016.

29. Lopez-Rojas, E. A. and S. Axelsson, "Social simulation of commercial and financial behaviour for fraud detection research", *Social Simulation Conference*, 2014.

30. Lopez-Rojas, E., A. Elmir and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection", *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, pp. 249–255, Dime University of Genoa, 2016.

31. Lopez-Rojas, E. A. and S. Axelsson, "Money laundering detection using synthetic data", *Annual workshop of the Swedish Artificial Intelligence Society (SAIS)*, Linköping University Electronic Press, Linköpings universitet, 2012.

32. Fakhraei, S., J. Foulds, M. Shashanka and L. Getoor, "Collective spammer detection in evolving multi-relational social networks", *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pp. 1769–1778, 2015.

33. Zheng, X., Z. Zeng, Z. Chen, Y. Yu and C. Rong, "Detecting spammers on social networks", *Neurocomputing*, Vol. 159, pp. 27–34, 2015.

34. Lindner, G., C. L. Staudt, M. Hamann, H. Meyerhenke and D. Wagner, "Structure-preserving sparsification of social networks", *2015 IEEE/ACM international conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp.

448–454, IEEE, 2015.

35. Satuluri, V., S. Parthasarathy and Y. Ruan, "Local graph sparsification for scalable clustering", *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 721–732, 2011.

36. Nick, B., C. Lee, P. Cunningham and U. Brandes, "Simmelian backbones: Amplifying hidden homophily in facebook networks", *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, pp. 525–532, 2013.

37. Leskovec, J. and C. Faloutsos, "Sampling from large graphs", *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 631–636, 2006.

38. Fortunato, S., "Community detection in graphs", *Physics Reports*, Vol. 486, No. 3-5, pp. 75–174, 2010.

39. Newman, M. E., "Community detection and graph partitioning", *EPL (Europhysics Letters)*, Vol. 103, No. 2, p. 28003, 2013.

40. Rosvall, M., J.-C. Delvenne, M. T. Schaub and R. Lambiotte, "Different approaches to community detection", *arXiv preprint arXiv:1712.06468*, 2017.

41. Bansal, N., A. Blum and S. Chawla, "Correlation clustering", *Machine Learning*, Vol. 56, No. 1-3, pp. 89–113, 2004.

42. Kappes, J. H., P. Swoboda, B. Savchynskyy, T. Hazan and C. Schnörr, "Multicuts and perturb & MAP for probabilistic graph clustering", *Journal of Mathematical Imaging and Vision*, Vol. 56, No. 2, pp. 221–237, 2016.

43. Newman, M. E., "Detecting community structure in networks", *The European Physical Journal B*, Vol. 38, No. 2, pp. 321–330, 2004.

44. Newman, M. E., "Fast algorithm for detecting community structure in networks", *Physical Review E*, Vol. 69, No. 6, p. 066133, 2004.

45. Yang, J. and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach", *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pp. 587–596, 2013.

46. Baumes, J., M. Goldberg and M. Magdon-Ismail, "Efficient identification of overlapping communities", *International Conference on Intelligence and Security Informatics*, pp. 27–36, Springer, 2005.

47. Page, L., S. Brin, R. Motwani and T. Winograd, *The PageRank citation ranking: Bringing order to the web.*, Tech. rep., Stanford InfoLab, 1999.

48. Li, Y., K. He, D. Bindel and J. E. Hopcroft, "Uncovering the small community structure in large networks: A local spectral approach", *Proceedings of the 24th International Conference on World Wide Web*, pp. 658–668, 2015.

49. Leskovec, J., K. J. Lang, A. Dasgupta and M. W. Mahoney, "Statistical properties of community structure in large social and information networks", *Proceedings of the 17th International Conference on World Wide Web*, pp. 695–704, 2008.

50. Gonzalez-Pardo, A., J. J. Jung and D. Camacho, "ACO-based clustering for Ego Network analysis", *Future Generation Computer Systems*, Vol. 66, pp. 160–170, 2017.

51. Crossley, N., E. Bellotti, G. Edwards, J. Koskinen, M. Tranmer and M. Everett, *Social Network Analysis for Ego-Nets*, Sage Publications Ltd, United Kingdom, 2015.

52. Xie, J., S. Kelley and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study", *ACM Computing Surveys*, Vol. 45, No. 4, pp. 1–35, 2013.

53. Rees, B. S. and K. B. Gallagher, "EgoClustering: overlapping community detection via merged friendship-groups", *The Influence of Technology on Social Network Analysis and Mining*, pp. 1–20, Springer, 2013.

54. Rossetti, G., "Exorcising the Demon: Angel, Efficient Node-Centric Community Discovery", *International Conference on Complex Networks and Their Applications*, pp. 152–163, Springer, 2019.

55. Sengupta, S., "Anomaly detection in static networks using egonets", *arXiv preprint arXiv:1807.08925*, 2018.

56. Akoglu, L., M. McGlohon and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs", *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 410–421, Springer, 2010.

57. Yan, B. and S. Gregory, "Detecting communities in networks by merging cliques", *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, Vol. 1, pp. 832–836, IEEE, 2009.

58. *H3M.IO - Artificial Intelligence in Compliance*, `https://h3m.io/main-1`, accessed in July 2020.

59. Freeman, L. C., "Centrality in social networks conceptual clarification", *Social Networks*, Vol. 1, No. 3, pp. 215–239, 1978.

60. Needham, M. and A. Hodler, *Graph Algorithms: Practical Examples In Apache Spark And Neo4j*, O'Reilly Media, Inc., 2019.

61. Watts, D. J. and S. H. Strogatz, "Collective dynamics of 'small-world'networks", *Nature*, Vol. 393, No. 6684, pp. 440–442, 1998.

62. Dijkstra, E. W., "A note on two problems in connexion with graphs", *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.

63. Opsahl, T., F. Agneessens and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths", *Social Networks*, Vol. 32, No. 3, pp. 245–251, 2010.

64. Kuhn, M. and K. Johnson, *Applied Predictive Modeling*, Vol. 26, Springer, 2013.

65. Krawczyk, B., "Learning from imbalanced data: open challenges and future directions", *Progress in Artificial Intelligence*, Vol. 5, No. 4, pp. 221–232, 2016.

66. He, H. and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, John Wiley & Sons, 2013.

67. Hastie, T., R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Science & Business Media, 2009.

68. Chen, T. and C. Guestrin, "Xgboost: A scalable tree boosting system", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

69. Friedman, J. H., "Greedy function approximation: a gradient boosting machine", *Annals of Statistics*, pp. 1189–1232, 2001.

70. Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree", *Advances in Neural Information Processing Systems*, pp. 3146–3154, 2017.

71. Fernández, A., S. García, M. Galar, R. Prati, B. Krawczyk and F. Herrera, *Learning from Imbalanced Data Sets*, Springer International Publishing, 2018.

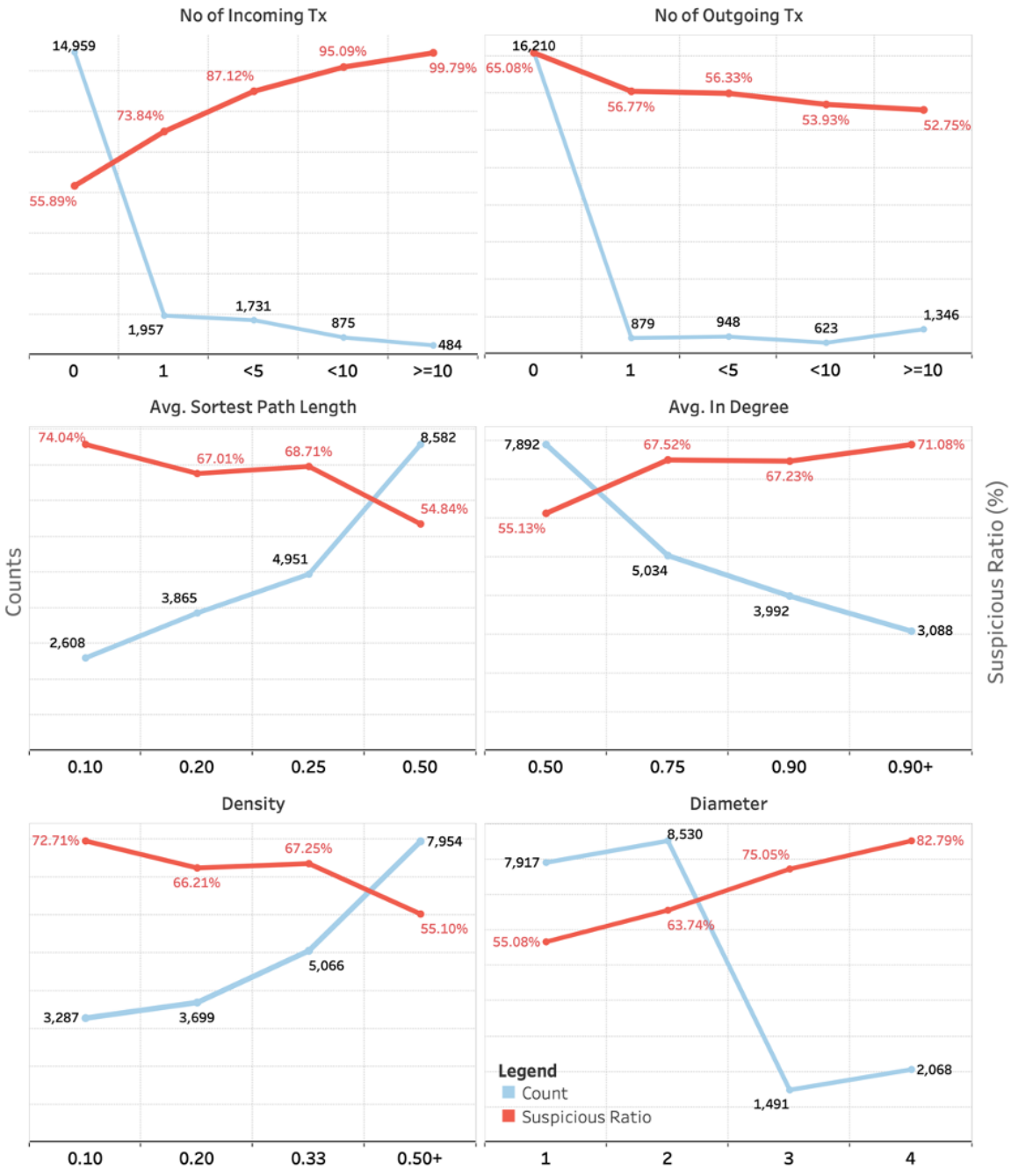
72. Fawcett, T., "An introduction to ROC analysis", *Pattern Recognition Letters*, Vol. 27, No. 8, pp. 861–874, 2006.

73. Davis, J. and M. Goadrich, "The relationship between Precision-Recall and ROC

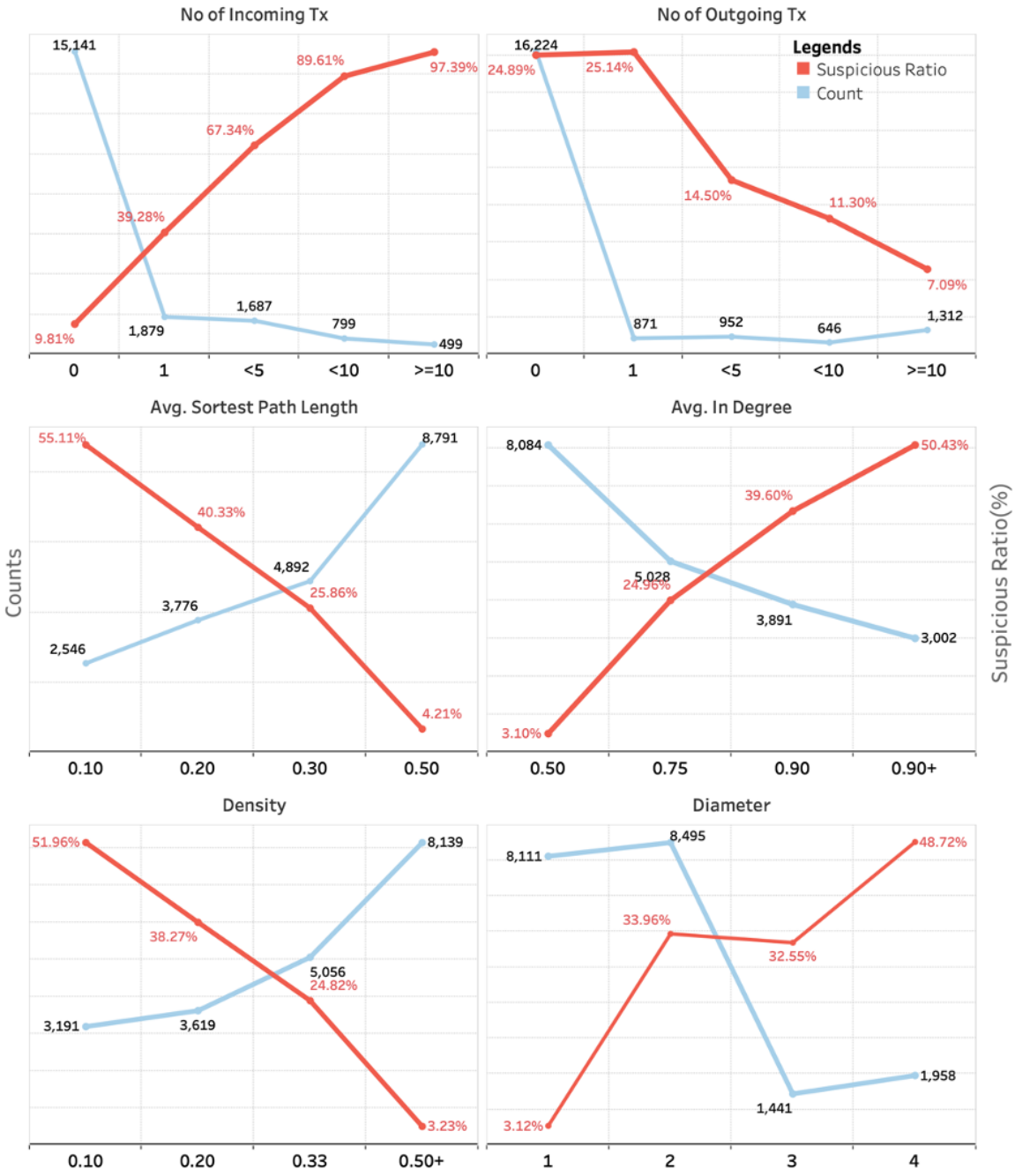
curves", *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, 2006.

74. Saito, T. and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets", *PloS One*, Vol. 10, No. 3, pp. 1–21, 2015.

APPENDIX A: THE RELATIONSHIP BETWEEN SUSPICIOUS SUBGRAPH RATIO AND IMPORTANT FEATURES FOR MODERATE RISK AVERSION SCENARIO



APPENDIX B: THE RELATIONSHIP BETWEEN SUSPICIOUS SUBGRAPH RATIO AND IMPORTANT FEATURES FOR LOW RISK AVERSION SCENARIO



APPENDIX C: AN EXAMPLE PYTHON CODE FOR HIGH RISK AVERSION SCENARIO WITH RANDOM SEED SELECTION

```
# Gizem Tas Thesis Experimentation Sample No. 1: Tagged.com & High Risk Aversion
# This anonymized dataset was sampled from the dataset collected from Tagged.com social network
#   website. It includes 119230 users among which 41981 are known spammers and 106851 edges between
#   them indication one specific type of action.
#
# A directed graph will be formed using these edges as the main object of analysis throughout
#the steps below.
#
# Remark: The tolerance percentage for suspecting a subgraph is set to 0 in this setup, meaning
#high risk aversion.

#Required libraries and methods
from datetime import datetime
import networkx as nx
from random import randint
from random import random
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import itertools
from collections import deque
from networkx.algorithms import community as community_louvain
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, make_scorer, accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, KFold, cross_validate
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier

#Ticking the clock for measuring the execution time of the entire work
from datetime import datetime
start_time = datetime.now()

#Importing the pickled data
taggedcom_data = pd.read_pickle('/Users/gizemtas/Desktop/THESIS/Datasets/relations.p')

#Filtering the first sample of users and their relations from the entire data
user_sample_1 = taggedcom_data[0]
relation_sample_1 = taggedcom_data[3]

#Modeling the assumed transactions from the relations sample as a directed network where source
#nodes are taken from the
#"source" column and target nodes from "destination" column of the dataframe. The consructed
#network is a primitive
#version, thus called "G_raw", in order to be reduced using various assumptions in the
#following steps.
G_raw = nx.from_pandas_edgelist(relation_sample_1, "source", "destination",
        create_using = nx.DiGraph())
```

```
nx.info(G_raw)
#Note: Because for every tie in the edgelist of a directed network there is a sender and a receiver,
#any attempt tp calculate

#the average in- or out-degree result in the same answer.
print(nx.info(G_raw))

import networkit as nk
nkG = nk.nxadapter.nx2nk(G_raw)
nk.overview(nkG)

#APPROXIMATION OF REAL-LIFE TRANSACTIONAL NETWORKS
#Weights will be assigned to each edge to imitate a financial transaction between two parties.
#The gamma distribution of the financial transaction amounts is verified by means of a simulated
#transaction sample and the shape and scale parameters of gamma distribution are estimated using
#the same sample. Definition of a function for adding gamma distributed weights to the edges
def tx_amount_update(e, alpha, beta):
        amount = np.random.gamma(shape = alpha, scale = beta)
        G_raw.edges[e]['weight'] = amount

#Associating gamma distributed weights to each edge if the given graph is not already weighted
#The parameters of the distribution are estimated using the generated transaction values
# by PaySim simulator.

if not nx.is_weighted(G_raw):
    alpha = 0.08871757178793419
    beta = 2027358.8672676312
    list(map(lambda x: tx_amount_update(x, alpha, beta), list(G_raw.edges)))

#PREPROCESSING (NETWORK REDUCTION): FILTERING OF THE EDGES
#Removal of the transactions having insignificant amounts
#amount lower bound is also estimated from the sample distribution of the synthetic data
amount_lower_bound = 10 #Approximately 35% of the total transactions could be eliminated this way.
small_tx_volumes = [e for e in list(G_raw.edges) if G_raw.edges[e]['weight'] < amount_lower_bound]
G_raw.remove_edges_from(small_tx_volumes)
nx.info(G_raw)

#PREPROCESSING: NETWORK REDUCTION OF THE NODES
#Removal of isolated nodes with no connections
#Note: In the networks constructed from and edgelist, no isolated nodes shold exist since each
#node is either the source or the target of a transaction.
#G_raw.remove_nodes_from(list(nx.isolates(G_raw)))

#PREPROCESSING: NETWORK REDUCTION OF THE NODES CONTINUED
#Removal of the nodes with extremely high degrees,
#which can be assumed to be institutions such as power administrations that have nothing to
#do with money laundering

node_degrees = sorted([d for n, d in G_raw.degree()], reverse=True)
degree_upper_bound = np.percentile(node_degrees, 99.9)
nodes_to_be_removed = [n for n in list(G_raw.nodes) if G_raw.degree[n] > degree_upper_bound]
G_raw.remove_nodes_from(nodes_to_be_removed)
#Removal of the nodes with a single link, mainly to simplify the massive network.
#The reason behind this simplification is that bank accounts active in only one transaction
#can be assumed insignificant.
single_edge_nodes = [n for n in list(G_raw.nodes) if G_raw.degree[n] < 1]
G_raw.remove_nodes_from(single_edge_nodes)

#THE REDUCED NETWORK CALLED G
G = G_raw.copy()
nx.info(G)
print(nx.info(G))

#THIS STEP SHOULD BE EXECUTED IN CASE THE REAL LABELS FOR FRAUDS OR SPAMS ARE NOT AVAILABLE
```

```
#LABELLING THE NODE DATA AS SUSPICIOUS, CLEAR & UNKNOWN REALISTICALLY
#Definition of a function for flagging part of the nodes with a predefined probability
def arbitrary_status_update(n, flag_rate, clear_rate):
    prob = random()
    if prob <= flag_rate:
        G.node[n]['status'] = 'SAR'
    elif prob <= flag_rate + clear_rate:
        G.node[n]['status'] = 'clear'
    else:
        G.node[n]['status'] = 'unknown'
#Tagging part of the nodes as suspicious, part of the rest as clear with a
#certain probability of being suspected, and the remained are labelled as unknown like in real life
#node_status = list(map(lambda x: arbitrary_status_update(x, 0.05, 0.2), list(G.nodes)))

#SINCE THE SAMPLE USERS DATA ALREADY HAS REAL LABELS AS SPAM (1) OR NOT (0), THE AVAILABLE
#INFORMATION CAN BE SERVED AS NODE STATUS IN THIS CASE.
for n in G.nodes:
    user_info = user_sample_1[user_sample_1['id'] == n]
    if user_info['val3'].values[0] == 1:
        G.node[n]['status'] = 'SAR'
    else:
        G.node[n]['status'] = 'clear'

#The ratio of labelled spammers in the reduced network
len([n for n in G.nodes if G.node[n]['status']=='SAR'])/G.number_of_nodes()

#Creating a list of candidate subgraphs as the first step in constructing the data with graph features
candidate_subgraphs = []

#Extracting the strongly connected component subgraphs in a list from the entire graph
#Strongly connected components imply circulation of funds within a group
scc = list(nx.strongly_connected_component_subgraphs(G))

#Only strongly connected components with more than two nodes are potentially meaningful for this work.
#Strongly connected components with one nodes are only making the analyses computationally
#expensive without any valuable contribution, where strongly connected components with two nodes
#could be indicating reciprocal trading most of the time
meaningful_scc = [g for g in scc if len(g) > 2]

#wcc.sort(key = len, reverse = True)
meaningful_scc.sort(key = len, reverse = True)
#the number of meaningful strongly connected components
len(meaningful_scc)

#Add the meaningful strongly connected components to the candidate subgraphs list
candidate_subgraphs.extend(meaningful_scc)

#Calculating the ratio of SAR, clear and unknown of nodes in each candidate subgraph
for i in range(len(candidate_subgraphs)):
    candidate_subgraphs[i].graph['spam_ratio'] = len([x for x,y in candidate_subgraphs[i].nodes(data=True)
                if y['status']=='SAR'])/len(candidate_subgraphs[i])
    candidate_subgraphs[i].graph['clear_ratio'] = len([x for x,y in candidate_subgraphs[i].nodes(data=True)
                if y['status']=='clear'])/len(candidate_subgraphs[i])
    #candidate_subgraphs[i].graph['unknown_ratio'] = len([x for x,y in candidate_subgraphs[i].nodes(data=True)
    #             if y['status']=='unknown'])/len(candidate_subgraphs[i])

#Labelling the candidate subgraphs with a risk averse perspective
tolerance_lower_bound = 0 #A lower bound is set for tolerating the spammers in candidate subgraphs
for i in range(len(candidate_subgraphs)):
    if candidate_subgraphs[i].graph['spam_ratio'] > tolerance_lower_bound: #Let's say presence of
    #even one single spammer can be suspicious
        candidate_subgraphs[i].graph['suspect?'] = 1 #The subgraph is labelled suspicious
    else: #If all the members of the strongly connected subgraph are innocent
        candidate_subgraphs[i].graph['suspect?'] = 0 #If there are not any SAR nodes present
```

```
        #in the subgraph, it is assumed clear.

#Calculating the number of flagged suspicious nodes and cleared ones among the labelled data
SAR_nodes = [x for x,y in G.nodes(data=True) if y['status']=='SAR']
clear_nodes = [x for x,y in G.nodes(data=True) if y['status']=='clear']
print(len(SAR_nodes) , len(clear_nodes))
#For the sake of computational efficiency, a subset of size 5000 from each label is
#designated for further calculations. Instead of constructing the subsets randomly,
#"High degree seed selection" rule is applied since highly connected nodes naturally bring
#more transactions into the analysis.
print(len(SAR_nodes) , len(clear_nodes))

#High degree seed selection as labelled seeds for upcoming ego network construction
#NOTE: For the sake of computational efficiency, 5000 centroids of each class are sampled.
clear_nodes.sort(key=G.degree(), reverse=True)
SAR_nodes.sort(key=G.degree(), reverse=True)

clear_sample = clear_nodes[:5000]
SAR_sample = SAR_nodes[:5000]

#Definition of a function for constructing the ego networks of the remaining labelled nodes
#Since this is the case where any risk is avoided, it is assumed that presence of any criminal
#node in a subgraph casts doubts on itself. Therefore, ego networks are classified according to
#their centroids. For this sample 1, 2-step neighborhood is chosen by setting the radius
#parameter to 2, since it is a very sparse sample.

def build_egonet(n, graph):
    ego = nx.DiGraph()
    ego = nx.ego_graph(graph, n, center=True, undirected=True, radius=2)
    ego.graph['central_customer'] = n
    if G.node[n]['status'] == 'SAR':
        ego.graph['suspect?'] = 1 #ego network of a suspicious node is also labelled suspicious
    elif G.node[n]['status'] == 'clear':
        ego.graph['suspect?'] = 0 #ego network of a cleared node is assumed clear too
    else:
        ego.graph['suspect?'] = -1 #ego network of an unlabelled node is unlabelled
    return ego

#Listing only the labelled nodes from the entire network to be the central nodes of ego networks
#This can be applied to all nodes of the network but building egonets are not computationally efficient
central_seeds = SAR_sample+ clear_sample

#Building the ego networks of the entire graph
egonets = list(map(lambda x: build_egonet(x, G), central_seeds))

#Add the ego networks to the candidate subgraphs list
candidate_subgraphs.extend(egonets)

candidate_subgraphs.sort(key=len)
len(candidate_subgraphs)

#Mapping the functions to calculate network features for each candidate subgraph
density = list(map(nx.density,candidate_subgraphs))
no_of_nodes = list(map(len, candidate_subgraphs))
avg_in_degree = list(map(lambda graph: sum(d for n, d in graph.in_degree())
           / float(graph.number_of_nodes()), candidate_subgraphs))
#avg_out_degree = list(map(lambda graph: sum(d for n, d in graph.out_degree())
          #/ float(graph.number_of_nodes()), candidate_subgraphs))
no_of_in_tx = list(map(lambda graph: graph.size(), candidate_subgraphs))
in_tx_volume = list(map(lambda graph: graph.size('weight'), candidate_subgraphs))
avg_clustering_coefficient = list(map(lambda graph:
         nx.average_clustering(graph, weight = False),
         candidate_subgraphs))
avg_weighted_clustering_coefficient = list(map(lambda graph:
```

```
          nx.average_clustering(graph, weight = True),
          candidate_subgraphs))
avg_shortest_path_length = list(map(lambda graph:
         nx.average_shortest_path_length(graph, weight = False),
         candidate_subgraphs))
avg_weighted_shortest_path_length = list(map(lambda graph:
        nx.average_shortest_path_length(graph, weight = True),
        candidate_subgraphs))

diameter = list(map(lambda graph: nx.diameter(nx.to_undirected(graph)), candidate_subgraphs))
radius = list(map(lambda graph: nx.radius(nx.to_undirected(graph)), candidate_subgraphs))
transitivity = list(map(nx.transitivity, candidate_subgraphs))
no_of_triangles = list(map(lambda graph:
     sum(nx.triangles(nx.to_undirected(graph)).values())/3, candidate_subgraphs))
no_of_outgoing_tx = list(map(lambda graph:
      len(nx.node_boundary(G,graph.nodes())), candidate_subgraphs))
no_of_incoming_tx = list(map(lambda graph:
      len(nx.node_boundary(G,nx.restricted_view(G,
            graph.nodes(),[]),graph.nodes())),
            candidate_subgraphs))
volume_of_outgoing_tx = list(map(lambda graph:
   sum(G.edges[e]['weight'] for e in nx.edge_boundary(G,graph.nodes())),
                candidate_subgraphs))
volume_of_incoming_tx = list(map(lambda graph:
   sum(G.edges[e]['weight'] for e in nx.edge_boundary(G,nx.restricted_view(G,
             graph.nodes(),[]),graph.nodes())),
             candidate_subgraphs))

#Constructing the suspect list as target column for supervised learning
is_subgraph_suspicious = list(g.graph['suspect?'] for g in candidate_subgraphs)

graph_data = pd.DataFrame(list(zip(density,
                                   no_of_nodes,
                                   avg_in_degree,
                                   #avg_out_degree,
                                   no_of_in_tx,
                                   in_tx_volume,
                                   avg_clustering_coefficient,
                                   avg_weighted_clustering_coefficient,
                                   avg_shortest_path_length,
                                   avg_weighted_shortest_path_length,
                                   diameter,
                                   radius,
                                   transitivity,
                                   no_of_triangles,
                                   no_of_outgoing_tx,
                                   no_of_incoming_tx,
                                   volume_of_outgoing_tx,
                                   volume_of_incoming_tx,
                                   is_subgraph_suspicious)),
                          columns=['density',
                                   'no_of_nodes',
                                   'avg_in_degree',
                                   'no_of_in_tx',
                                   'in_tx_volume',
                                   'avg_clustering_coefficient',
                                   'avg_weighted_clustering_coefficient',
                                   'avg_shortest_path_length',
                                   'avg_weighted_shortest_path_length',
                                   'diameter',
                                   'radius',
                                   'transitivity',
                                   'no_of_triangles',
                                   'no_of_outgoing_tx',
```

```
                                        'no_of_incoming_tx',
                                        'volume_of_outgoing_tx',
                                        'volume_of_incoming_tx',
                                        'is_subgraph_suspicious'])

#Stopping the clock for reporting the so far execution time for extracting candidate subgraphs
end_time = datetime.now()
print('Duration: {}'.format(end_time - start_time))

#Required libraries and methods
from datetime import datetime
import networkx as nx
from random import randint
from random import random
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import itertools
from collections import deque
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, make_scorer, accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, precision_recall_curve
from sklearn.metrics import auc, roc_curve, roc_auc_score, average_precision_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from lightgbm import LGBMClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, KFold, cross_validate
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
import logging
from pandas import Series
logger = logging.getLogger()
logger.setLevel(logging.INFO)
logging.debug('test')

def tp_ratio(y_true, y_pred, pct=0.2):
    if y_pred.ndim == 2:
        y_pred = y_pred[:, 1]
    n = int(round(len(y_true) * pct))
    idx = np.argsort(y_pred)[-n:]
    return y_true[idx].sum() / float(n)

def performance_report(y_test, preds):
    result = 'Performance report for the model: \n'
    result += 'Overall mean of the target: %.3f%%\n' % (y_test.mean() * 100)
    preds_bin = preds.values > 0.5
    result += 'Accuracy: %.3f\n' % accuracy_score(y_test.values, preds_bin)
    try:
        result += 'ROC_AUC: %.3f\n' % roc_auc_score(y_test.values, preds.values)
        result += 'Log_loss: %.3f\n' % log_loss(y_test.values, preds_bin)
        result += 'F1_score: %.3f\n' % f1_score(y_test.values, preds_bin)
        result += 'MCC: %.3f\n' % matthews_corrcoef(y_test.values, preds_bin)
    except:
        result += 'Could not calculate the binary performance metrics.'
    result += '\nClassification report for the model:\n'
    result += '\n' + classification_report(y_test.values, preds_bin)
    result += '\nTrue positive ratio at top 20%%: %0.2f%%\n\n' % (tp_ratio(y_test.values,
                    preds.values) * 100)
    return result
```

```
graph_data = pd.read_csv(
  '/Users/gizemtas/Desktop/THESIS/Experimentations-random/exp_sample_1_tolerance0_rand.csv',
  index_col=[0])
print(len(graph_data[graph_data.is_subgraph_suspicious==1]),
  len(graph_data[graph_data.is_subgraph_suspicious==0]))

X_train, X_test, y_train, y_test = train_test_split(graph_data[graph_data.columns[0:16]],
                                                    graph_data['is_subgraph_suspicious'],
                                                    test_size = 0.2,
                                                    random_state = 10)

X_test.to_csv (r'/Users/gizemtas/Desktop/THESIS/X_test_random_tol0.csv',
    index = True, header=True)
y_test.to_csv (r'/Users/gizemtas/Desktop/THESIS/y_test_random_tol0.csv',
    index = True, header=True)

#RANDOM FOREST
estimator = RandomForestClassifier(n_jobs=2, random_state=42)
param_grid = {#'learning_rate': [0.01, 0.05, 0.1],
              'n_estimators': [400, 500, 750, 1000],
              #'scale_pos_weight':[class_ratio,1],
              #'min_child_weight':[1, 5, 10],
              #'gamma':[0.5, 1, 1.5, 2, 5],
              #'subsample':[i/10.0 for i in range(6,11)],
              #'colsample_bytree':[i/10.0 for i in range(6,11)],
              'max_depth': [2,3,4,5,6]}

model = GridSearchCV(estimator, param_grid, cv=5)
model.fit(X_train, y_train)
print('RF_Best_parameters_found_by_grid_search_are:', model.best_params_)
print(confusion_matrix(y_test, model.predict(X_test)))
print(classification_report(y_test, model.predict(X_test)))
print('RF_Area_under_ROC_curve:', roc_auc_score(y_test, model.predict_proba(X_test)[:,1]))
print('RF_Area_under_precision-recall_curve:',
  average_precision_score(y_test, model.predict_proba(X_test)[:,1], pos_label=1))

#Best Random Forest
best_rf = RandomForestClassifier(n_jobs=2, random_state=42, max_depth=6, n_estimators=750)
best_rf.fit(X_train, y_train)
print(confusion_matrix(y_test, best_rf.predict(X_test)))
print(classification_report(y_test, best_rf.predict(X_test)))
print('RF_Area_under_ROC_curve:', roc_auc_score(y_test, best_rf.predict_proba(X_test)[:,1]))

fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, best_rf.predict_proba(X_test)[:,1])
precision_rf, recall_rf, thresholds_rf = precision_recall_curve(y_test,
               best_rf.predict_proba(X_test)[:,1])
print('RF_Area_under_precision-recall_curve:',auc(recall, precision))
logger.info(performance_report(y_test, Series(best_rf.predict_proba(X_test)[:,1])))

pyplot.plot(recall, precision, marker='.', label='Random_Forest', color = 'orange')
# axis labels
pyplot.title('Precision-Recall_Curve')
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()

# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_test, model.predict_proba(X_test)[:,1])
print('RF_Area_under_precision-recall_curve:',auc(recall, precision))
print ("Features_sorted_by_their_score:")
```

```
print (sorted(zip(map(lambda x: round(x, 4),
      model.feature_importances_),
      graph_data.columns), reverse=True))

neg_class = len(graph_data[graph_data.is_subgraph_suspicious==0])
pos_class = len(graph_data[graph_data.is_subgraph_suspicious==1])
class_ratio = neg_class/pos_class
print(pos_class/(neg_class+pos_class), class_ratio)

neg_class_train = len(y_train[y_train==0])
pos_class_train = len(y_train[y_train==1])
class_ratio_train = neg_class_train/pos_class_train
print(pos_class_train/(neg_class_train+pos_class_train), class_ratio_train)

#XGBOOST
estimator = XGBClassifier(objective="binary:logistic", random_state=42)
param_grid = {'learning_rate': [0.01, 0.05, 0.1],
              'n_estimators': [400, 500, 750, 1000],
              'scale_pos_weight':[class_ratio,1],
              #'min_child_weight':[1, 5, 10],
              #'gamma':[0.5, 1, 1.5, 2, 5],
              #'subsample':[i/10.0 for i in range(6,11)],
              #'colsample_bytree':[i/10.0 for i in range(6,11)],
              'max_depth': [2,3,4,5,6]}

gbm = GridSearchCV(estimator, param_grid, cv=5)
gbm.fit(X_train, y_train)
print('XGB_Best_parameters_found_by_grid_search_are:', gbm.best_params_)
print(confusion_matrix(y_test, gbm.predict(X_test)))
print(classification_report(y_test, gbm.predict(X_test)))
print('XGB_Area_under_ROC_curve:', roc_auc_score(y_test, gbm.predict_proba(X_test)[:,1]))
print('XGB_Area_under_precision-recall_curve:',
  average_precision_score(y_test, gbm.predict_proba(X_test)[:,1], pos_label=1))

#Best XGBoost
best_xg = XGBClassifier(objective="binary:logistic",
      random_state=42, learning_rate = 0.01, max_depth = 6,
      n_estimators = 400, scale_pos_weight = 1)
best_xg.fit(X_train, y_train)
print(confusion_matrix(y_test, best_xg.predict(X_test)))
print(classification_report(y_test, best_xg.predict(X_test)))
print('RF_Area_under_ROC_curve:', roc_auc_score(y_test, best_xg.predict_proba(X_test)[:,1]))

fpr_xg, tpr_xg, thresholds_xg = roc_curve(y_test, best_xg.predict_proba(X_test)[:,1])
precision_xg, recall_xg, thresholds_xg = precision_recall_curve(y_test,
              best_xg.predict_proba(X_test)[:,1])
print('RF_Area_under_precision-recall_curve:',auc(recall, precision))
logger.info(performance_report(y_test, Series(best_xg.predict_proba(X_test)[:,1])))

#LGBOOST
estimator = LGBMClassifier(importance_type='gain')
param_grid = {'learning_rate': [0.01, 0.05, 0.1],
              'n_estimators': [400, 500, 750, 1000],
              'scale_pos_weight':[class_ratio,1],
              #'min_child_weight':[1, 5, 10],
              #'gamma':[0.5, 1, 1.5, 2, 5],
              #'subsample':[i/10.0 for i in range(6,11)],
              #'colsample_bytree':[i/10.0 for i in range(6,11)],
              'num_leaves':[50,100,200],
              'max_depth': [3,4,5]}

gbm = GridSearchCV(estimator, param_grid, cv=5)
gbm.fit(X_train, y_train)
print('LGB_Best_parameters_found_by_grid_search_are:', gbm.best_params_)
```

```
print(confusion_matrix(y_test, gbm.predict(X_test)))
print(classification_report(y_test, gbm.predict(X_test)))
print('LGB Area under ROC curve:', roc_auc_score(y_test, gbm.predict_proba(X_test)[:,1]))
print('LGB Area under precision-recall curve:',
  average_precision_score(y_test, gbm.predict_proba(X_test)[:,1], pos_label=1))

#BEST LGBOOST
best_lg = LGBMClassifier(importance_type='gain', learning_rate = 0.01,
      max_depth = 5, n_estimators = 1000,
      scale_pos_weight = 1, num_leaves=50)
best_lg.fit(X_train, y_train)
print(confusion_matrix(y_test, best_lg.predict(X_test)))
print(classification_report(y_test, best_lg.predict(X_test)))
print('RF Area under ROC curve:', roc_auc_score(y_test, best_lg.predict_proba(X_test)[:,1]))

fpr_lg, tpr_lg, thresholds_lg = roc_curve(y_test, best_lg.predict_proba(X_test)[:,1])
precision_lg, recall_lg, thresholds_lg = precision_recall_curve(y_test,
              best_lg.predict_proba(X_test)[:,1])
print('RF Area under precision-recall curve:',auc(recall, precision))
logger.info(performance_report(y_test, Series(best_lg.predict_proba(X_test)[:,1])))

# Feature Importance
model =  best_lgboost
cols = list(X_train.columns)
imps = model.feature_importances_
feats = pd.Series(imps, index=cols).sort_values(ascending=False)
feats = pd.DataFrame(feats)
feats.reset_index(level=0, inplace=True)
feats.rename(columns={'index':'feature', 0:'importance'}, inplace=True)
feats['importance'] = feats['importance'].astype(float)
feats = feats[feats['importance']>0]
logger.info(performance_report(y_test, Series(best_lgboost.predict_proba(X_test)[:,1])))

from matplotlib import pyplot
fig, ((ax1,ax2,ax3),(ax4,ax5,ax6)) = pyplot.subplots(2,3)
fig.set_figheight(9)
fig.set_figwidth(15)
fig.tight_layout(pad=3)
ax1.plot(fpr_rf, tpr_rf, marker='.', label='Random Forest', color='green')
ax1.set_title('ROC Curve')
ax1.set(xlabel='False Positive Rate', ylabel='True Positive Rate')
ax1.legend()
ax4.plot(recall_rf, precision_rf, marker='.', label='Random Forest', color = 'orange')
ax4.set_title('Precision-Recall Curve')
ax4.set(xlabel='Recall', ylabel='Precision')
ax4.legend()
ax2.plot(fpr_xg, tpr_xg, marker='.', label='XGBoost', color='green')
ax2.set_title('ROC Curve')
ax2.set(xlabel='False Positive Rate', ylabel='True Positive Rate')
ax2.legend()
ax5.plot(recall_xg, precision_xg, marker='.', label='XGBoost', color = 'orange')
ax5.set_title('Precision-Recall Curve')
ax5.set(xlabel='Recall', ylabel='Precision')
ax5.legend()
ax3.plot(fpr_lg, tpr_lg, marker='.', label='LightGBM', color='green')
ax3.set_title('ROC Curve')
ax3.set(xlabel='False Positive Rate', ylabel='True Positive Rate')
ax3.legend()
ax6.plot(recall_lg, precision_lg, marker='.', label='LightGBM', color = 'orange')
ax6.set_title('Precision-Recall Curve')
ax6.set(xlabel='Recall', ylabel='Precision')
ax6.legend()
```