

**MESTRADO**  
ECONOMIA E ADMINISTRAÇÃO DE EMPRESAS

**SPOTTING FRAUD: DETECTING  
PATTERNS AND RED FLAGS IN  
FINANCIAL NETWORKS**

*Joana Isabel Cortez Trindade*

**M**

2022



---

SPOTTING FRAUD: DETECTING PATTERNS AND RED FLAGS IN  
FINANCIAL NETWORKS

**Joana Isabel Cortez Trindade**

---

Dissertation  
Master in Economics and Business Administration

---

Supervised by  
**Professor Pedro Campos, PhD**  
**Professor Pedro Ribeiro, PhD**

---

2022



## **Acknowledgements**

I would like to express my deep gratitude to Professor Pedro Campos and Professor Pedro Ribeiro, my supervisor and cosupervisor for all the support, guidance and encouragement.

I would like to acknowledge everyone who played a role in this academic accomplishment.

To my parents and brother, for the unconditional support, for being an example of commitment and hard work. Thank you for always giving me strength to pursue every dream.

To João for all the emotional support and everything. Without you everything would be harder, you have been amazing and you are an example to me.

To my friends for all the support and understanding my absence during these times.

## **Abstract**

Fraud is an old but always evolving practice. In our days' the digitalization of money, banks and financial transactions has provided fraudsters with infinite opportunities to commit crime from behind a screen, anywhere in the world. The impact of fraud is wide, with direct consequences for business and the economy.

The main goal of this study is to create a model able to classify fraudulent transaction, as well as detecting patterns and red flags associated with fraud. analyze the impact of network measures in fraud detection.

For that purpose, a synthetic dataset including financial transactions data, Paysim was used. Firstly, the dataset is studied, identifying all the sub networks of directly connected accounts. G-tries is the algorithm used used to scan the networks to find 3 and 4 nodes subgraphs in an attempt to look for motifs associated with fraud. Then a classification model using Random Forest including network measures is created. A Logistic Regression model was also created in order to identify the best attribute for fraud detection in Paysim dataset. Finally, the performance of the model and results are presented and discussed.

In the end of this work, the main patterns and red flags associated with fraud are presented, as well as the model providing information about fraudulent schemes and how the people committing fraud can be spotted within a given financial network.

# Index

1. Introduction .....	1
2. Literature Review .....	3
2.1. Network Motifs.....	10
2.2. Classification Algorithms.....	12
3. Dataset, Software, Questions and Methodology .....	17
3.1. Software.....	17
3.2. Dataset's Nature.....	17
3.3. Descriptive Analysis of Paysim.....	20
3.4. Dataset and Fraud.....	30
3.5. Questions .....	36
3.6. Gtries .....	37
3.7. Three Nodes Subgraphs .....	38
3.8. Four Nodes Subgraphs .....	41
4. Classification Problem.....	45
4.1. Random Forest.....	45
4.2. Logistic Regression .....	48
5. Model Results and Performance .....	49
5.1. Random Forest.....	49
5.2. Logistic Regression .....	51
6. Conclusions and Future Work .....	58
7. Bibliography .....	60
8. Attachments .....	64
8.1. Paysim Data Analysis .....	64
8.2. Network and Dataset creation .....	91
8.3. Gtries and Subgraphs Detection.....	102

8.4. RapidMiner's Process and Model.....	105
--	-----

## Index of Figures

Figure 1: Occupational Fraud Classification (ACFE, 2012).....	4
Figure 2: Real Networks vs Randomized Networks, (Milo, et al., 2002) .....	10
Figure 3: Visualization of the full Paysim dataset network with a random layout. Source: the author. ....	19
Figure 4: Visualization of the full Paysim dataset network with an circular layout. Source: the author. ....	19
Figure 5: Frequency of transactions per step (hours of the 30 day of the simulation). Source: the author.....	21
Figure 6: Frequency of networks per network size (nodes). Source: the author.....	31
Figure 7: Visualisation of Paysim’s Sub networks. Source: the author. ....	32
Figure 8: G-tries’ example. (Ribeiro & Silva, 2014) .....	37
Figure 9: gtries input file. Source: the author. ....	38
Figure 10: g-tries output file. Source: the author. ....	38
Figure 11: g-tries occurrences file. Source: the author.....	39
Figure 12: gtries input file. Source: the author. ....	41
Figure 13: g-tries output file. Source: the author. ....	41
Figure 14: g-tries occurrences file. Source: the author.....	42
Figure 15: centralitySender and centralityReceiver example on decision tree. Source: the author. ....	54
Figure 16: Amount attribute example on decision tree. Source: the author.....	55
Figure 17: newBalanceDest, newBalanceOrig, oldBalanceOrig and oldBalanceDest example on decision tree. Source: the author. ....	57
Figure 18: RapidMiner’s Process before Under Sampling. Source: the author. ....	105
Figure 19: RapidMiner's Dataset for model creation. Source: the author.....	106
Figure 20: Final RapidMiner process including Random Under Sampling. Source: the author. ....	107
Figure 21: Final RapidMiner process including Logistic Regression. Source: the author..	107



## Index of Tables

Table 1: Data Mining and Network Analysis techniques for fraud detection summary.....	9
Table 2: Classification of most used strategies for network motifs discovery (Ribeiro, Silva, & Kaiser, Strategies for Network Motifs Discovery, 2009). .....	11
Table 3: Confusion matrix.....	16
Table 4: Overview of step variable values.....	21
Table 5: Relative frequency of transaction types .....	22
Table 6: Overview of amount variable values .....	23
Table 7: Overview of amount values per transaction type.....	23
Table 8: Number of times each account on «nameOrig» appears on the transaction dataset .....	24
Table 9: Overview of nameOrig accounts .....	24
Table 10: Overview of nameDest accounts.....	25
Table 11: Overview of oldbalanceOrg .....	25
Table 12: Overview of newbalanceOrg.....	25
Table 13: Overview of oldbalanceDest .....	26
Table 14: Overview of newbalanceDest .....	26
Table 15: Overview of Merchants accounts.....	27
Table 16: Overview of isFraud .....	28
Table 17: Overview of isFlaggedFraud .....	28
Table 18: Overview of is FlaggedFraud ==1 .....	29
Table 19: Overview of fraudulent transactions.....	30
Table 20: Fraudulent transactions per transaction type.....	30
Table 21: Description of non-fraudulent (isFraud==0) sub networks transactions .....	33
Table 22: Description of fraudulent (isFraud== 1) sub networks transactions .....	34
Table 23: Three nodes subgraphs found running g-tries on Paysim sub networks.....	39
Table 24: Frequency of three nodes subgraphs in fraudulent networks .....	40
Table 25: Four nodes subgraphs found running g-tries on Paysim subnetworks.....	43
Table 26: Frequency of three nodes subgraphs in fraudulent networks .....	44
Table 27: Models results per dataset.....	47
Table 28: RapidMiner Random Forest calculation parameters.....	49
Table 29: Random Forest model performance after Under Sampling.....	50
Table 30: Logistic Regression results.....	51

Table 31: Logistic Regression Model performance after Under Sampling .....52

# 1. Introduction

Fraud is a widespread phenomenon. There is often news about fraud using computer networks and, in some cases, only after some time the situation is detected. The impact of fraud is wide, with direct consequences for business and the economy. Methodologies for detecting fraud are therefore essential. Statistical and data mining methods provide effective tools to support business activities. Fraud detection is one of them, and there are applications to detect cases such as credit card fraud or intrusion into computer systems.

The Association of Certified Fraud Examiners (ACFE), concluded in the Report to The Nations published in 2020, that organizations worldwide lose about 5% of revenue every year due to fraud. This figure represents about 4.5 million of millions of USD lost annually due to fraud (ACFE, 2020). Several studies on this phenomenon report shocking numbers: 46% of companies worldwide have reported experiencing fraud, corruption or other economic crime in the past year. PriceWaterhouse & Coopers calls out for the increasing number of cybercrime, customer fraud and asset misappropriation (PwC, 2022).

The motivation for this work came from the possible synergy between economic areas that focus on fraud detection, analysis and prevention and the data analysis methods and the recent advances in areas such as Data Mining and Machine Learning. Some of the attributes of this type of methods such as the capability of analysing great quantities of transactions and the detection of fraudulent networks can be used as a major tool to detect and prevent fraud, as well as, diminishing the negative impact that fraud has in organizations worldwide and overall macro economy (Gee, 2015).

As stated by Milo et al. (2002) network motifs are universally found in all fields of study from biochemistry, neurobiology, to ecology, and engineering. Motifs may therefore define universal classes of networks. The author insists that it is of value to detect and understand network motifs in order to gain insight into their dynamical behaviour and to define networks (Milo, et al., 2002).

The aim of this work is to investigate the use of network motifs and patterns for fraud detection. For this, a synthetic financial transaction dataset will be studied. The goal is to identify and characterize the financial transaction network. Then, to identify what network motifs and patterns are related with fraudulent transactions, the frequency and characteristics of these fraudulent network motifs. These network motifs are carrying information about

fraudulent schemes and how the people committing fraud can be spotted within a given financial network.

This dissertation is divided as follows: in chapter 2 the literature review is presented, we reviewed previous work related to fraud identification, networks, subgraphs and patterns identification and finally data mining classification algorithms

In chapter 3 an extensive description of the dataset is presented and the methodology is explained. This chapter includes some preliminary results, namely the subgraphs existent on the Paysim sub networks.

The main analysis was performed in chapter 4, where the final dataset is built and used to build a classification model, resulting on the identification of the best fraudulent transactions indicators which are presented on chapter 5.

Finally, on chapter 6 the conclusions of this study are presented as well as suggestions for future work.

## 2. Literature Review

We start with a short definition of fraud as outlined in Black's Law Dictionary:

«An act of intentional deception or dishonesty perpetrated by one or more individuals, generally for financial gain. » (Garner, 2019)

Simmons, on the other hand, states that we are faced with fraud when a series of elements occur (Simmons, 1995):

- An individual or organization intentionally makes false claims about an important fact or event.
- The false statement is believed by the victim (the person or organization to whom the statement was presented).
- The victim trusts and acts upon the false statement.

This implies that a simple mistake or misunderstanding is not fraudulent when there was no intent to harm the victim.

Occupational fraud implies that an employee violates the trust associated with his or her duties and conceals the fraud. The employee takes action to conceal the fraud and hopes that it will not be discovered in a timely manner or until it is never uncovered.

Fraud is not the norm. Other anomalies such as accounting record anomalies are the result of inadequate process or procedures and other weaknesses in internal control processes. Sometimes they are even repeated consistently at a certain frequency or at certain times of the year such as the end of the month or the end of the year. Understanding the practices and procedures inherent in a given organization helps to explain and find most of these anomalies (Gee, 2015).

The ACFE in the 2012 report to the nations, (ACFE, 2012) divides occupational fraud into three major categories, as depicted below in Figure 1.

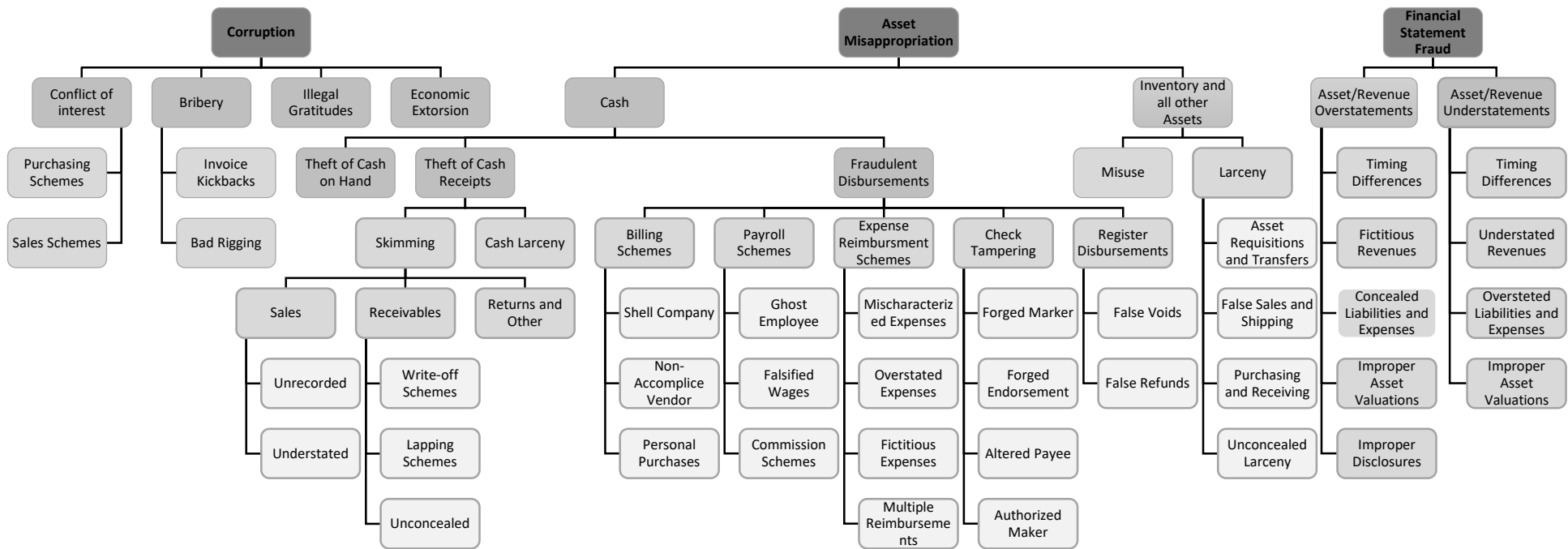


Figure 1: Occupational Fraud Classification (ACFE, 2012)

Pimenta (2009) affirms that when we talk about fraud we are dealing with a vast set of situations, mostly intentional, in which some citizens or institutions deceive others, causing direct or indirect economic and social damage. We are, above all, considering processes that are part of the economic fabric of this increasingly globalized society (Pimenta, 2009).

Currently, most fraud cases are brought to light because of someone's tip, meaning most fraud is discovered only because it is reported. On the other hand, a lack of internal controls contributed for a third of the fraud cases (ACFE, 2020).

However, with technological development, the increasing complexity of problems, and the increasing volume of data available, has become a greater need and ability to develop sophisticated and autonomous computational tools. Nowadays, this kind of tools can have a wide range of practical applications.

The object of financial fraud detection in this work has two particular characteristics: a high number of transactions to be handled and a short time to decide what to do with the information. Even with Covid-19 pandemic, credit/debit card transactions instead of money have been widely used to purchase items or to pay for almost anything (Kosse & Szemere, 2021). Consequently, millions of people perform millions of transactions every minute of the day. All these transactions must be analysed in order to determine if they are fraudulent or not. The problem is not only the huge amount of data to be analysed but, also, the short time in which this must be done (Almeida, 2009).

The main goal here is to prevent a fraudulent transaction before it is labelled as illegal, by red flagging it.

Financial fraud can be hard to detect when transaction characteristics are taken individually. For instance, a large transaction is not *prima facie* suspicious, unless it is performed at usual times (for example., at night) or in an unusual store (a store never visited before by the card owner, located in a different city, etc.) (Zanin, Romance, Moral, & Criado, 2018). Data mining and network analysis are successfully able to detect patterns and have this multi characteristics view over the transaction data.

When applying network analysis for fraud detection, one must keep in mind that in real world, false-positive alarms might lead to the accusation of innocent people. On the

other hand, false-negative alarms mean that a fraudster might succeed. However, the fine-tuning of both is usually coupled. More sensitive algorithms lead to the detection of more frauds, but also more false-positive alarms will be generated. To this end, the calibration of their sensitiveness is essential.

While network analysis algorithms can identify undetected patterns in data, they usually lack the capacity of synthesizing metrics describing the global structure created by the interactions between the different features. The use of Visual Analytics can be essential in order to easily interpret the network and adapt the fraud detection algorithm (Leite, Gschwandtner, Gstrein, & Kuntner, 2018).

Sherly and Nedunchezian (2010) built a model capable of detecting fraud by analysing consumer's profile and historic behaviour. This behaviour includes data regarding transaction amounts, category of the items, purchase address, among others. Then, the system compares the incoming transaction to the consumer's history, and, through BOAT (Bootstrapped Optimistic Algorithm for Tree Construction) algorithm, it is detected (or not) any anomaly. This anomaly is marked as possible fraud.

Malekian and Hashemi (2013) used a learning algorithm and studied the dynamic and nonstationary behaviour. It was introduced in the system a temporary profile and the algorithm is capable of, regardless the historical information, retain new concepts from the incoming data.

Chandola *et al.* (2009) analysed two objects, the customer and the operation. The first one is characterized by comparing the incoming data with the credit card history and if both data do not match, the transaction is flagged as an anomaly. On the other hand, operation approach is based on geographic location: if the location of the incoming transaction does not match with the profile, it is flagged as an anomaly.

Francis *et al.* (2011) used Support Vector Machines (SVM) to look for fraud on medical insurance. Using SVM, it was possible to accelerate the time of detection fraud to the real time.

Tsai *et al.* (2014) stated that the information is stored in heterogeneous databases. Using a methodology called CommonKADS, that develops a comprehension of the system. Using this methodology, the authors claim, fraud detection will be easier and quicker, and the labour costs will be reduced.

Jeh and Widom (2002) presented an approach to detect anomalies that is applicable to any object-to-object relationships domain. Basing on the object's relationship with other



objects, the authors focused in measuring the similarity of the structural context where the object occurs. Two objects are similar if they are related to similar objects.

Noble and Cook (2003) proposed graph-based anomaly detection techniques using two methods. The first one consists in detecting anomalous substructure analysing unusual substructures in a graph and, in the second one, the authors partitioned the graphs into several sets of subgraphs and tested each one against the others, looking for unusual patterns.

Xu *et al.* (2007) worked on Structural Clustering Algorithm for Networks (SCAN) in order to detect clusters and outliers in networks, using structural similarity measures. To detect clusters vertices, they use common neighbours, and two vertices are assigned to a cluster depending on their neighbours.

Akoglu *et al.* (2010) showed how the discovery of some rules regarding density, weights, ranks and eigenvalues can help the anomaly detection. They also focused on questions such as “what features should we use to characterize a neighbourhood?” and “what does a “normal” neighbourhood look like?”. OddBall, the algorithm they created, detects anomalies in weighted, unlabelled graphs.

The authors Yan and Han (2002) developed the algorithm gSpan, that is capable of discover frequent substructures. In other words, gSpan allows to detect frequent patterns of connectivity within a subgraph building a new lexicographic order and, in each graph, it is associated to a unique code.

Yue *et al.* made a review of Data Mining based financial fraud detection methods and claimed that regression was the most popular method, using financial ratios such as the inventories to sales ratio, the ratio of the total debt to total assets, the working capital to total assets ratio, financial distress, amongst others (Yue, Wu, Wang, LI, & Chu, 2007).

Fontes, *et al.* are the only authors found that studied motifs and anti-motifs in the context of banking fraud. Recently they published a work where they used 3 nodes graph representations to build both entity graphs and transaction graphs. As future work, the authors propose one can investigate whether different banking datasets have similar motifs (and anti-motifs) and if those patterns are different in merchant datasets. It is also proposed to extend the analysis to larger motif sizes, different temporal windows, and the inclusion of transaction amounts or fraud labels as graph properties (Fontes, et al., 2021).

Others used artificial neural network, not only generalized adaptive neural network architectures and the adaptive logic network but also fuzzy rule was integrated with a neural

network. (Lin, Hwang, & Becker, 2003) And some studies used a combination of neural network, decision tree and Bayesian (Kirkos, Spathis, & Manolopoulos, 2007).

On Table 1, a summary of the different application of data mining and network analysis for fraud detection is presented:

Table 1: Data Mining and Network Analysis techniques for fraud detection summary

<b>AUTHOR, YEAR</b>	<b>TOPIC/TITLE</b>	<b>OBJECTIVE</b>	<b>METHODOLOGY</b>
Pouramirarsalani et al. (2017)	Fraud detection in E-banking by using the hybrid feature selection and evolutionary algorithms	To provide a new method to fraud detection in e-banking.	Used a hybrid feature selection and generic algorithm.
Jeh and Widom (2002)	SimRank: A Measure of Structural- Context Similarity	To measure the similarity of structural context of an object.	Computing a measure, two objects are similar if they are related to similar objects.
Noble and Cook (2003)	Graph-based anomaly detection	To detect unusual patterns in graph-based data.	The authors created a measure that calculates the regularity of a graph, using conditional entropy.
Xu et al. (2007)	SCAN: A Structural Clustering Algorithm for Networks	To develop a method capable of detecting clusters based on individual's common neighbours.	The individuals are assigned to the same cluster depending on how they share the neighbours.
Akoglu et al. (2010)	OddBall: Spotting anomalies in weighted graphs	To build an algorithm that is scalable and work for unsupervised data for anomaly detection.	Through some discovered rules in density, weights, ranks and egonets, the authors show how to detect anomaly in graphs.
Yan and Han (2002)	gSpan: graphbased substructure pattern mining	To find frequent substructures within a graph.	The authors adopted a DFS strategy to find patterns of connectivity in subgraphs.
Sherly and Nedunchezhian (2010)	BOAT adaptive credit card fraud detection system.	To build a system capable of detecting fraud through behaviour analysis.	Compare income behaviour with historical information and using BOAT algorithm, anomalies are detected and identified (or not) as fraud. Also, by combining classification and clustering techniques.
Malekian and Hashemi, (2013)	An Adaptive Profile based Fraud Detection Framework For Handling Concept Drift.	Study the dynamic and non-stationary behaviour.	It is introduced in the system a temporary profile and the algorithm is retain new concepts from the incoming data, regardless the historical information.
Chandola et al., (2009)	Anomaly detection: a survey.	Overview of the study of anomaly detection.	It was used two approaches technique: owner and operation approach.
Francis et al., (2011)	Using support vector machines to detect medical fraud and abuse	Analyse the efficiency of Quash, a bill processing system.	Using support vector machine, the authors accelerated the time of detection of fraud.
Tsai et al., (2014)	Using CommonKADS method to build prototype system in medical insurance fraud detection	Improve the inefficiency of the healthcare system.	Using a process which the main goal is to have a comprehension of the system (CommonKADS).
Fontes, et al. (2021)	Finding NeMo: Fishing in banking networks using network motifs	Find fraud in banking networks by studying motifs and anti-motifs using 3 noded subgraphs	Using two graph representations, entity graphs and transaction graphs, they proposed a novel randomization method that operates directly on tabular data.

## 2.1. Network Motifs

The “network motifs” are those patterns for which the probability  $P$  of appearing in a randomized network an equal or greater number of times than in the real network is lower than a cut-off value (Milo, et al., 2002).

In other words, network motifs are patterns that appear more frequently or less frequently than they are supposed to, i.e., than they would appear in randomly generated networks.

These network motifs, specific subgraphs, can be evidence of irregular activity in the network, fraudulent activity.

Figure 2 presents a visual explanation of the motif and the difference between real networks and randomized networks.

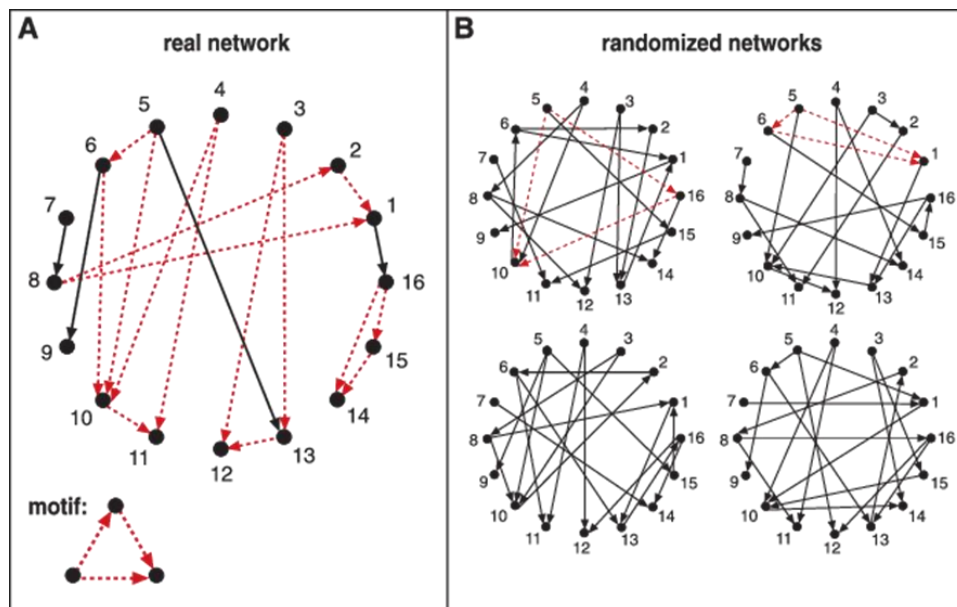


Figure 2: Real Networks vs Randomized Networks, (Milo, et al., 2002)

### How to detect network motifs?

Finding network motifs is a computationally hard task, as the size of the motifs increases, the time needed to calculate them grows exponentially.

Ribeiro, Silva, & Kaiser, studied the strategies for network motifs discovery and selected those that rely on the theoretical definition of motif and are the basis of almost every motif application found in the literature. They summarized the different strategies on a table where they present the main differences using a set of relevant distinguishable parameters (Ribeiro, Silva, & Kaiser, 2009).

Table 2: Classification of most used strategies for network motifs discovery (Ribeiro, Silva, & Kaiser, Strategies for Network Motifs Discovery, 2009).

<b>METHOD</b>	<b>BIAS</b>	<b>SYMMETRY BREAKING</b>	<b>NETWORK CENTRIC</b>	<b>PUBLIC TOOL</b>	<b>MOTIF SIZE</b>
<b>MFinder</b>	Yes	No	Yes	Yes	small
<b>FanMod</b>	No	Yes	Yes	Yes	medium
<b>Grochow</b>	Yes	Yes	No	No	large

When one needs to enumerate and consider all subgraphs, the authors advise FanMod strategy, unless we have small motif sizes and undirected subgraphs, in which case Grochow can be the better option. If trying to find if a relatively small set of specific subgraphs is a motif, then Grochow is also the better option.

According to the authors, an approach with potential success is to initially use FanMod to enumerate all subgraphs of the original network, and then use Grochow to count the occurrences on the random ensemble of random networks (Ribeiro, Silva, & Kaiser, 2009).

Regarding network motifs counting methods, Ribeiro et al., present a extensive review on existing practical methods to solve the subgraph counting problem, dividing them in three different categories:

- algorithms that efficiently perform exact counting, which is an intrinsically computationally hard task;
- algorithms that perform an approximation of subgraph frequencies, making the process faster, considering the accuracy of their estimation;
- algorithms that efficiently exploit parallel architectures despite the unbalanced nature of subgraph counting (Ribeiro, Paredes, Silva, Aparício, & Silva, 2021).

Ribeiro & Silva, develop a new method, g-tries, a novel data-structure created for storing and finding subgraphs. Inspired by prefix trees, g-tries store the subgraphs in a multiway tree that encapsulates information about common substructure, with related subgraphs being stored in common branches (Ribeiro & Silva, 2014).

In this work, Gtries Scanner was used. Nevertheless, all these methodologies are useful for network motifs identification and were considered for this work.

## 2.2. Classification Algorithms

In addition to investigate the connection (or not) between the patterns and graphs in financials transactions and fraud, we also tried to classify and predict transactions, so that we may ascertain how likely a certain transaction is to be fraudulent.

Databases and large quantities of data have hidden information that can be used for smart decision making. In our days, most decision support systems are built from data mining and machine learning algorithms. Witten, Frank and Hall defined data mining as the process of discovering patterns in data. The process must be automatic or (more usually) semi-automatic. The patterns discovered must be meaningful in that they lead to some advantages, usually an economic advantage (Witten, Frank, & Hall, 2011).

In general, for for Fraud Detection, data mining techniques can be classified into two categories according to the type of the machine learning techniques as:

### **Supervised Learning for Fraud Detection**

This method uses supervised learning in which all the available records are classified as «fraudulent» and «non-fraudulent». The machine is taught by example to identify records according to this classification.

The operator provides the machine learning algorithm with a known dataset that includes desired inputs and outputs, and the algorithm must find a method to determine how to arrive at those inputs and outputs. While the operator knows the correct answers to the problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and is corrected by the operator – and this process continues until the algorithm achieves a high level of accuracy/performance.

### **Unsupervised Learning for Fraud Detection**

In this method the machine learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analyzing available data. In an unsupervised learning process, the machine learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organize that data in some way to describe its

structure. This might mean grouping the data into clusters or arranging it in a way that looks more organized. Some examples are Clustering and Dimension reduction.

Within the supervised learning method, we can find Classification, Regression and prediction analysis. Classification and prediction are two forms of data analysis that can be used to extract models and predict future data trends. Such analysis allows us to have an understanding of large amounts of data, transforming data into information. Classification predicts categorical labels (discrete, unordered) whilst prediction models continuous-valued functions.

For example, a classification model can be used to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation (Han, 2006).

These provide a powerful tool for solving many economic substances. They are successfully used for inflation-deflation forecasting, for the currency exchange rates predictions, for the prediction of share prices and many other applications in this field.

In this situation it is clear that we are standing in front of a classification problem, the aim is to classify the transaction as fraudulent or non-fraudulent. The method chosen to reach this classification the Random Forest method.

### **Random Forests**

Tree models where the target variable can take a discrete set of values are called classification or decision trees. A decision tree is a hierarchical structure that represents a classification model. Internal tree nodes correspond to splits applied to decompose the domain into regions, and terminal nodes (or leaves) assign class labels to regions believed to be sufficiently small or sufficiently uniform. The topmost node in a tree is the root node. (Cichosz, 2015)

Random forests combine two approaches to base model creation: instance sampling (using bootstrap samples) and algorithm nondeterminism. The latter is achieved by randomizing the split selection operation used for decision tree or regression tree growing. The randomization consists in drawing a random subset of available attributes in each node and restricting the subsequent split selection process to splits using attributes from that subset. The usual split evaluation criteria for decision trees or for regression trees are then applied. (Cichosz, 2015)

A standard heuristic is to use the square root of the number of all available attributes as the size of the randomly drawn subset of attributes. Typically, at least several hundred base models are created. Their individual overfitting is canceled out by the aggregation process, which makes the random forest ensemble highly resistant to overfitting. (Han, 2006)

Randomized decision trees or regression trees used as base models for random forests are aggregated via unweighted voting or averaging/summation.

Typically, the first step in a classification problem, a classifier is built describing a predetermined set of data classes. This is the learning step or training phase. In this phase the classification algorithm builds the classifier learning from the training set and its associated class labels.

### **Logistic Regression**

Logistic regression models are statistical models in which an evaluation is made of the relationship between a dependent qualitative, dichotomic variable (binary or binomial logistic regression) and one or more independent explanatory variables, whether qualitative or quantitative. (Hosmer, Lemeshow, & Sturdivant, 2013)

These regression models the probability of some event occurring as a linear function of a set of predictor variables. The logistic regression model is very appropriate for addressing problems where the response variable is of the dichotomic type. (Wilcox, 2019)

It is advisable to adhere to the following recommendations when coding the variables of a logistic regression model, since it facilitates interpretation of the results:

- Dependent variable: we code as 1 the occurrence of the event of interest, and as 0 the absence of the event. For example, in this particular application 1 for fraudulent, 0 for non-fraudulent.

- Independent variables: these may be of different types: - Numerical variable: in order to introduce the variable in the model, it must satisfy the linearity hypothesis, meaning for each unit increase in the numerical variable, the odds of the response variable increase by a constant multiplicative value. (Domínguez-Almendros, Benítez-Parejo, & Gonzalez-Ramirez, 2011)



It is important to note that with logistic function probability  $P$ , ranges between 0 and 1, while logit function can be any real number from minus infinity to positive infinity. For  $P \in [0,1]$ :

$$odds = \frac{P}{1-P} \rightarrow \text{logit}(P) = \ln\left(\frac{P}{1-P}\right) \quad (2.1)$$

Setting logit of  $P$  to be equal to  $mx+b$ :

$$\text{logit}(P) = mx + b \rightarrow mx + b = \ln\left(\frac{P}{1-P}\right) \quad (2.2)$$

$$\left(\frac{P}{1-P}\right) = e^{(mx+b)} \rightarrow \frac{e^{(mx+b)}}{1 + e^{(mx+b)}} \rightarrow P(x) = \frac{1}{1 + e^{-(mx+b)}} \quad (2.3)$$

The logit representation combined with a linear inner representation function is used to represent linear logit classification models, more commonly referred to as logistic regression models. (Belyadi & Haghghat, 2021)

### **Performance Measures**

In many applications, it may not be sufficient to know how often the evaluated model is wrong or even how costly its mistakes are on average. It may be similarly or even more important to know how often it fails to predict correctly particular classes. This is especially true whenever classes of the target concept have different predictability or have different occurrence rates.

In such cases, model performance can be more deeply evaluated based on the confusion matrix. Rows of the confusion matrix correspond to class labels predicted by the model  $h$  and columns correspond to true class labels assigned by the concept  $c$ , as shown below on Table 3. Please note that is it considered the following:

- TP is the number of true positives;
- TN is the number of true negatives;
- FP is the number of false positives;
- FN is the number of false negatives.

Table 3: Confusion matrix

		c (true values)	
		0	1
h (predicted)	0	TN	FN
	1	FP	TP

Considering this denotation, several performance measures can be used.

### Accuracy

Accuracy is the ratio of correctly classified positives and negatives to all instances:

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (2.4)$$

### Precision

Precision is the ratios of instances correctly classified as positive to all instances classified as positive:

$$\frac{TP}{TP + FP} \quad (2.5)$$

### Recall

Recall (or sensitivity) is the ratio of real positives that are correctly predicted as positives:

$$\frac{TP}{TP + FN} \quad (2.6)$$

### 3. Dataset, Software, Questions and Methodology

#### 3.1. Software

Regarding data processing, network creation and as a general tool, Python programming language was used. Within Python several libraries were used including the pandas library (for data manipulation and analysis), and NetworkX library (for studying graphs and networks). Python was also used to run the G-Tries Scanner for all networks.

Jupyter Lab, an open-source web application was used as an interactive computing for Python.

In order to build the classification model, RapidMiner<sup>1</sup>, a data science platform that provides data mining and machine learning procedures, data preprocessing and visualization and evaluation, was used. This tool allows the user to easily build data mining models using blocks.

The version of each software used for developing this work was the following:

- Python - Version: 3.9
- Jupyter Lab - Version 3.4.2
- G-Tries - Version: 0.1
- Rapid Miner - Version: 9.10

#### 3.2. Dataset's Nature

The data that will be used is a synthetic dataset generated by Paysim<sup>2</sup> simulator.

Paysim is a simulation of mobile money transactions with the objective to generate a synthetic transactional data set that can be used for research into fraud detection (Lopez-Rojas, Elmir, & Axelsson, 2016).

The reason why a synthetic dataset was chosen for this study was essentially the struggle in having access to real data due to the intrinsic private nature of financial transactions. Nevertheless, the authors of used techniques such as agent based simulations to create a realistic synthetic data set from a real confidential data set, providing a complete

---

<sup>1</sup> <https://rapidminer.com/>

<sup>2</sup> <https://www.kaggle.com/datasets/ealaxi/paysim1>

and robust dataset very used for fraud detection studies and is very suitable for this particular use including network theory. (Lopez-Rojas, Elmir, & Axelsson, 2016)

In Paysim, the predictive attribute is the presence or absence of fraudulent behaviour associated to the transactions.

In this work, transactions will be seen as relationships within a network perspective.

In order to make a proper analysis of a network, the first step that needs to be considered is the analysis of its characteristics.

Paysim simulates mobile money transactions based on a sample of real transactions extracted from the logs of a mobile money service implemented in an African country. Each row of the dataset describes a transaction.

The information given for each transaction is the ID, the step, the type, the amount, the name of the origin account, the old balance of the origin account, the new balance of the origin account, the name of the destiny account, the old balance of the destiny account, the new balance of the destiny account, if the transaction is fraudulent or not and if is flagged as fraudulent or not.

There are three main entities or agents in the system:

- Clients,
- Merchants and
- Fraudsters.

Clients are the normal customers of the system; merchants play a passive role during the simulation and only serve the clients in certain operations and finally the fraudsters are the threat to the system and the principal focus of our study in fraud detection.

On the following chapters, a description of each column of the dataset will be given.

On Figure 3 and Figure 4 two representations of Paysim network are presented. On the first one the network is randomly presented and in the second one the network is presented with a circular layout. This vast network has a series of smaller networks composed of directly connected nodes, smaller circles of accounts that transfer money among them.

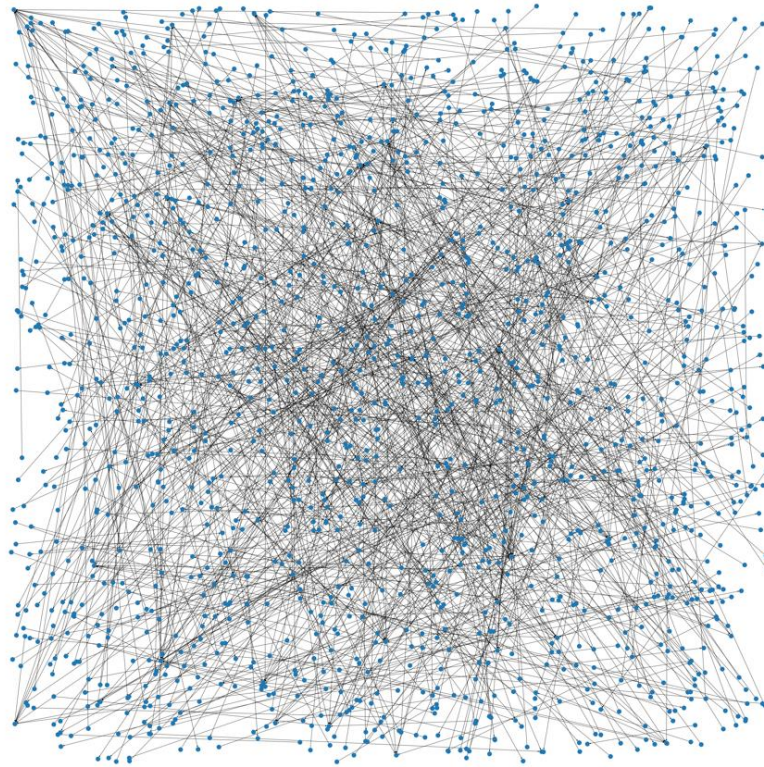


Figure 3: Visualization of the full Paysim dataset network with a random layout. Source: the author.

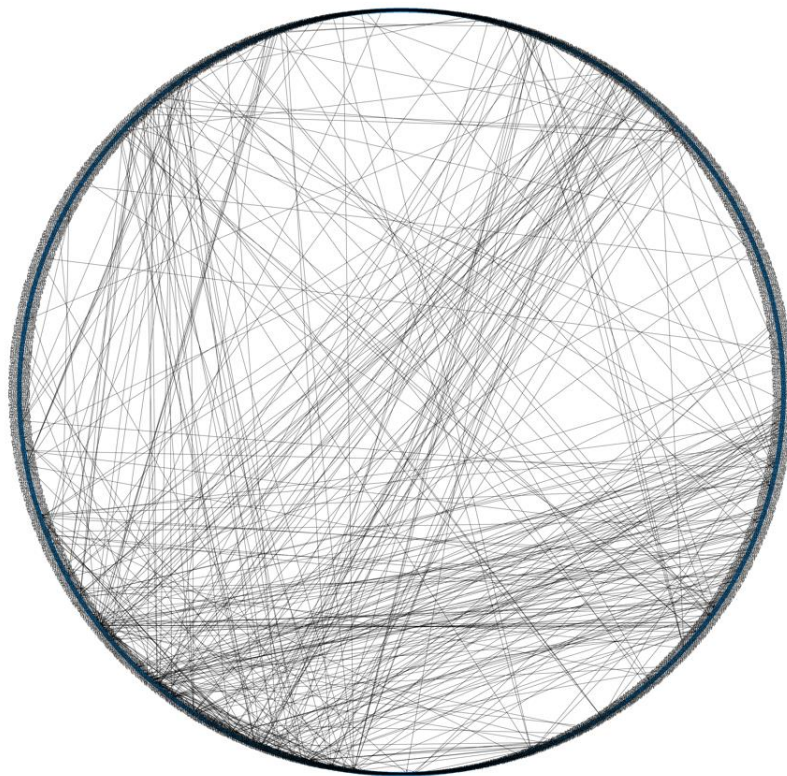


Figure 4: Visualization of the full Paysim dataset network with an circular layout. Source: the author.

### 3.3. Descriptive Analysis of Paysim

COLUMN	MEANING	TYPE	POSSIBLE VALUES [MIN;MAX]		UNIQUE		COUNT		MEAN		STD DEV	
			all data	isFraud	all data	isFraud	all data	isFraud	all data	isFraud	all data	isFraud
-	-	-	all data	isFraud	all data	isFraud	all data	isFraud	all data	isFraud	all data	isFraud
step	time	integer float64	[1;743]	[1;743]	-	-	6.362.620	8.213	243	368	142	216
type	type of transaction between a client and a merchant(possible fraudster)	string object	CASH_OUT PAYMENT CASH_IN TRANSFER DEBIT	-	-	6.362.620	8.213	-	-	-	-	-
amount	amount of the transaction	real float64	[0;92.445.517]	[0;10.000.000]	-	-	6.362.620	8.213	179.862	1.467.967	603.858	2.404.253
nameOrig	Origin/Client account ID	int64	Cxxxxxxxxxx	-	6.353.307	-	6.362.620	8.213	-	-	-	-
oldbalanceOrig	balance of origin account before the transaction occurs	real float64	[0;59.585.040]	[0;59.585.040]	-	-	6.362.620	8.213	833.883	1.649.668	2.888.243	3.547.719
newbalanceOrig	balance of origin account after the transaction occurs	real float64	[0;49.585.040]	[0;49.585.040]	-	-	6.362.620	8.213	855.114	192.393	2.924.049	1.965.666
nameDest	balance of destination account before the transaction occurs	int64	Cxxxxxxxxxx Mxxxxxxxxxx	-	2.722.362	-	6.362.620	8.213	-	-	-	-
oldbalanceDest	balance of destination account before the transaction occurs Note that there is no information for customers that start with M (Merchants).	real float64	[0;356.015.889 ]	[0;236.230.517]	-	-	6.362.620	8.213	1.100.70 2	544.250	3.399.180	3.336.421
newbalanceDest	balance of destination account after the transaction occurs Note that there is no information for customers that start with M (Merchants).	real float64	[0;356.179.279 ]	[0;236.726.495]	-	-	6.362.620	8.213	1.224.99 6	1.279.708	3.674.129	3.908.817
isFraud	This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control of customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.	0/1 float64	[0;1]	-	-	-	-	-	-	-	-	-
isFlaggedFraud	The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction	0/1 float64	[0;1]	-	-	-	-	-	-	-	-	-

In the following paragraphs, we describe the several attributes of Paysim.

## STEP

The column «step» maps a unit of time in the real world. In this case, 1 step is 1 hour of time. Paysim has a total of 744 steps (30 days simulation).

On Table 4, some basic statistics of variable step are presented.

Table 4: Overview of step variable values

COUNT	MEAN	STD	MIN	MAX	25%	50%	75%
6362620	243	142	1	743	156	239	335

As shown on Figure 5 , in the first 14 days of the month there is a higher number of transactions compared to the remaining days. This is perhaps a phenomenon present due to the introduction of income during the first days of the month.

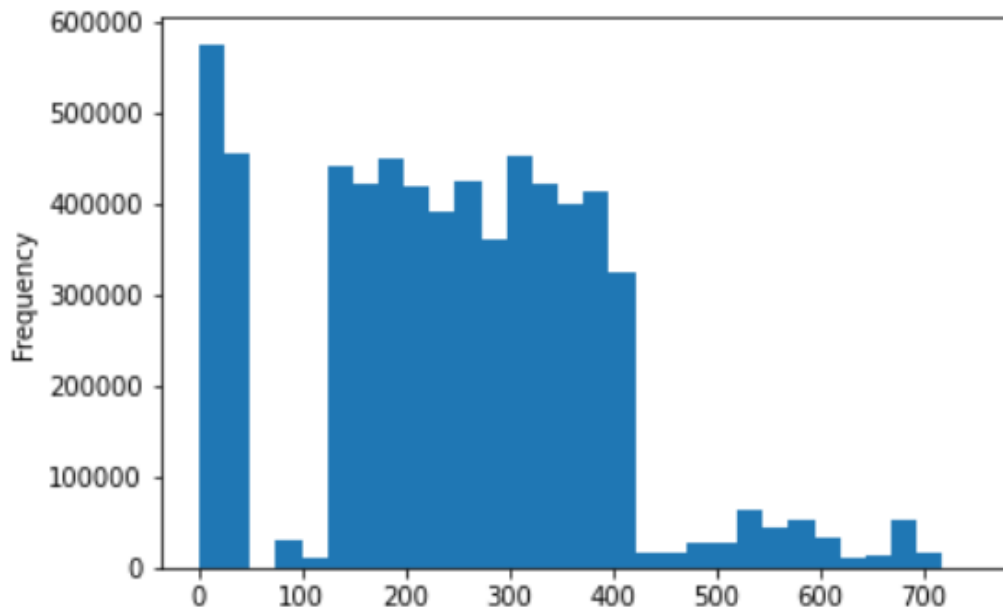


Figure 5: Frequency of transactions per step (hours of the 30 day of the simulation). Source: the author.

## TYPE

As previously mentioned, Paysim is based on real data from an African mobile money transactions service. The mobile money system is a way to digitally send money. However, like any other debit accounts you also have the possibility to withdraw cash or deposit cash.

These operations are only possible using a merchant (or kiosk), the client gives digital money to merchants, and they give back cash in return for CASH\_OUT, similarly to this, is CASH\_IN operation.

The type column is a string that describes the type of the transaction. Transactions can be one of the following 5 types:

- CASH-IN is the process of increasing the balance of account by paying in cash to a merchant;
- CASH-OUT is the opposite process of CASH-IN, it means to withdraw cash from a merchant which decreases the balance of the account;
- DEBIT is similar process than CASH-OUT and involves sending the money from the mobile money service to a bank account;
- PAYMENT is the process of paying for goods or services to merchants which decreases the balance of the account and increases the balance of the receiver;
- TRANSFER is the process of sending money to another user of the service through the mobile money platform.

On, some basic statistics of variable step are presented.

Table 5: Relative frequency of transaction types

TRANSACTION TYPE	FREQUENCY	%
CASH_OUT	2237500	35%
PAYMENT	2151495	37%
CASH_IN	1399284	22%
TRANSFER	532909	8%
DEBIT	41432	1%
TOTAL	6362620	100%

## AMOUNT

The column «amount» includes the values of the amount of each transaction on local currency. On Table 6 , some basic statistics of variable amount are presented.



Table 6: Overview of amount variable values

COUNT	MEAN	STD	MIN	MAX	25%	50%	75%
6362620	179862	603858	0	92445517	13390	74872	208721

Most of the transactions (75%) are below the amount of 208721. Nevertheless, as observed on Table 7, where the mean amount values per transaction type is depicted, the mean amount for TRANSFER transactions is 910647.

Table 7: Overview of amount values per transaction type

TYPE	COUNT	MEAN	STD	MIN	MAX
<b>CASH_IN</b>	1399284	168920	126508	0	1915268
<b>CASH_OUT</b>	2237500	176274	175330	0	10000000
<b>DEBIT</b>	41432	5484	13319	1	569078
<b>PAYMENT</b>	2151495	13058	12556	0	238638
<b>TRANSFER</b>	532909	910647	1879574	3	92445517

## ACCOUNTS

The column «nameOrig» contains the name of the origin account and the column «nameDest» contains the name of the destiny account.

If the account name starts with a C, it means the account is a customer account. If on the other hand, the account name starts with a M, it means the account is a merchant account.

Looking at all different accounts, customer and merchants, origin or destiny accounts, there are a total of 9073900 different accounts on the dataset.

Most of the origin accounts appear only once on the data set but some accounts appear in 2 or 3 transactions, as depicted below on Table 8.

Table 8: Number of times each account on «nameOrig» appears on the transaction dataset

NAMEORIG	COUNT
C1065307291	3
C1784010646	3
C1902386530	3
C1832548028	3
C545315117	3
...	...
C1645325210	1
C1645325020	1
C1645324530	1
C1645324143	1
C999999784	1

On Table 9, some basic statistics of nameOrig accounts are presented.

Table 9: Overview of nameOrig accounts

COUNT	UNIQUE	TOP	FREQ.
6362620	6353307	C1902386530	3

Contrary to the origin accounts, destiny accounts appear more repeatedly, the most feature destiny account is C11286084959, being the destiny account on 113 transactions. On Table 10, some basic statistics of nameDest accounts are presented.

Table 10: Overview of nameDest accounts

COUNT	UNIQUE	TOP	FREQ.
6362620	2722362	C11286084959	113

Since there is not record of balance from clients that start with M (Merchants), these types of accounts will be not given as much importance.

## OLD AND NEW BALANCES

The «oldbalanceOrg» is the initial balance of the origin account before the transaction. On **Erro! A origem da referência não foi encontrada.**, some basic statistics of oldbalanceOrg accounts are presented.

Table 11: Overview of oldbalanceOrg

COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
6362620	833883	2888243	0	0	14208	107315	59585040

The «newbalanceOrg» is the final balance of the origin account after the transaction. On Table 11, some basic statistics of newbalanceOrg accounts are presented.

Table 12: Overview of newbalanceOrg

COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
6362620	855114	2924049	0	0	0	144258	49585040

The «oldbalanceDest» is the initial balance of the destiny account before the transaction. On Table 13, some basic statistics of oldbalanceDest accounts are presented. Note that there is not information for customers that start with M (Merchants).

Table 13: Overview of oldbalanceDest

COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
6362620	1100702	3399180	0	0	132706	943037	356015889

The «newbalanceDest» is the final balance of the destiny account after the transaction. On Table 14, some basic statistics of newbalanceDest accounts are presented.

Table 14: Overview of newbalanceDest

COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
6362620	1224996	3674129	0	0	214661	1111909	356179279

Note that there is no information for the accounts that start with M (Merchants), as depicted on Table 15.

Table 15: Overview of Merchants accounts

STEP	TYPE	AMOUNT	NAMEORIG	OLDBALANCE ORG	NEWBALANCE ORIG	NAMEDEST	OLDBALANCE DEST	NEWBALANCE DEST	ISFRAUD	ISFLAGGED FRAUD
1	PAYMENT	9840	C1231006815	170136	160296	M1979787155	0	0	0	0
1	PAYMENT	1864	C1666544295	21249	19385	M2044282225	0	0	0	0
1	PAYMENT	11668	C2048537720	41554	29886	M1230701703	0	0	0	0
1	PAYMENT	7818	C90045638	53860	46042	M573487274	0	0	0	0
1	PAYMENT	7108	C154988899	183195	176087	M408069119	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
718	PAYMENT	8178	C1213413071	11742	3564	M1112540487	0	0	0	0
718	PAYMENT	17841	C1045048098	10182	0	M1878955882	0	0	0	0
718	PAYMENT	1023	C1203084509	12	0	M675916850	0	0	0	0
718	PAYMENT	4110	C673558958	5521	1411	M1126011651	0	0	0	0
718	PAYMENT	8634	C642813806	518802	510168	M747723689	0	0	0	0

## ISFRAUD

This variable identifies a fraudulent transaction (1) and non-fraudulent (0). These are the transactions made by the fraudulent agents inside the dataset. In this specific dataset the fraudulent behaviour of the agents aims to profit by taking control or customers' accounts and try to empty the funds by transferring to another account and then cashing out of the system. On Table 16, some basic statistics of isFraud are presented.

Table 16: Overview of isFraud

ISFRAUD	0	1
<b>count</b>	6354407	8213
<b>mean</b>	178197	1467967
<b>std</b>	596237	2404253
<b>min</b>	0	0
<b>max</b>	92445517	10000000

## ISFLAGGEDFRAUD

This variable flags illegal attempts to transfer more than 200.000 in a single transaction. The aim is to control massive transfers from one account to another and flags illegal attempts. On Table 17, some basic statistics of isFlaggedFraud are presented.

Table 17: Overview of isFlaggedFraud

ISFLAGGEDFRAUD	0	1
<b>count</b>	6362604	16
<b>mean</b>	179850	4861598
<b>std</b>	603788	3572499
<b>min</b>	0	3538874
<b>max</b>	92445517	10000000

When the record has the isFlaggedFraud = 1, this means that the transaction was detected and stopped from being processed, that is the reason why it doesn't affect the account destination/origin, as observed on Table 18.

Table 18: Overview of is FlaggedFraud ==1

STEP	TYPE	AMOUNT	NAMEORIG	OLDBALANCE ORG	NEWBALANCE ORIG	NAMEDEST	OLDBALANCE DEST	NEWBALANCE DEST	ISFRAUD	ISFLAGGED FRAUD
212	TRANSFER	4953893.08	C728984460	4953893.08	4953893.08	C639921569	0.0	0.0	1	1
250	TRANSFER	1343002.08	C1100582606	1343002.08	1343002.08	C1147517658	0.0	0.0	1	1
279	TRANSFER	536624.41	C1035541766	536624.41	536624.41	C1100697970	0.0	0.0	1	1
387	TRANSFER	4892193.09	C908544136	4892193.09	4892193.09	C891140444	0.0	0.0	1	1
425	TRANSFER	10000000.00	C689608084	19585040.37	19585040.37	C1392803603	0.0	0.0	1	1
425	TRANSFER	9585040.37	C452586515	19585040.37	19585040.37	C1109166882	0.0	0.0	1	1
554	TRANSFER	3576297.10	C193696150	3576297.10	3576297.10	C484597480	0.0	0.0	1	1
586	TRANSFER	353874.22	C1684585475	353874.22	353874.22	C1770418982	0.0	0.0	1	1
617	TRANSFER	2542664.27	C786455622	2542664.27	2542664.27	C661958277	0.0	0.0	1	1
646	TRANSFER	10000000.00	C19004745	10399045.08	10399045.08	C1806199534	0.0	0.0	1	1
646	TRANSFER	399045.08	C724693370	10399045.08	10399045.08	C1909486199	0.0	0.0	1	1

### 3.4. Dataset and Fraud

Taking a closer look on the fraudulent transactions of the dataset, depicted on Table 19, we can state that the mean mount for fraudulent transactions is 1467967. In the total of 6.362.620 transactions, 8213 are fraudulent.

Table 19: Overview of fraudulent transactions

PARAMETER	COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
<b>step</b>	8213	368	216	2	181	367	558	743
<b>amount</b>	8213	1467967	2404253	1	127091	441423	1517771	10000000
<b>oldBalanceOrg</b>	8213	1649668	3547719	0	125822	438983	1517771	59585040
<b>newbalanceOrig</b>	8213	192393	1965666	0	0	0	0	49585040
<b>oldbalanceDest</b>	8213	544250	3336421	0	0	0	147829	236230517
<b>newbalanceDest</b>	8213	1279708	3908817	0	0	4676	1058725	236726495
<b>isFraud</b>	8213	1	0	1	1	1	1	1
<b>isFlaggedFraud</b>	8213	0	0	0	0	0	0	1

On the other hand, all fraudulent transactions are equally divided between either CASH\_OUT or TRANSFER types, as shown on Table 20.

Table 20: Fraudulent transactions per transaction type

TYPE	COUNT	MEAN	STD	MIN	MAX
<b>CASH_OUT</b>	4116	1455103	2393842	0	10000000
<b>TRANSFER</b>	4097	1480892	2414890	64	10000000



### Sub Networks of directly connected accounts

From the original dataset of transactions (big Paysim network that included 6362620 transactions) a set of smaller networks were identified, with accounts (nodes) with uninterrupted transactions (edges). For all the accounts presents in Paysim, the accounts transferring money directly to each other were included in the same network. This means that if two accounts are part of the same network, they are either directly connected, or they have accounts relating each other. Networks that included only two accounts transferring money to each other were removed, so that a network has a minimum of 3 accounts.

The result was a set of 456187 smaller networks from 3 to 121 nodes, as depicted on Figure 6.

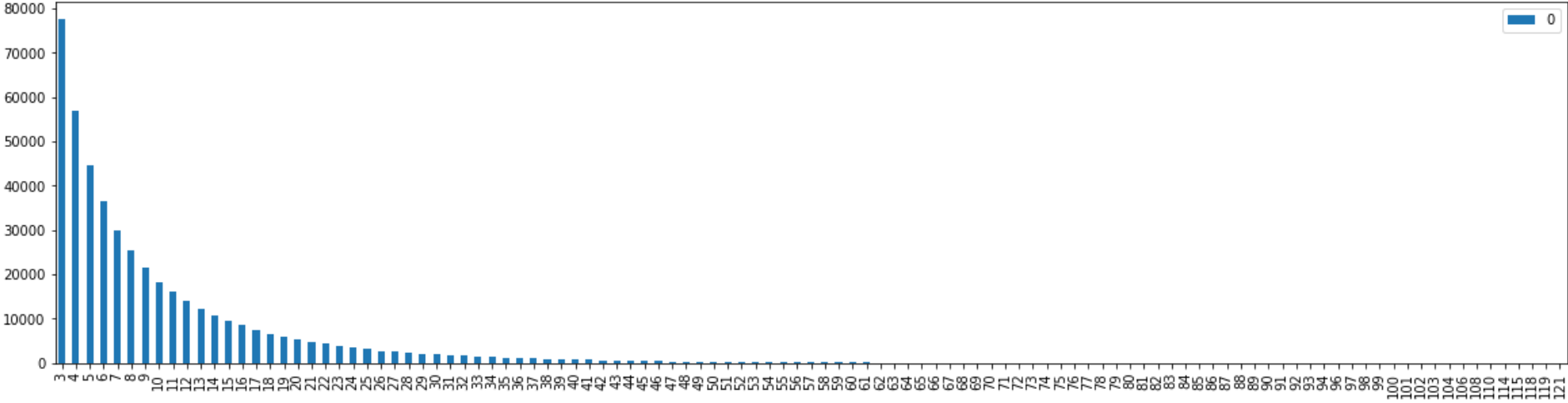


Figure 6: Frequency of networks per network size (nodes). Source: the author.

On Figure 7 – Network 1, one of the sub network presents on Paysim is represented. , as it can be seen this network has 8 nodes and 7 edges or transactions.

Most of the networks are in a star shaped, like Network 1 and Network 2, in this networks there all nodes are making transactions to the same node.

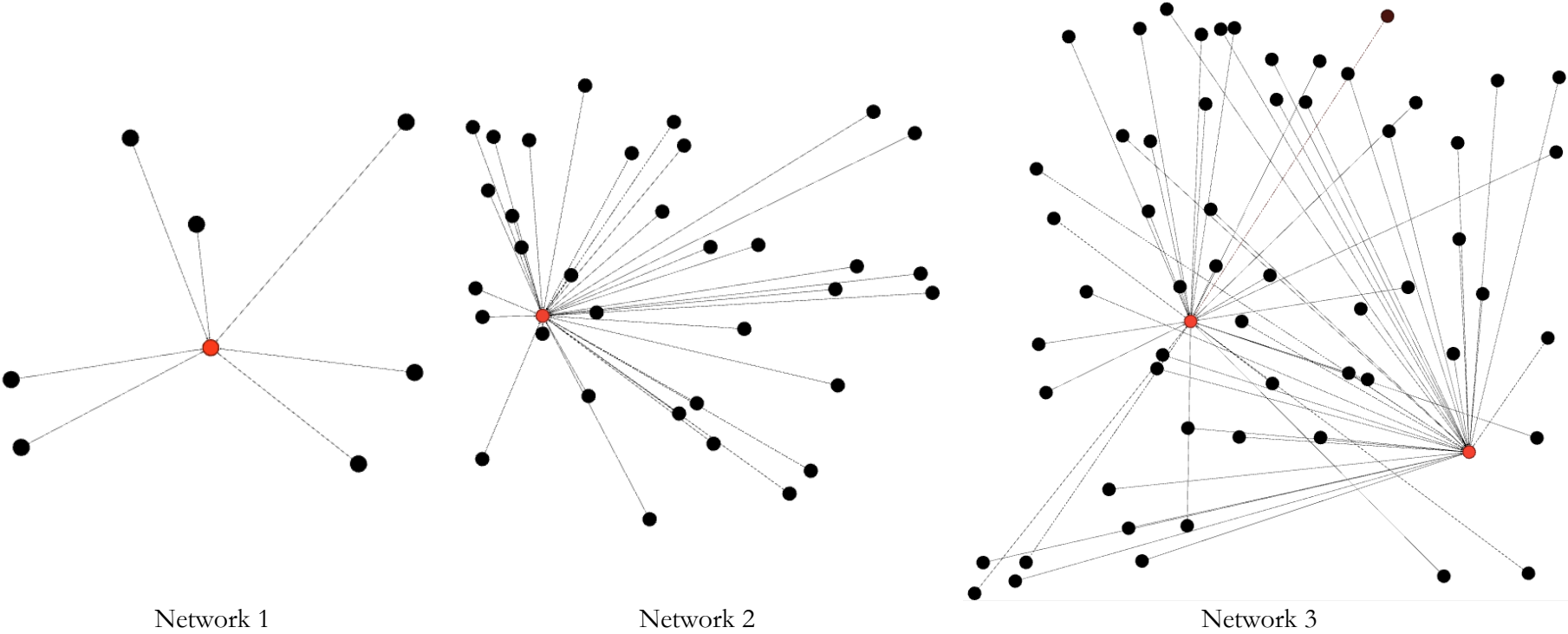


Figure 7: Visualisation of Paysim's Sub networks. Source: the author.

Some network measurements were also calculated such as the node transitivity and node degree centrality.

The transitivity or clustering coefficient of a network is a measure of the tendency of the nodes to cluster together. The transitivity of a node measures the average probability that two neighbors of a vertex are themselves neighbors. High transitivity means that the network contains communities or groups of nodes that are densely connected internally.

The degree centrality assigns an importance score based simply on the number of links held by each node. Basically for each account it tells us the number links that account has, the number of transactions.

On Table 21, an overview of the non-fraudulent transactions in the sub networks it is presented.

Table 21: Description of non-fraudulent (isFraud==0) sub networks transactions

PARAMETER	COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
<b>step</b>	4101972	238	140	1	154	235	331	718
<b>amount</b>	4101972	264248	725509	1	76156	159291	278979	92445520
<b>oldBalanceOrg</b>	4101972	1244402	3510193	0	0	16946	198427	43818860
<b>newbalanceOrig</b>	4101972	1282214	3553259	0	0	0	290327	43686620
<b>oldbalanceDest</b>	4101972	1702163	4104323	0	160314	581382	1744077	356015900
<b>newbalanceDest</b>	4101972	1890756	4429070	0	239696	715403	1961666	356179300

<b>isFraud</b>	4101972	0	0	0	0	0	0	0
<b>isFlaggedFraud</b>	4101972	0	0	0	0	0	0	0

On Table 22, an overview of the non-fraudulent transactions in the sub networks it is presented.

Table 22: Description of fraudulent (isFraud== 1) sub networks transactions

PARAMETER	COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
<b>step</b>	5553	291	203	1	120	254	437	743
<b>amount</b>	5553	1415542	2321782	0	127091	432958	1503035	10000000
<b>oldBalanceOrg</b>	5553	1487364	2830166	0	125822	430722	1503035	59585040
<b>newbalanceOrig</b>	5553	74762	1165376	0	0	0	0	49585040
<b>oldbalanceDest</b>	5553	797852	4027940	0	0	0	440777	236230500
<b>newbalanceDest</b>	5553	1750765	4592590	0	0	387460	236230500	236726500
<b>isFraud</b>	5553	1	0	1	1	1	1	1
<b>isFlaggedFraud</b>	5553	0	0	0	0	0	0	1

Comparing the information of the fraudulent and non-fraudulent transactions a few deductions can already be made. The amount of the mean fraudulent transaction is significant higher. The mean value of the new balance on the Origin account is lower on the fraudulent transactions. The old balance on the destiny accounts is also significantly lower when there is fraud.

### 3.5. Questions

Now that we are more familiar with the dataset and Paysim networks and subnetworks, we can establish a series of questions that may or not be directly linked with fraudulent transactions.

These questions focus on the relationship between some network features and fraud, and are defined taken into account the available information from the original dataset, network parameters, the potential of network motifs in fraud detection and the tools available for network detection.

The formulation of these questions were based on Paysim dataset attributes and network theory: testing the possibility of identifying networks patterns and network metrics such as degree centrality and transitivity (or clustering coefficient) that include information not only of the account itself but the surrounding accounts as well.

Questions:

1. Is there a relationship between fraud and network subgraphs/motifs?
2. Is there a relationship between node centrality and fraud?
3. Is there a relationship between node transitivity and fraud?
4. Is there a relationship between the amount of the transaction and fraud?
5. Is there a relationship between transaction type and fraud?
6. Is there a relationship between the step (time stamp of the transaction and fraud?
7. Is there a relationship between the account balance before and after the transaction and fraud?

### 3.6. Gtries

A G-Tries is a data structure specialized in efficiently representing sets of subgraphs.

A g-trie is a multiway tree encapsulates the structure of the entire graph set, taking advantage of common topologies in the same way a prefix tree takes advantage of common prefixes. This avoids redundancy in the representation of the graphs, thus allowing for both memory and computation time savings.

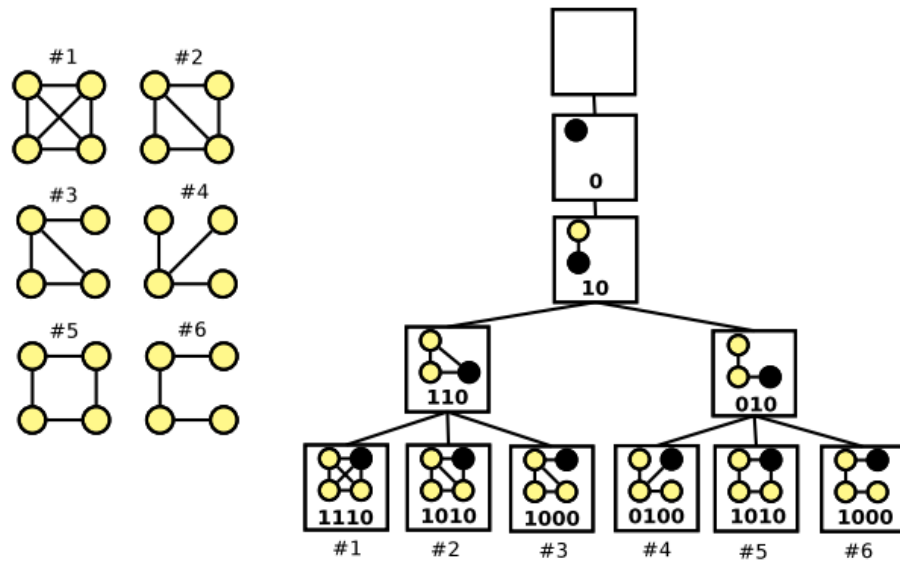
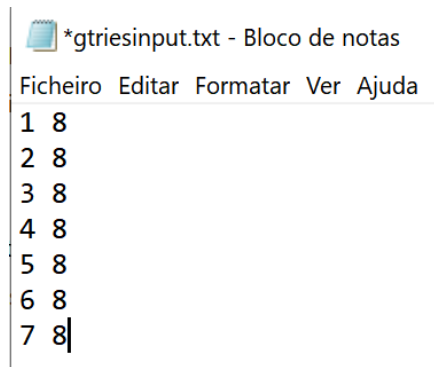


Figure 8: G-tries' example. (Ribeiro & Silva, 2014)

G-Tries can significantly compress the representation of a set of subgraphs and when compared with previously existent algorithms. The execution times can be orders of magnitude better and in particular with small motif sizes and not too dense networks the advantage in using g-tries is overwhelming. It can be used to do the more usual network census (looking for triads), or to discover larger motifs, specifying beforehand larger subgraphs. (Ribeiro & Silva, 2010)

### 3.7. Three Nodes Subgraphs

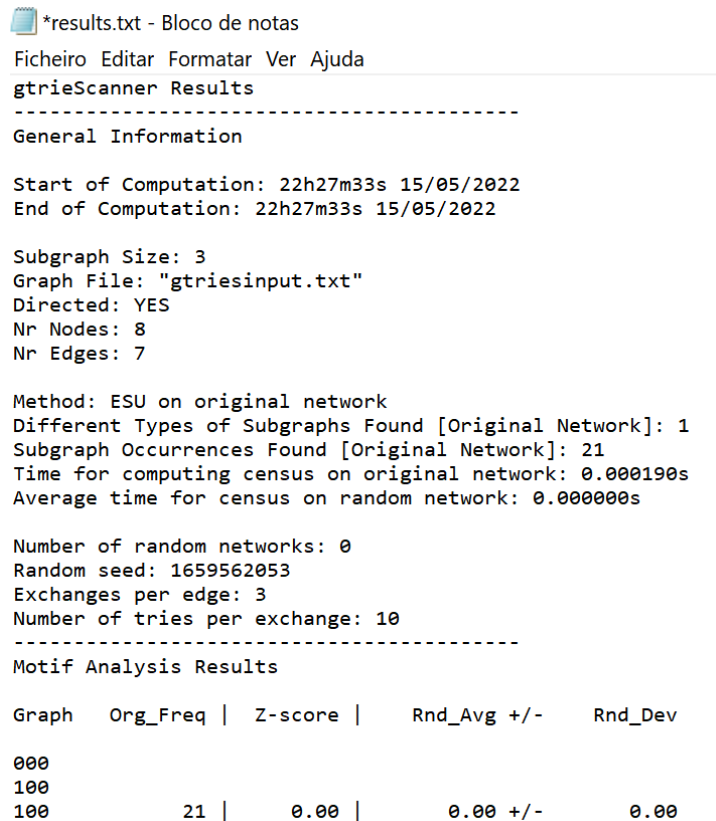
To search for four nodes subgraphs g-tries was executed with the command presented on Annex 8.3. g-tries looked for subgraphs with size 3, directed subgraphs, unweighted (simple) using the method esu, the input was written in gtriesinput.txt and the occurrences of each graph was written the file.txt.



```
*gtriesinput.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
1 8
2 8
3 8
4 8
5 8
6 8
7 8
```

Figure 9: gtries input file. Source: the author.

The output file for the output of g-tries and for the number of occurrences of each graph are presented below on Figure 10 and Figure 11, respectively.



```
*results.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
gtrieScanner Results
-----
General Information

Start of Computation: 22h27m33s 15/05/2022
End of Computation: 22h27m33s 15/05/2022

Subgraph Size: 3
Graph File: "gtriesinput.txt"
Directed: YES
Nr Nodes: 8
Nr Edges: 7

Method: ESU on original network
Different Types of Subgraphs Found [Original Network]: 1
Subgraph Occurrences Found [Original Network]: 21
Time for computing census on original network: 0.000190s
Average time for census on random network: 0.000000s

Number of random networks: 0
Random seed: 1659562053
Exchanges per edge: 3
Number of tries per exchange: 10
-----
Motif Analysis Results

Graph  Org_Freq | Z-score | Rnd_Avg +/-  Rnd_Dev
000
100
100          21 |  0.00 |  0.00 +/-  0.00
```

Figure 10: g-tries output file. Source: the author.



file.txt - Bloco de notas

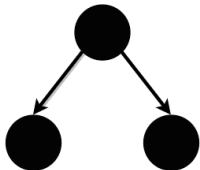
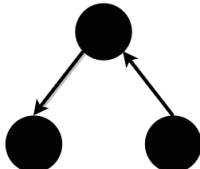
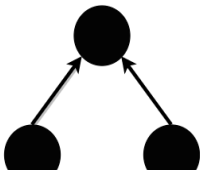
Ficheiro Editar Formatar

```
000100100: 1 8 7  
000100100: 1 8 6  
000100100: 1 8 5  
000100100: 1 8 4  
000100100: 1 8 3  
000100100: 1 8 2  
000100100: 2 8 7  
000100100: 2 8 6  
000100100: 2 8 5  
000100100: 2 8 4  
000100100: 2 8 3  
000100100: 3 8 7  
000100100: 3 8 6  
000100100: 3 8 5  
000100100: 3 8 4  
000100100: 4 8 7  
000100100: 4 8 6  
000100100: 4 8 5  
000100100: 5 8 7  
000100100: 5 8 6  
000100100: 6 8 7
```

Figure 11: g-tries occurrences file. Source: the author.

The three nodes subgraphs found on the networks are depicted on Table 23.

Table 23: Three nodes subgraphs found running g-tries on Paysim sub networks

SUBGRAPH	REPRESENTATION
011000000	
001100000	
000100100	

Observing the presence of subgraphs in the networks it is not clear if there is any association between a particular 3 nodes subgraph and fraudulent transactions.

Table 24: Frequency of three nodes subgraphs in fraudulent networks

SUBGRAPH	TOTAL FREQUENCY	FREQUENCY IN FRAUDULENT NETWORKS	
		Absolute	Relative%
011 000 000	9030	159	1.76%
001 100 000	1762	34	1.93%
000 100 100	454857	5506	1.21%

### 3.8. Four Nodes Subgraphs

To search for four nodes subgraphs g-tries was executed with the presented on Annex 8.3. g-tries looked for subgraphs with size 4, directed subgraphs, unweighted (simple) using the method esu, the input was written in gtriesinput.txt and the occurrences of each graph was written on file.txt.

```
gtriesinput.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
1 4
2 4
3 4
5 4
6 4
```

Figure 12: gtries input file. Source: the author.

The output file for the output of g-tries and for the number of occurrences of each graph are presented below on Figure 13 and Figure 14, respectively.

```
gtrieScanner Results
-----
General Information

Start of Computation: 22h31m38s 27/05/2022
End of Computation: 22h31m38s 27/05/2022

Subgraph Size: 4
Graph File: "gtriesinput.txt"
Directed: YES
Nr Nodes: 6
Nr Edges: 5

Method: ESU on original network
Different Types of Subgraphs Found [Original Network]: 1
Subgraph Occurrences Found [Original Network]: 10
Time for computing census on original network: 0.000094s
Average time for census on random network: 0.000000s

Number of random networks: 0
Random seed: 1658957498
Exchanges per edge: 3
Number of tries per exchange: 10
-----
Motif Analysis Results

Graph  Org_Freq | Z-score | Rnd_Avg +/-  Rnd_Dev
0000
1000
1000
1000          10 |   0.00 |   0.00 +/-   0.00
```

Figure 13: g-tries output file. Source: the author.

file.txt - Bloco de notas

```
Ficheiro Editar Formatar Ver Ajuda
0000100010001000: 1 4 6 5
0000100010001000: 1 4 6 3
0000100010001000: 1 4 6 2
0000100010001000: 1 4 5 3
0000100010001000: 1 4 5 2
0000100010001000: 1 4 3 2
0000100010001000: 2 4 6 5
0000100010001000: 2 4 6 3
0000100010001000: 2 4 5 3
0000100010001000: 3 4 6 5
```

Figure 14: g-tries occurrences file. Source: the author.

The four nodes subgraphs found on the networks are depicted on Table 25

Table 25: Four nodes subgraphs found running g-tries on Paysim subnetworks

SUBGRAPH	REPRESENTATION
0111 1010 1100 1000	
0111 1000 1000 1000	
0110 1001 1000 0100	

Considering the presence of subgraphs in both fraudulent and non-fraudulent transactions it is not evident if there is any association between a particular 4 nodes subgraph and fraudulent transactions.

Table 26: Frequency of three nodes subgraphs in fraudulent networks

SUBGRAPH	TOTAL FREQUENCY	FREQUENCY IN FRAUDULENT NETWORKS	
		Absolute	Relative%
0111 1010 1100 1000	2	1	50.00%
0111 1000 1000 1000	321404	4249	1.32%
0110 1001 1000 0100	2320	50	2.15%

## 4. Classification Problem

The present chapter describes how the Random Forest model was created, the blocks used and the division of the dataset for training and testing.

### 4.1. Random Forest

Using RapidMiner<sup>3</sup>, the process for building the model and testing its performance was created.

To build the model a dataset made from the Paysim data and the information of the sub networks present in the larger network of Paysim was used. This data has 49701 transactions and took about 15 hours to write.

The process starts with the dataset, then setting the role i.e. identifying the goal of the classification. In this case the role is the label «hasfraud» which defines if a transaction is fraudulent or non-fraudulent. The data is split in the ratio 70:30, as in the holdout cross validation approach, where 70% of the data will be used for the model creation or training, note that this data is sent to the Random Forest block. The remaining data (30%) will be used for testing the model (Apply Model) and finally the performance of the model is evaluated (Performance block).

The dataset contains information about the transactions (amount, step, old balance origin, new balance origin, new balance destination and account type) and network measures as the transitivity (or clustering coefficient) and degree centrality values for the sender and receiver accounts. Please note that the network measures are calculated within each set of sub network and not taking into account the major Paysim network, in order to get information from the smaller clusters of account that transfer money directly to each other. The model and the dataset used can be seen on Annex 8.4.

---

<sup>3</sup> <https://rapidminer.com/>

The transitivity or clustering coefficient of a network is a measure of the tendency of the nodes to cluster together. The transitivity of a node measures the average probability that two neighbors of a vertex are themselves neighbors. High transitivity means that the network contains communities or groups of nodes that are densely connected internally.

The degree centrality assigns an importance score based simply on the number of links held by each node. Basically for each account it tells us the number links that account has, the number of transactions.

Please note that for the accounts type the following notation was considered:

- 1 - transaction between customer and customer;
- 2 - transaction between merchant and merchant;
- 3 - transaction between customer and merchant.

Certain combinations of parameters were included and excluded from the dataset and the model ran with the different input datasets, in order to investigate which columns or parameters deliver a better performance, allowed us to create a better classification model.

The results, including the parameters present on the dataset and the performance indicators obtained for each model are presented below on Table 27.



Table 27: Models results per dataset

DATASET	PARAMETERS	ACCURACY	PRECISION	RECALL
DB_A	amount, step accounts type	99.91%	0%	0%
DB_B	old balance origin, new balance origin, old balance destination, new balance destination,	99.91%	0%	0%
DB_C	transitivity sender, transitivity receiver, centrality sender, centrality receiver	99.91%	0%	0%
DB_D	amount, step, old balance origin, centrality receiver	99.91%	0%	0%

The number of true positives (fraudulent transactions identified as fraudulent by the model) is zero. This happens because the number of fraudulent transactions on the dataset is too low, in 49701 transactions only 46 are fraudulent (0.09%).

Nevertheless, the parameters identified as most relevant, responsible for causing the branches or splits in the decision trees were the amount, the step, the old balance of the origin account and the centrality of the receiver account.

In order to overcome the difficulty of having a very low fraudulent transaction ratio the technique of Under Sampling was used. This technique involves deleting random samples

from the majority class to compensate for an imbalance that is present in the data. (Chawla, 2005)

To have more transactions a larger data base was created. This dataset has 784002 transactions and took about 78 h to write. In these 784004 transactions, 761 are fraudulent.

Random Under Sampling was applied to this data and the final training dataset has 1522 transactions, of which 761 are fraudulent and 761 are non-fraudulent. The final RapidMiner process is presented on Annex 8.4.

## 4.2. Logistic Regression

In order to have a better understanding about the influence of each attribute on the model prediction a Logistic Regression Model was implemented. The final RapidMiner process is presented on Annex 8.4.

The final results, including the parameters present on the dataset and the performance indicators obtained are presented in the next chapter.

## 5. Model Results and Performance

The present chapter describes the parameters used for the Random Forest model. The performance of the model and the final results are presented. The same is presented for the Logistic Regression model.

### 5.1. Random Forest

At this point the final dataset has the following parameters: amount, step, accounts type, old balance origin, new balance origin, old balance destination, new balance destination, transitivity sender, transitivity receiver, centrality sender and centrality receiver. The model is being fed with the Under Sampled dataset for training and the original dataset for testing.

The Random Forest method was applied using the parameters described on Table 28.

Table 28: RapidMiner Random Forest calculation parameters

PARAMETER	VALUE	DESCRIPTION
Number of Trees	1000	This parameter specifies the number of random trees to generate.
Criterion	gain_ratio	Selects the criterion on which attributes will be selected for splitting. For gain_ratio is a variant of information gain that adjusts the information gain for each attribute to allow the breadth and uniformity of the attribute values.
Maximal Depth	7	The depth of a tree varies depending upon the size and characteristics of the dataset.
Apply Pruning	yes	The random trees of the random forest model can be pruned after generation. If checked, some branches are replaced by leaves according to the confidence parameter.
Confidence	0.01	This parameter specifies the confidence level used for the pessimistic error calculation of pruning.
Apply PrePruning	yes	This parameter specifies if more stopping criteria than the maximal depth should be used during generation of the decision trees.
Minimal Gain	0.01	The gain of a node is calculated before splitting it. The node is split if its gain is greater than the minimal gain. A

higher value of minimal gain results in fewer splits and thus smaller trees. A value that is too high will completely prevent splitting and trees with single nodes are generated.

Please note, the dataset with Under Sampling (1522 transactions) was used for training and the total dataset (784003 transactions) was used for testing.

The overall accuracy obtained was 93.99%. The class recall for the fraudulent transaction was 98.03% and the class precision was 1.56%.

Table 29: Random Forest model performance after Under Sampling

	TRUE 0 (NON-FRAUD.)	TRUE 1 (FRAUD.)	CLASS PRECISION
PRED. 0 (NON-FRAUD.)	736161	15	100.00%
PRED. 1 (FRAUD.)	47081	746	1.56%
CLASS RECALL	93.99%	98.03%	-

The ratio of correctly classified positives and negatives to all instances is high. The recall or sensitivity, the ratio of real fraudulent transactions classified as fraudulent is high. Nonetheless, the precision is low. Precision of the ratio of instances correctly classified as positive to all instances classified as positive. Which means for the fraudulent transactions class, precision is given by:

$$\frac{746}{746 + 47081} = 1.56\% \quad (5.1)$$

This happens because of the fact that the fraudulent transactions represent a very small portion of all transactions. In real life this means a fraudulent red flag would be given to the major part of the fraudulent transactions (98.03%) are classified as fraudulent but a lot of non-fraudulent transactions are also classified as fraudulent by the model.

## 5.2. Logistic Regression

Using the Under Sampling dataset for training and the full dataset for testing the results obtained for the Logistic Regression model are depicted below, on Table 30.

Table 30: Logistic Regression results

ATTRIBUTE	COEFFICIENT	STD. COEFFICIENT	STD.ERROR	Z-VALUE	P-VALUE
amount	-9.52E-7	-1.61	1.49E-6	-0.64	0.521
step	1.65E-4	0.0284	5.88E-4	0.28	0.78
oldBalanceOrig	2.10E-5	65.36	1.49E-6	14.12	0.0
oldBalanceDest	6.08E-6	19.44	1.20E-6	5.06	4.15E-7
newBalanceOrig	-2.45E-5	-64.90	2.10E-6	-11.64	0.0
newBalanceDest	-6.20E-6	-22.60	1.20E-6	-5.16	2.42E-7
accountsType	0.0	0.0	NaN	NaN	NaN
transitivitySender	0.0	0.0	NaN	NaN	NaN
transitivityReceiver	0.0	0.0	NaN	NaN	NaN
centralitySender	-0.153	-0.0129	1.35	-0.116	0.909
centralityReceiver	2.300	0.293	0.99	2.30	0.0212
Intercept	-2.73	6.40	0.98	-2.79	0.00529

Looking at the coefficients for each attribute on the Logistic Regression output, we can notice right away that the attributes `accountsType`, `transitivitySender` and `transitivityReceiver` have no influence on the output (fraud. Or non-fraud.) since their coefficient is zero. On the other hand, attributes such as `centralityReceiver`, `step`, `oldBalanceOrig` and `oldBalanceDest` have positives coefficients, meaning its increase in value is reflected in an increase on the output. The probability of the transaction being fraudulent is increased with the increase of this attributes. The

probability for the transaction to be fraudulent is inversely proportional to the centralitySender, newBalanceOrig, newBalanceDest and amount attributes.

The overall accuracy obtained was 94.29%%. The class recall for the fraudulent transaction was 86.33% and the class precision was 1.45%.

Table 31: Logistic Regression Model performance after Under Sampling

	TRUE 0 (NON-FRAUD.)	TRUE 1 (FRAUD.)	CLASS PRECISION
PRED. 0 (NON-FRAUD.)	738611	104	100.00%
PRED. 1 (FRAUD.)	44631	657	1.56%
CLASS RECALL	94.30%	98.03%	-

Taking a closer look into the analysis done to Paysim and the random trees created by the model and see what were the branches and divisions used for the classification, we can answer the questions made before the creation of the model.

1. Is there a relationship between fraud and network subgraphs/motifs?

No evident relationship was found for 3 nodes and neither for 4 nodes subgraphs. There was no direct association found between any 3 node or 4 node subgraphs and the fraudulent transactions. The subgraphs found in networks with fraudulent and non- fraudulent transactions were the same.

2. Is there a relationship between node centrality and fraud?

There is some evidence point that way. In most trees the centrality of the sender is used to divide the decision tree and top classify the transaction as fraudulent or not. With origin accounts with higher values of centrality (higher than 0.029 in this dataset) being more prone to be linked with fraudulent transactions. The same applies to the centrality of the destiny accounts. Even though it is not as frequent as an attribute as the centrality of the origin account, higher values of centrality of the destiny account appear to be linked with fraudulent transactions. An example of this can be seen on the result of one decision tree, in Figure 15.

Additionally, on the Logistic Regression centralitySender has a negative coefficient, meaning the higher the centrality of the sender account, the fewer the odds that the transaction is fraudulent. centralityReceiver's coefficient is the higher coefficient in the regression. The higher the centrality value of the destiny account, the more likely is for the transaction to be fraudulent.

## Tree

```
oldBalanceOrig > 157.500
|   newBalanceOrig > 547.105
|   |   centralitySender > 0.118
|   |   |   amount > 2669570.695: 1 {0=0, 1=6}
|   |   |   amount ≤ 2669570.695: 0 {0=39, 1=0}
|   |   centralitySender ≤ 0.118: 0 {0=233, 1=0}
|   newBalanceOrig ≤ 547.105
|   |   centralityReceiver > 0.307: 1 {0=146, 1=797}
|   |   centralityReceiver ≤ 0.307: 0 {0=3, 1=2}
oldBalanceOrig ≤ 157.500
|   oldBalanceDest > 14911.835: 0 {0=289, 1=4}
|   oldBalanceDest ≤ 14911.835: 1 {0=0, 1=3}
```

Figure 15: centralitySender and centralityReceiver example on decision tree. Source: the author.

3. Is there a relationship between node transitivity and fraud?

No. There is no apparent relationship between the transitivity of the origin accounts or the destiny account and the fraudulent transactions. This parameter is not used as attribute by the model. The fact that transitivity (both the origin and sender account) attribute of the Logistic Regression has a coefficient of zero, corroborates that there seems to be no apparent link between these attribute and fraud

4. Is there a relationship between the amount of the transaction and fraud?

Yes. On the Random Forest model decisions trees, amount seems to be the most used attribute, as seen on the decision tree below on Figure 16. On the Logistic Regression the attribute amount as a coefficient of  $-9.52E-7$ , a rather low value. Nevertheless, there can be specific intervals of amount more linked to fraudulent transactions.





Is there a relationship between the account balance before and after the transaction and fraud?

7.

There is some evidence that the balances may be a good indicator for the evaluation of the transaction. The newBalanceDest, newBalanceOrig, the oldBalanceOrig and the oldBalanceDest frequently appear as attribute on the decision trees. An example of how these parameters are used as attributes by the decision trees is presented below, on Figure 17. On the Logistic Regression the coefficients obtained for oldBalanceOrig and oldBalanceDest are positive, which means the higher the values of the old origin and destiny balances the higher the chances of the transaction to be fraudulent. For newBalanceOrig, newBalanceDest the coefficients are negative which means the higher these balances, fewer the probability for the transaction to be fraudulent. Both for the new and old balance the influence of the origin balance is higher.

## Tree

```
oldBalanceOrig > 149.500
| newBalanceDest > 32685259.830: 0 {0=3, 1=0}
| newBalanceDest ≤ 32685259.830
| | newBalanceOrig > 547.105
| | | centralitySender > 0.118
| | | | centralitySender > 0.292: 0 {0=11, 1=0}
| | | | centralitySender ≤ 0.292
| | | | | oldBalanceOrig > 23015021.165: 1 {0=0, 1=3}
| | | | | oldBalanceOrig ≤ 23015021.165: 0 {0=36, 1=4}
| | | | centralitySender ≤ 0.118: 0 {0=230, 1=0}
| | | newBalanceOrig ≤ 547.105
| | | | newBalanceDest > 16445.180
| | | | | oldBalanceDest > 12589084.160: 1 {0=0, 1=9}
| | | | | oldBalanceDest ≤ 12589084.160
| | | | | | oldBalanceOrig > 72821.900: 1 {0=15, 1=494}
| | | | | | oldBalanceOrig ≤ 72821.900: 0 {0=159, 1=77}
| | | | | newBalanceDest ≤ 16445.180
| | | | | | oldBalanceOrig > 21583.500: 1 {0=0, 1=168}
| | | | | | oldBalanceOrig ≤ 21583.500
| | | | | | | oldBalanceOrig > 20389.500: 0 {0=4, 1=0}
| | | | | | | oldBalanceOrig ≤ 20389.500: 1 {0=0, 1=26}
oldBalanceOrig ≤ 149.500
| step > 8.500
| | amount > 1669823.690
| | | amount > 2017206.950: 0 {0=5, 1=0}
| | | amount ≤ 2017206.950: 1 {0=0, 1=3}
| | | amount ≤ 1669823.690
| | | | oldBalanceDest > 44488.435
| | | | | oldBalanceDest > 2088481.430
| | | | | | oldBalanceDest > 2130596.460: 0 {0=97, 1=0}
| | | | | | oldBalanceDest ≤ 2130596.460: 1 {0=0, 1=3}
| | | | | | oldBalanceDest ≤ 2088481.430: 0 {0=168, 1=0}
| | | | | | oldBalanceDest ≤ 44488.435: 1 {0=0, 1=3}
| | step ≤ 8.500
| | | amount > 585228.415: 1 {0=0, 1=2}
| | | amount ≤ 585228.415: 0 {0=2, 1=0}
```

Figure 17: newBalanceDest, newBalanceOrig, oldBalanceOrig and oldBalanceDest example on decision tree. Source: the author.

## 6. Conclusions and Future Work

In recent years' fraud losses intensified, at the same time, the tolerance to this matter decreased and the transparency and the rules applied in organizations worldwide are more critical because companies are aware that fraud hold back their competitive skills on a global scale and ignore the presence of fraudulent acts turned too costly. (PwC, 2022)

The main challenge of this study relied difficulty that is to have access to real financial transactions data. The data used is a synthetic dataset which can contain information not entirely representative of the reality. In Paysim there is no fraudulent activity in what regards to payments, debits nor cash-in transaction. Additionally, there is no information about the Merchants accounts. The fraudulent activity is only focused on cash-out transactions or transferences. When there is a transference immediately followed by a cash-out transaction, fraudulent activity takes place. When the amount of the transaction is equal to the old balance of the origin, the transaction is fraudulent.

Regarding the association of 3 and 4 node subgraphs and fraud, it was concluded that no connection was found. G-tries was used to scan Paysim network for subgraphs and there was no link between the 3 and 4 subgraph nodes found and the fraudulent transactions for this particular dataset.

It was found that within the Paysim dataset there is a total of 456187 directly connected networks. These networks have 3 to 121 nodes (or accounts). Most of these networks have 3 nodes with more than half of them having less than 10 nodes.

A dataset including original Paysim data and calculated network measures such as the accounts transitivity (or clustering coefficient) and degree centrality was generated. Using this dataset, a Random Forest model was created to classify the transactions and fraudulent or non-fraudulent.

Analyzing the Random Forest and the Logistic Regression models it was possible to conclude that the best attributes parameters were the account degree centrality, the amount being transferred, the step (or time stamp) and the account balanced before and after the transaction, both for the origin and the destiny account. On the other hand, and taken into

account these particular dataset, the account transitivity and the transaction type don't appear to be relevant in classifying the transaction as fraudulent or non-fraudulent.

The overall accuracy of the model is 93.99%. This model has a class recall for fraudulent transactions of 98.03% and class precision of 1.56%. In real life this means a fraudulent red flag would be given to the major part of the fraudulent transactions but a lot of non-fraudulent transactions would also be red flagged in the process which is not necessarily bad since these non-fraudulent transactions can be discarded after an individual investigation is carried out by the bank.

For future work it is suggested to perform an analysis between the existence of fraud and network motifs using real data, as well as including additional network measures on fraud classifications problems.

## 7. Bibliography

- (10 de outubro de 2021). Obtido de Dicionário Priberam da Língua Portuguesa: <https://dicionario.priberam.org/fraude>
- ACFE, A. o. (2012). *Report to the Nations On Occupational Fraude and Abuse, Global Fraud Study*.
- ACFE, A. o. (2020). *Report to the Nations On Occupational Fraud and Abuse, Global Fraud Study*.
- Adewumi, A., & A. A. Akinyelu, (. “.-l. (2017). A survey of machine-learning and nature inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*, pp. 937-953.
- Akoglu, L., McGlohon, M., & Faloutsos, C. (2010). OddBall: Spotting anomalies in weighted graphs. *Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.
- Almeida, M. (2009). *Classification for Fraud Detection with Social Network Analysis*. Instituto Superior Técnico- Universidade Técnica de Lisboa.
- Belyadi, H., & Haghghat, A. (2021). *achine Learning Guide for Oil and Gas Using Python*. ulf Professional Publishing.
- Brito, J., Campos, P., & Leite, R. (2018). An Agent-Based Model for Fraud Detection in Economic Networks. *Highlight of Practical Applications of Agents, Multi-Agent Systems and Complexity. The PAAMS Collections*, pp. 105-115.
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: a Survey. *ACM Computing Surveys*, pp1-58.
- Chawla, N. V. (2005). Data Mining for Imbalanced Datasets: An Overview. Em O. Maimon, & L. Rokach, *Data Mining and Knowledge Discovery Handbook* (pp. 853-867). Boston, MA: Springer US.
- Cichosz, P. (2015). *Data Mining Algorithms: Explained Using R*. Poland: John Wiley & Sons, Ltd.

- Domínguez-Almendros, S., Benítez-Parejo, N., & Gonzalez-Ramirez, A. R. (2011). Logistic Regression Models. *Allergologia et Immunopathologia*, pp. 295-305.
- Ferreira, M. I. (2020). *Fraud Detection and Prevention using Network Mining*. Faculdade de Economia da Universidade do Porto.
- Fontes, X., Aparício, D., Silva, M., Malveiro, B., Ascensão, J., & Bizarro, P. (2021). Finding NeMo: Fishing in banking networks using network motifs.
- Francis, C., Pepper, N., & Strong, H. (2011). Using Support Vector Machines to Detect Medical Fraud and Abuse. *Proceedings of Conference of the IEEE Engineering in Medicine and Biology Society*, 8291-8294.
- Garner, B. A. (2019). *Black's Law Dictionary, 11th Edition*. Thomson Reuters.
- Gee, S. (2015). *Fraud and Fraud Detection: A Data Analytics Approach*. Wiley.
- Han, J. (2006). *Data Mining: Concepts and Techniques*. Illinois: Elsevier.
- Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression*. Wiley Series in Probability and Statistics.
- Jeh, G., & Widom, J. (January 2002). SimRank: A Measure of Structural-Context Similarity. *The Eighth ACM SIGKDD International Conference*.
- Kirkos, E., Spathis, C., & Manolopoulos, Y. (2007). Data mining techniques for the detection of fraudulent financial statements. *Expert Systems with Applications* , 32-23.
- Kosse, A., & Szemere, R. (December de 2021). Covid-19 accelerated the digitalisation of payments. *Commentary on the BIS Committee on Payments and Market Infrastructures (CPMI)*.
- Lavion, D. (2020). *Pulling Fraud Out of the Shadows: PwC's Global Economic Crime and Fraud Survey 2018*. Obtido de <https://www.pwc.com/gx/en/forensics/global-economic-crime-and-fraud-survey-2018.pdf> access on the 7th of january of 2020
- Leite, R., Gschwandtner, T. ., Gstrein, E., & Kuntner, J. (2018). Network Analysis for Financial Fraud Detection. *EUROVIS*.
- Lin, J. W., Hwang, M. I., & Becker, J. D. (2003). A fuzzy neural network for assessing the risk of fraudulent financial reporting vol. *Managerial Auditing Journal vol. 18*, 657-665.

- Lopez-Rojas, E. A., Elmir, A., & Axelsson, S. (2016). PAYSIM: A Financial Mobile Money Simulator for Fraud Detection. *Blekinge Institute of Technology, The Norwegian University of Science and Technology*.
- Malekian, D., & Hashemi, M. R. (2013). An Adaptive Profile Based Fraud Detection Framework For Handling Concept Drift. *10th International ISC Conference on Information Security and Cryptology (ISCISC)*.
- Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002). Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 824-7.
- Nick, F., Krause, P., & Garnc, W. (2018). How Artificial Intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 130 – 157. Obtido de pp. 130 – 157.
- Noble, C., & Cook, D. (2003). Graph-based anomaly detection” in Proceedings of ACM. *SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Oana, F. I. (2018). Economic Fraud in International Business: Forms and Implications. *“Ovidius” University Annals, Economic Sciences Series*, Vol. 18.
- Pimenta, C. (2009). *Esboço de Quantificação da Fraude em Portugal*. Working Papers - OBEGEF.
- PwC. (2022). Protecting the perimeter: The rise of external fraud. *Technical report, PwC's Global Economic Crime and Fraud Survey 2022*.
- R. Ferreira, M. I. (2020). *FRAUD DETECTION AND PREVENTION USING NETWORK MINING*. Porto: University of Porto- Faculty of Economy.
- Ribeiro, P. (2021). *Network Science - Measuring Networks and Random Graph Models*. Universidade do Porto - Faculdade de Ciências .
- Ribeiro, P., & Silva, F. (2010). G-Tries: an efficient data structure for discovering network motifs. *ACM 25th Symposium On Applied Computing - Bioinformatics Track*, (pp. pp 1559-1566). Sierre, Switzerland.
- Ribeiro, P., & Silva, F. (2014). G-Tries: a data structure for storing and finding subgraphs. *Data Mining and Knowledge Discovery*, 28, pp 337–377.



- Ribeiro, P., Paredes, P., Silva, M. E., Aparício, D., & Silva, F. (2021). A Survey on Subgraph Counting: Concepts, Algorithms and Applications to Network Motifs and Graphlets. *ACM Computing Surveys*, pp 1-36.
- Ribeiro, P., Silva, F., & Kaiser, M. (2009). Strategies for Network Motifs Discovery. *Fifth IEEE International Conference on e-Science*.
- Sherly, K., & Nedunchezian, R. (2010). Boat adaptive credit card fraud detection system. *IEEE International Conference on Computational Intelligence and Computing Research*.
- Simmons, M. R. (1995). *Recognizing the Elements of Fraud*.
- Tsai, Y. H., Ko, C. H., & Lin, K. C. (2014). Using CommonKADS method to build prototype system in medical insurance fraud detection. *Journal of Networks*.
- Wilcox, R. R. (2019). Logistic Regression: An Inferential Method for Identifying the Best Predictors. *Journal of Modern Applied Statistical Methods*.
- Witten, I. H., Frank, E., & Hall, H. A. (2011). *Data Mining Practical Machine Learning Tools and Techniques*. United States: Elsevier.
- Xu, X., Yuruk, N., Feng, Z., & Schweiger, T. (2007). SCAN: A Structural Clustering Algorithm for Networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Yan, X., & Han, J. (2002). gSpan: Graph-Based Substructure Pattern Mining. *Proceedings - IEEE International Conference on Data Mining*, pp 721-724.
- Yue, D., Wu, X., Wang, Y., LI, Y., & Chu, C.-H. (2007). A Review of Data Mining-based Financial Fraud. *Hebei University of Technology, Pennsylvania State University*.
- Zanin, M. Z., Romance, M., Moral, S., & Criado, R. (2018). Credit Card Fraud Detection through Parenclitic Network Analysis. *Hindawi - Complexity*.

## 8. Attachments

### 8.1. Paysim Data Analysis

```
# Load Pkgs
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline

#
import warnings

df = pd.read_csv("Paysim.csv")

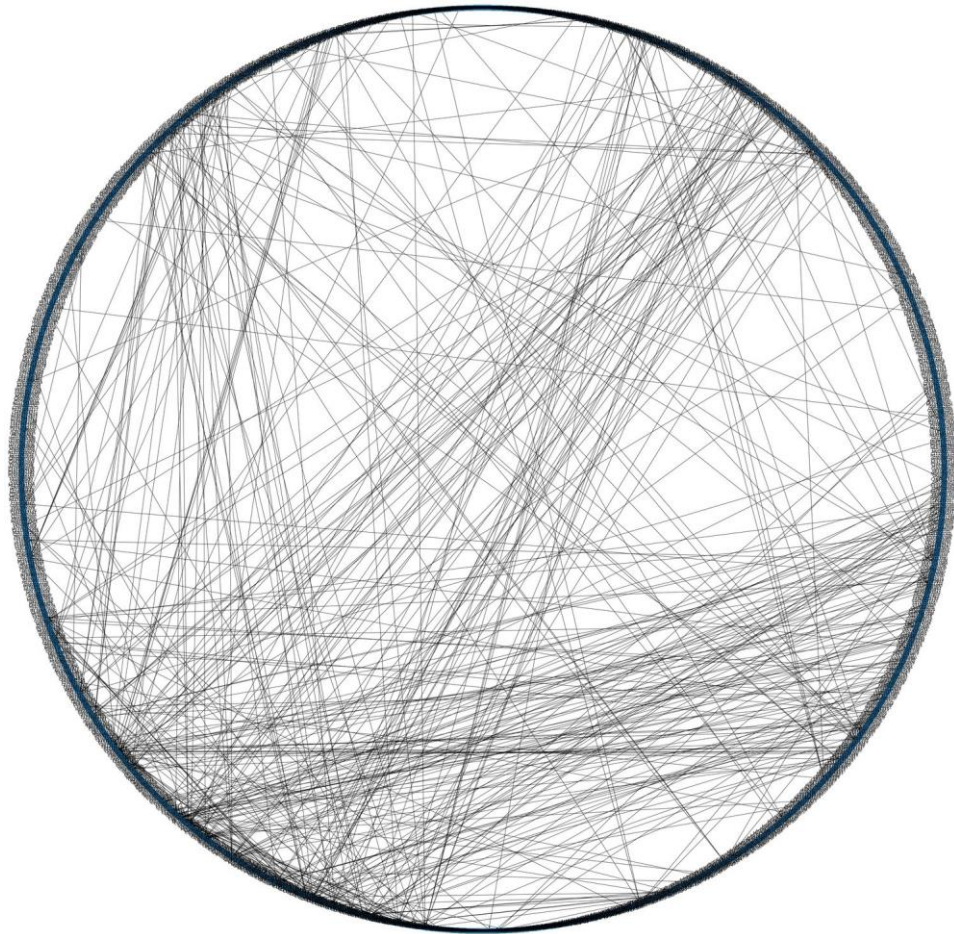
Paysim_graph = nx.MultiGraph()

# Read Our Edgelist
Paysim_graph =
nx.from_pandas_edgelist(df, source="nameOrig", target="nameDest", edge_attr="amount", create_using=nx.DiGraph())

plt.figure(figsize=(50,50))

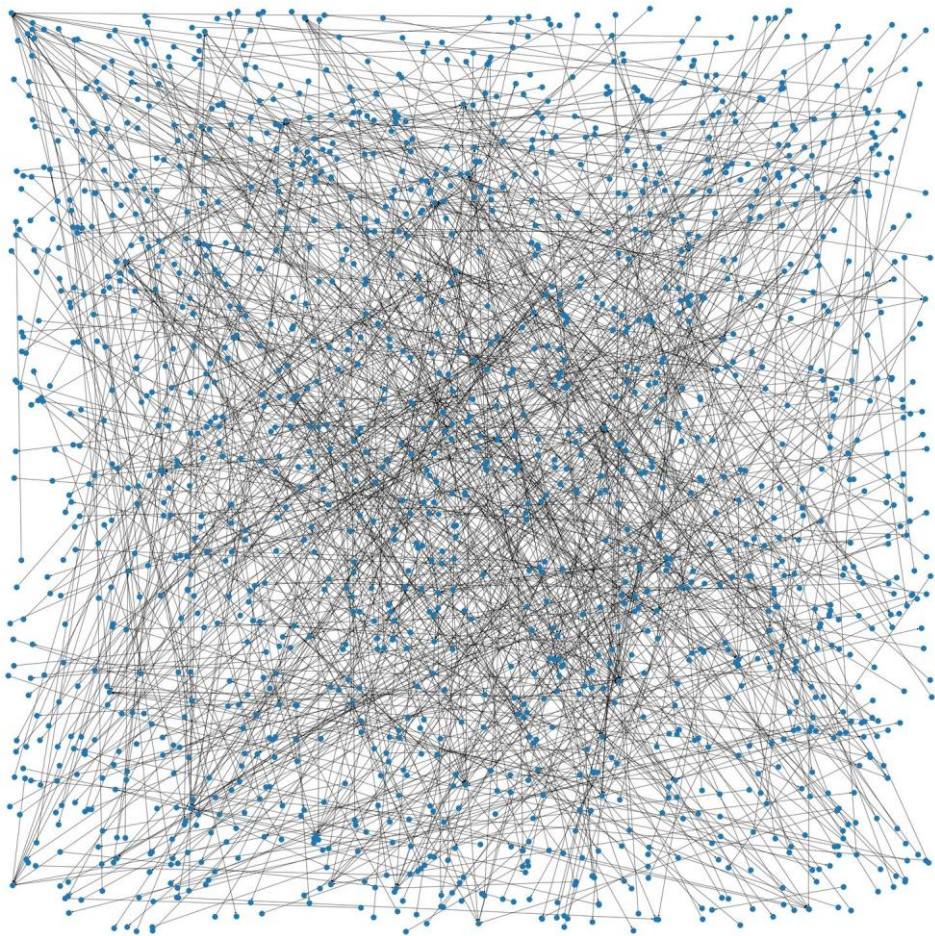
nx.draw_shell(Paysim_graph, with_labels=True)

plt.show()
```



```
pos = nx.random_layout(Paysim_graph)
plt.figure(figsize=(50,50))
```

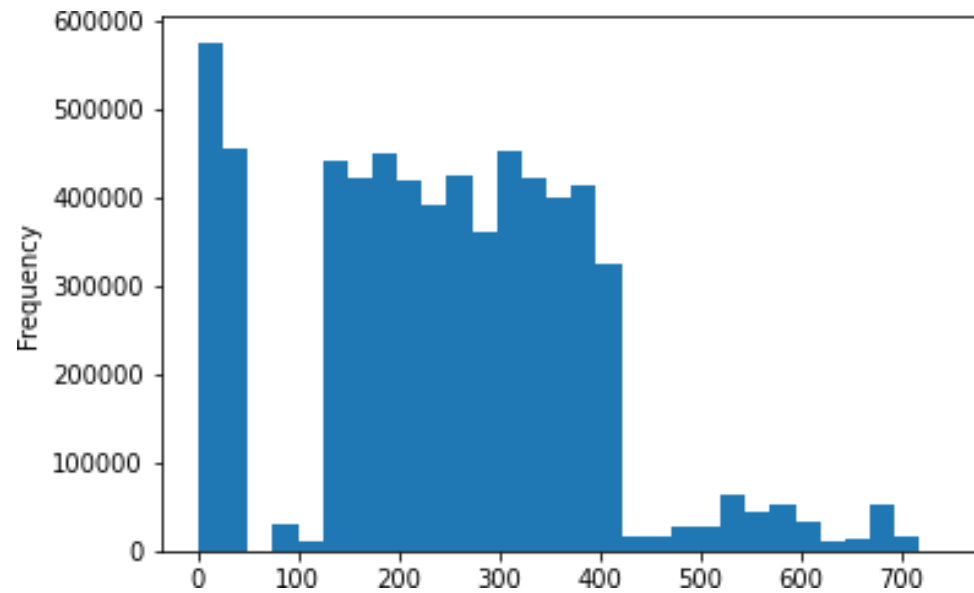
```
nx.draw(Paysim_graph, pos)  
plt.show()
```



```
df["step"].describe()
```

```
count 6362620  
mean 243  
std 142  
min 1  
25% 156  
50% 239  
75% 335  
max 743  
Name: step, dtype: float64
```

```
df["step"].plot.hist(bins=30, sharex=True, sharey=True, orientation="vertical", cumulative=False);
```



```
df["amount"].describe()
```

```
count 6362620
mean 179862
std 603858
min 0
25% 13390
50% 74872
75% 208721
max 92445517
Name: amount, dtype: float64
```

```
df.groupby(["type"])["amount"].agg(["count", "mean", "std", "min", "max"]),
(
    count    mean    std    min    max
type
```

CASH_IN	1399284	168920	126508	0 1915268
CASH_OUT	2237500	176274	175330	0 10000000
DEBIT	41432	5484	13319	1 569078
PAYMENT	2151495	13058	12556	0 238638
TRANSFER	532909	910647	1879574	3 92445517,)

```
df.groupby("nameOrig") ["nameOrig"].count().sort_values(ascending=False)
```

```
nameOrig C1065307291 3
C1784010646 3
C1902386530 3
C1832548028 3
C545315117 3
```

```
..
C1645325210 1
C1645325020 1
C1645324530 1
C1645324143 1
C999999784 1
```

```
Name: nameOrig, Length: 6353307, dtype: int64
```

```
df.groupby(df.nameOrig.str[:1]) ["nameOrig"].count().sort_values(ascending=False)
```

```
nameOrig
C 6362620
Name: nameOrig, dtype: int64
```

```
df.query("(newbalanceDest-oldbalanceDest != amount) and nameDest <'J' and type != 'DEBIT'")
```

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2	1	TRANSFER	181	C1305486145	181	0	C553264065	0	0	1
3	1	CASH_OUT	181	C840083671	181	0	C38997010	21182	0	1
15	1	CASH_OUT	229134	C905080434	15325	0	C476402209	5083	51513	0
19	1	TRANSFER	215310	C1670993182	705	0	C1100439041	22425	0	0
24	1	TRANSFER	311686	C1984094095	10835	0	C932583850	6267	2719173	0
...	...	...	...	...	...	...	...	...	...	...
6362614	743	TRANSFER	339682	C2013999242	339682	0	C1850423904	0	0	1
6362616	743	TRANSFER	6311409	C1529008245	6311409	0	C1881841831	0	0	1
6362617	743	CASH_OUT	6311409	C1162922333	6311409	0	C1365125890	68489	6379898	1
6362618	743	TRANSFER	850003	C1685995037	850003	0	C2080388513	0	0	1



step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
<b>6362619</b>	743	CASH_OUT	850003	C1280323807	850003	0	C873221189	6510099	7360102	1

```
df.query("(abs(oldbalanceOrg-newbalanceOrig) != amount) and nameDest < 'J' and type == 'TRANSFER'")
```

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
<b>19</b>	1	TRANSFER	215310	C1670993182	705	0	C1100439041	22425	0	0
<b>24</b>	1	TRANSFER	311686	C1984094095	10835	0	C932583850	6267	2719173	0
<b>78</b>	1	TRANSFER	42712	C283039401	10363	0	C1330106945	57902	24044	0
<b>79</b>	1	TRANSFER	77958	C207471778	0	0	C1761291320	94900	22234	0
<b>80</b>	1	TRANSFER	17231	C1243171897	0	0	C783286238	24672	0	0
...	...	...	...	...	...	...	...	...	...	...
<b>6362322</b>	718	TRANSFER	82096	C614459560	13492	0	C855350324	0	82096	0

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6362456	730	TRANSFER	10000000	C1277761503	37316255	27316255	C500987951	0	0	1
6362460	730	TRANSFER	10000000	C2140038573	17316255	17316255	C1395467927	0	0	1
6362462	730	TRANSFER	7316255	C1869569059	17316255	17316255	C1861208726	0	0	1
6362584	741	TRANSFER	5674548	C992223106	5674548	5674548	C1366804249	0	0	1

```
df.query("(abs(oldbalanceOrg-newbalanceOrig) != amount) and nameDest < 'J' and type == 'CASH_OUT')")
```

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
15	1	CASH_OUT	229134	C905080434	15325	0	C476402209	5083	51513	0
42	1	CASH_OUT	110415	C768216420	26845	0	C1509514333	288800	2415	0
47	1	CASH_OUT	56954	C1570470538	1942	0	C824009085	70253	64106	0
48	1	CASH_OUT	5347	C512549200	0	0	C248609774	652637	6453431	0

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
51	1	CASH_OUT	23261	C2072313080	20412	0	C2001112025	25742	0	0
...	...	...	...	...	...	...	...	...	...	...
6362306	718	CASH_OUT	169291	C1569237054	26919	0	C342077848	2684602	2853894	0
6362313	718	CASH_OUT	111964	C1438119383	4514	0	C99772923	154925	266889	0
6362317	718	CASH_OUT	317177	C857156502	170	0	C784108220	345042	662220	0
6362320	718	CASH_OUT	159188	C691808084	3859	0	C1818183087	0	159188	0
6362321	718	CASH_OUT	186274	C102120699	168046	0	C1515639522	24894	211168	0

df.max()

```

step          743
type          TRANSFER
amount       92445517
nameOrig     C999999784
oldbalanceOrg 59585040
newbalanceOrig 49585040
nameDest     M999999784

```

```
oldbalanceDest      356015889
newbalanceDest      356179279
isFraud              1
isFlaggedFraud      1
dtype: object
```

```
df.min()
```

```
step                1
type                CASH_IN
amount              0
nameOrig            C1000000639
oldbalanceOrg       0
newbalanceOrig      0
nameDest            C1000004082
oldbalanceDest      0
newbalanceDest      0
isFraud             0
isFlaggedFraud      0
dtype: object
```

```
df.nameOrig.describe()
```

```
count      6362620
unique      6353307
top         C1902386530
freq        3
Name: nameOrig, dtype: object
```

```
df.oldbalanceOrg.describe()
```

```
count 6362620
mean 833883
std 2888243
min 0
25% 0
50% 14208
75% 107315
max 59585040
Name: oldbalanceOrg, dtype: float64
```

```
df.isFraud.describe()
```

```
count 6362620
mean 0
std 0
min 0
25% 0
50% 0
75% 0
max 1
```

```
Name: isFraud, dtype: float64
```

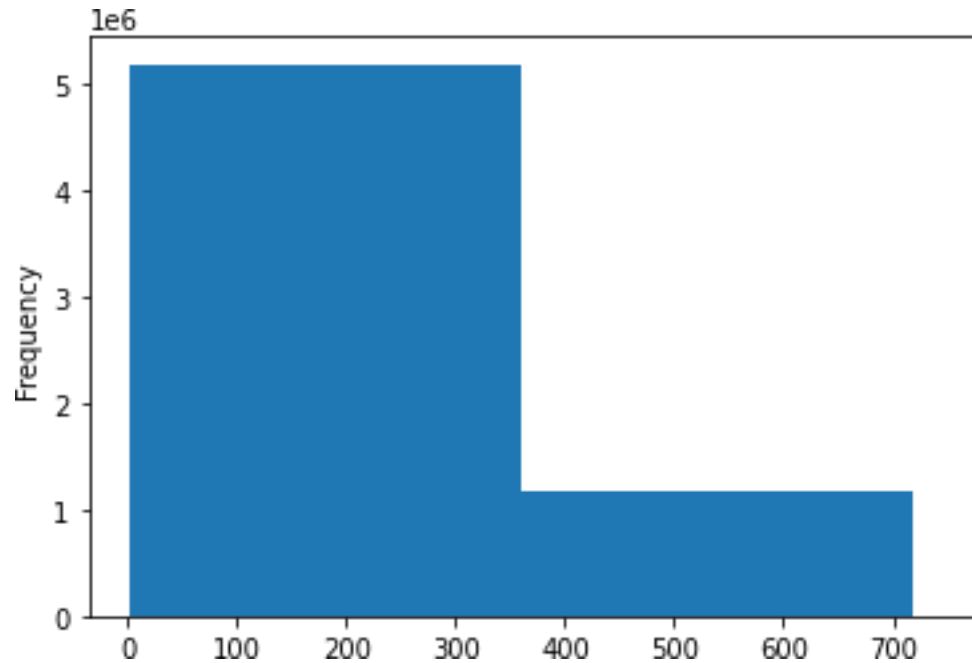
```
df.groupby("isFlaggedFraud") ["amount"].count().sort_values(ascending=False)
```

```
isFraud
0 6354407
1 8213
Name: amount, dtype: int64
```

```
df.groupby("isFlaggedFraud") ["amount"].count().sort_values(ascending=False)
```

```
isFlaggedFraud
0      6362604
1         16
Name: amount, dtype: int64
```

```
df.groupby("isFraud") ["step"].plot.hist(bins=2, sharex=True, sharey=True,
orientation="vertical", cumulative=False);
```



```
df.query("isFlaggedFraud == 1").groupby("isFraud") ["isFraud"].count()
```

```
isFraud
```

```

1      16
Name: isFraud, dtype: int64

df.query("isFraud == 1").to_csv('Paysim_isFraud==1.csv')

dfisFraud = pd.read_csv("Paysim_isFraud==1.csv")

dfisFraud.describe()

```

Unnamed: 0	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud		
<b>count</b>	8213	8213	8213	8213	8213	8213	8213	8213	8213	8213
<b>mean</b>	4244980	368	1467967	1649668	192393	544250	1279708	1	0	
<b>std</b>	2157498	216	2404253	3547719	1965666	3336421	3908817	0	0	
<b>min</b>	2	1	0	0	0	0	0	1	0	
<b>25%</b>	2065130	181	127091	125822	0	0	0	1	0	
<b>50%</b>	5188057	367	441423	438983	0	0	4676	1	0	
<b>75%</b>	6168689	558	1517771	1517771	0	147829	1058725	1	0	

Unnamed: 0	step	amount	oldbalanceOrig	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
max	6362619	743	10000000	59585040	49585040	236230517	236726495	1 1

```
dfisFraud.groupby(["type"])["amount"].agg(["count", "mean", "std", "min", "max"]),
```

```
(
      count      mean      std  min  max  type
CASH_OUT    4116 1455103 2393842  0 10000000
TRANSFER    4097 1480892 2414890 64 10000000,)
```

```
dfisFraud.query("(abs(oldbalanceOrig-newbalanceOrig) == amount) and nameDest > 'J' and type == 'TRANSFER'")
```

Empty DataFrame

Columns: [Unnamed: 0, step, type, amount, nameOrig, oldbalanceOrig, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, isFlaggedFraud]

Index: []

```
dfisFraud.query("(abs(oldbalanceOrig-newbalanceOrig) != amount) and nameDest < 'J' and type == 'CASH_OUT'")
```

Unnamed: 0	step	type	amount	nameOrig	oldbalanceOrig	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6	724	1	CASH_OUT	416001	C749981943	0	0	C667346055	102 9291620	1	



<b>Unnamed: 0</b>	<b>step</b>	<b>type</b>	<b>amount</b>	<b>nameOrig</b>	<b>oldbalanceOrg</b>	<b>newbalanceOrig</b>	<b>nameDest</b>	<b>oldbalanceDest</b>	<b>newbalanceDest</b>	<b>isFraud</b>	<b>isFlaggedFraud</b>
<b>13</b>	1911	1	CASH_OUT	132843	C13692003	4499	0	C297927961	0	132843	1
<b>78</b>	14861	8	CASH_OUT	181728	C2102265902	0	0	C789014007	11397	184478	1
<b>111</b>	77745	10	CASH_OUT	277971	C489647033	0	0	C571514738	0	277971	1
<b>135</b>	169998	12	CASH_OUT	149669	C227115333	0	0	C460735540	44170	193839	1
<b>138</b>	178668	12	CASH_OUT	222049	C265790428	0	0	C1700442291	2979	225028	1
<b>147</b>	200845	13	CASH_OUT	454859	C1274887619	0	0	C2146670328	0	454859	1
<b>179</b>	291459	15	CASH_OUT	95428	C947728507	0	0	C1720721903	0	95428	1
<b>180</b>	296686	15	CASH_OUT	39713	C1404885898	0	0	C1795377601	1274867	1314580	1
<b>203</b>	377151	17	CASH_OUT	42063	C897869440	340830	298768	C616721459	398931	678420	1
<b>208</b>	408955	18	CASH_OUT	314252	C1462280812	75956	0	C90486891	7962205	8276457	1

<b>Unnamed: 0</b>	<b>step</b>	<b>type</b>	<b>amount</b>	<b>nameOrig</b>	<b>oldbalanceOrg</b>	<b>newbalanceOrig</b>	<b>nameDest</b>	<b>oldbalanceDest</b>	<b>newbalanceDest</b>	<b>isFraud</b>	<b>isFlaggedFraud</b>
<b>217</b>	424928	18	CASH_OUT	508782	C576782065	0	0	C2090737806	1082008	1590790	1
<b>224</b>	479636	19	CASH_OUT	122102	C149845822	0	0	C1200316948	0	639940	1
<b>239</b>	543928	21	CASH_OUT	23292	C1861214292	0	0	C1834461593	392365	415657	1
<b>256</b>	559979	22	CASH_OUT	89571	C1022920965	4506	0	C1460548505	1929428	2018999	1
<b>389</b>	643671	35	CASH_OUT	112281	C609821524	0	0	C708118422	40512	152793	1
<b>414</b>	694551	36	CASH_OUT	234377	C1737133410	0	0	C877378703	34938	269315	1
<b>429</b>	732891	37	CASH_OUT	112486	C124687089	0	0	C179706450	257274	369761	1
<b>434</b>	750755	38	CASH_OUT	577419	C1907944035	0	0	C541373010	0	577419	1
<b>443</b>	764187	38	CASH_OUT	407006	C1948189565	0	0	C1059308371	0	407006	1
<b>2050</b>	2058343	181	CASH_OUT	332730	C645124798	0	0	C1797918851	613712	946442	1

Unnamed: 0	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2117	2242699	186	CASH_OUT	229910	C260366436	0	0	C1620159110	0	229910	1
2330	2622102	208	CASH_OUT	291520	C485214392	0	0	C1138105020	0	291520	1
2585	2946481	230	CASH_OUT	40611	C1617773163	0	0	C478307499	0	40611	1
2608	2983493	231	CASH_OUT	94373	C1451318490	0	0	C421443093	471783	566156	1

dfisFraud.query("nameDest < 'J'")

Unnamed: 0	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6	724	1	CASH_OUT	416001	C749981943	0	0	C667346055	102	9291620	1
13	1911	1	CASH_OUT	132843	C13692003	4499	0	C297927961	0	132843	1
78	14861	8	CASH_OUT	181728	C2102265902	0	0	C789014007	11397	184478	1
111	77745	10	CASH_OUT	277971	C489647033	0	0	C571514738	0	277971	1

Unnamed: 0	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
<b>135</b>	169998	12	CASH_OUT	149669	C227115333	0	0	C460735540	44170	193839	1
<b>138</b>	178668	12	CASH_OUT	222049	C265790428	0	0	C1700442291	2979	225028	1
<b>147</b>	200845	13	CASH_OUT	454859	C1274887619	0	0	C2146670328	0	454859	1
<b>179</b>	291459	15	CASH_OUT	95428	C947728507	0	0	C1720721903	0	95428	1
<b>180</b>	296686	15	CASH_OUT	39713	C1404885898	0	0	C1795377601	1274867	1314580	1
<b>203</b>	377151	17	CASH_OUT	42063	C897869440	340830	298768	C616721459	398931	678420	1
<b>208</b>	408955	18	CASH_OUT	314252	C1462280812	75956	0	C90486891	7962205	8276457	1
<b>217</b>	424928	18	CASH_OUT	508782	C576782065	0	0	C2090737806	1082008	1590790	1
<b>224</b>	479636	19	CASH_OUT	122102	C149845822	0	0	C1200316948	0	639940	1
<b>239</b>	543928	21	CASH_OUT	23292	C1861214292	0	0	C1834461593	392365	415657	1

<b>Unnamed: 0</b>	<b>step</b>	<b>type</b>	<b>amount</b>	<b>nameOrig</b>	<b>oldbalanceOrg</b>	<b>newbalanceOrig</b>	<b>nameDest</b>	<b>oldbalanceDest</b>	<b>newbalanceDest</b>	<b>isFraud</b>	<b>isFlaggedFraud</b>
<b>256</b>	559979	22	CASH_OUT	89571	C1022920965	4506	0	C1460548505	1929428	2018999	1
<b>389</b>	643671	35	CASH_OUT	112281	C609821524	0	0	C708118422	40512	152793	1
<b>414</b>	694551	36	CASH_OUT	234377	C1737133410	0	0	C877378703	34938	269315	1
<b>429</b>	732891	37	CASH_OUT	112486	C124687089	0	0	C179706450	257274	369761	1
<b>434</b>	750755	38	CASH_OUT	577419	C1907944035	0	0	C541373010	0	577419	1
<b>443</b>	764187	38	CASH_OUT	407006	C1948189565	0	0	C1059308371	0	407006	1
<b>2050</b>	2058343	181	CASH_OUT	332730	C645124798	0	0	C1797918851	613712	946442	1
<b>2117</b>	2242699	186	CASH_OUT	229910	C260366436	0	0	C1620159110	0	229910	1
<b>2330</b>	2622102	208	CASH_OUT	291520	C485214392	0	0	C1138105020	0	291520	1
<b>2585</b>	2946481	230	CASH_OUT	40611	C1617773163	0	0	C478307499	0	40611	1

Unnamed: 0	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2608	2983493	231	CASH_OUT	94373	C1451318490	0	0	C421443093	471783	566156	1

```
df.query("isFraud == 1 and nameDest > 'J'")Empty DataFrame
Columns: [step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, isFlaggedFraud] Index: []
```

```
df.query("isFraud == 1 and nameDest < 'J'")
```

Unnamed: 0	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
6	724	1	CASH_OUT	416001	C749981943	0	0	C667346055	102 9291620	1	1
13	1911	1	CASH_OUT	132843	C13692003	4499	0	C297927961	0 132843	1	1
78	14861	8	CASH_OUT	181728	C2102265902	0	0	C789014007	11397 184478	1	1
111	77745	10	CASH_OUT	277971	C489647033	0	0	C571514738	0 277971	1	1
135	169998	12	CASH_OUT	149669	C227115333	0	0	C460735540	44170 193839	1	1

<b>Unnamed: 0</b>	<b>step</b>	<b>type</b>	<b>amount</b>	<b>nameOrig</b>	<b>oldbalanceOrg</b>	<b>newbalanceOrig</b>	<b>nameDest</b>	<b>oldbalanceDest</b>	<b>newbalanceDest</b>	<b>isFraud</b>	<b>isFlaggedFraud</b>
<b>138</b>	178668	12	CASH_OUT	222049	C265790428	0	0	C1700442291	2979	225028	1
<b>147</b>	200845	13	CASH_OUT	454859	C1274887619	0	0	C2146670328	0	454859	1
<b>179</b>	291459	15	CASH_OUT	95428	C947728507	0	0	C1720721903	0	95428	1
<b>180</b>	296686	15	CASH_OUT	39713	C1404885898	0	0	C1795377601	1274867	1314580	1
<b>203</b>	377151	17	CASH_OUT	42063	C897869440	340830	298768	C616721459	398931	678420	1
<b>208</b>	408955	18	CASH_OUT	314252	C1462280812	75956	0	C90486891	7962205	8276457	1
<b>217</b>	424928	18	CASH_OUT	508782	C576782065	0	0	C2090737806	1082008	1590790	1
<b>224</b>	479636	19	CASH_OUT	122102	C149845822	0	0	C1200316948	0	639940	1
<b>239</b>	543928	21	CASH_OUT	23292	C1861214292	0	0	C1834461593	392365	415657	1
<b>256</b>	559979	22	CASH_OUT	89571	C1022920965	4506	0	C1460548505	1929428	2018999	1

<b>Unnamed: 0</b>	<b>step</b>	<b>type</b>	<b>amount</b>	<b>nameOrig</b>	<b>oldbalanceOrg</b>	<b>newbalanceOrig</b>	<b>nameDest</b>	<b>oldbalanceDest</b>	<b>newbalanceDest</b>	<b>isFraud</b>	<b>isFlaggedFraud</b>
<b>389</b>	643671	35	CASH_OUT	112281	C609821524	0	0	C708118422	40512	152793	1
<b>414</b>	694551	36	CASH_OUT	234377	C1737133410	0	0	C877378703	34938	269315	1
<b>429</b>	732891	37	CASH_OUT	112486	C124687089	0	0	C179706450	257274	369761	1
<b>434</b>	750755	38	CASH_OUT	577419	C1907944035	0	0	C541373010	0	577419	1
<b>443</b>	764187	38	CASH_OUT	407006	C1948189565	0	0	C1059308371	0	407006	1
<b>2050</b>	2058343	181	CASH_OUT	332730	C645124798	0	0	C1797918851	613712	946442	1
<b>2117</b>	2242699	186	CASH_OUT	229910	C260366436	0	0	C1620159110	0	229910	1
<b>2330</b>	2622102	208	CASH_OUT	291520	C485214392	0	0	C1138105020	0	291520	1
<b>2585</b>	2946481	230	CASH_OUT	40611	C1617773163	0	0	C478307499	0	40611	1
<b>2608</b>	2983493	231	CASH_OUT	94373	C1451318490	0	0	C421443093	471783	566156	1



```
df.query("isFlaggedFraud == 1 and nameDest < 'J'")
```

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	CASH_OUT	416001	C749981943	0	0	C667346055	102	9291620	1	
1	CASH_OUT	132843	C13692003	4499	0	C297927961	0	132843	1	
8	CASH_OUT	181728	C2102265902	0	0	C789014007	11397	184478	1	
10	CASH_OUT	277971	C489647033	0	0	C571514738	0	277971	1	
12	CASH_OUT	149669	C227115333	0	0	C460735540	44170	193839	1	
12	CASH_OUT	222049	C265790428	0	0	C1700442291	2979	225028	1	
13	CASH_OUT	454859	C1274887619	0	0	C2146670328	0	454859	1	
15	CASH_OUT	95428	C947728507	0	0	C1720721903	0	95428	1	
15	CASH_OUT	39713	C1404885898	0	0	C1795377601	1274867	1314580	1	

<b>step</b>	<b>type</b>	<b>amount</b>	<b>nameOrig</b>	<b>oldbalanceOrg</b>	<b>newbalanceOrig</b>	<b>nameDest</b>	<b>oldbalanceDest</b>	<b>newbalanceDest</b>	<b>isFraud</b>	<b>isFlaggedFraud</b>
17	CASH_OUT	42063	C897869440	340830	298768	C616721459	398931	678420	1	
18	CASH_OUT	314252	C1462280812	75956	0	C90486891	7962205	8276457	1	
18	CASH_OUT	508782	C576782065	0	0	C2090737806	1082008	1590790	1	
19	CASH_OUT	122102	C149845822	0	0	C1200316948	0	639940	1	
21	CASH_OUT	23292	C1861214292	0	0	C1834461593	392365	415657	1	
22	CASH_OUT	89571	C1022920965	4506	0	C1460548505	1929428	2018999	1	
35	CASH_OUT	112281	C609821524	0	0	C708118422	40512	152793	1	
36	CASH_OUT	234377	C1737133410	0	0	C877378703	34938	269315	1	
37	CASH_OUT	112486	C124687089	0	0	C179706450	257274	369761	1	
38	CASH_OUT	577419	C1907944035	0	0	C541373010	0	577419	1	

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
38	CASH_OUT	407006	C1948189565	0	0	C1059308371	0	407006	1	
181	CASH_OUT	332730	C645124798	0	0	C1797918851	613712	946442	1	
186	CASH_OUT	229910	C260366436	0	0	C1620159110	0	229910	1	
208	CASH_OUT	291520	C485214392	0	0	C1138105020	0	291520	1	
230	CASH_OUT	40611	C1617773163	0	0	C478307499	0	40611	1	
231	CASH_OUT	94373	C1451318490	0	0	C421443093	471783	566156	1	

```
df.query("nameDest > 'J'")
```

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9840	C1231006815	170136	160296	M1979787155	0	0	0

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	1	PAYMENT	1864	C1666544295	21249	19385	M2044282225	0	0	0
4	1	PAYMENT	11668	C2048537720	41554	29886	M1230701703	0	0	0
5	1	PAYMENT	7818	C90045638	53860	46042	M573487274	0	0	0
6	1	PAYMENT	7108	C154988899	183195	176087	M408069119	0	0	0
...	...	...	...	...	...	...	...	...	...	...
6362312	718	PAYMENT	8178	C1213413071	11742	3564	M1112540487	0	0	0
6362314	718	PAYMENT	17841	C1045048098	10182	0	M1878955882	0	0	0
6362316	718	PAYMENT	1023	C1203084509	12	0	M675916850	0	0	0
6362318	718	PAYMENT	4110	C673558958	5521	1411	M1126011651	0	0	0
6362319	718	PAYMENT	8634	C642813806	518802	510168	M747723689	0	0	0

## 8.2. Network and Dataset creation

```
# Load Pkgs
import pandas as pd
import networkx as nx
#import matplotlib.pyplot as plt
import pickle
##matplotlib inline
import warnings
import subprocess
import os
import time

#Creation of network object with the respective attributes

class Network:
def _____init__(self, nodes, edges, motifs,
hasFraud):
self.nodes = nodes
self.edges = edges
self.motifs = motifs
self.hasFraud = hasFraud

class Node:
def _____init__(self, name, fraudIn,
fraudOut):
self.name = name
self.fraudIn = fraudIn
self.fraudOut = fraudOut

class Edge:
def _____init__(self, sender,
receiver):
self.sender = sender
self.receiver = receiver

class Motif:
def _____init__(self, adjmatrix, occ_original, z_score, avg_random,
stdev_random):
self.adjmatrix = adjmatrix
self.occ_original = occ_original
self.z_score = z_score
self.avg_random = avg_random
self.stdev_random = stdev_random
```

*#Creation of objects from the file with the subnetworks including  
gtries 4 nodes subgraphs information*

```
import csv
```

```
networks = list()
```

```
file = open('e:/joana/tese/Paysim4nodes.txt', 'r')  
lines = file.readlines()
```

```
nodesList = list()  
edgesList = list()  
motifsList = list()
```

```
for line in lines:  
line = line.replace("\n", "").replace(", ", "")  
row=line.split(" ")
```

```
if row == ['*']:  
    if len(nodesList) > 0 and len(edgesList) > 0:  
        newNetwork = Network(nodesList, edgesList, motifsList,  
                               hasFraud)  
        networks.append(newNetwork)  
        nodesList = list()  
        edgesList = list()  
        motifsList = list()  
        lastNode = 0  
        lastEdge = 0  
        c = -1  
        continue
```

```
if c == 0:  
    lastNode = int(row[0])  
    lastEdge = int(row[1]) + lastNode  
    hasFraud = int(row[2])  
    c = c + 1;  
    continue;
```

```
if c > lastEdge + 2:  
    if "Unable" not in line:  
        adjmatrix = row[0]  
        occ_original = row[1]  
        z_score = row[2]  
        avg_random = row[3]  
        stdev_random = row[4]  
  
        newMotif=Motif(adjmatrix, occ_original, z_score,avg_random,  
                        stdev_random)  
        motifsList.append(newMotif)
```

```
if c > 0 and c <= lastNode:  
    nodesList.append(Node(row[1], row[2], row[3]))
```

```

if c > lastNode and c <= lastEdge:
    edgesList.append(Edge(row[0], row[1]))
c = c + 1

#number of frequency of subgraphs in all networks/in fraudulent networks

motifsPresent = set()

for network in networks:
    for motif in network.motifs:
        motifsPresent.add(motif.adjmatrix)

for motifMatrix in motifsPresent:
    count = 0;
    fraudCount = 0;
    for network in networks:
        for motif in network.motifs:
            if motif.adjmatrix ==motifMatrix:
                count = count + 1
                if network.hasFraud == 1:
                    fraudCount = fraudCount + 1
    print(motifMatrix + ": " + str(count) + " " + str(fraudCount))

"0111101011001000": 2 1
"0111100010001000": 321404 4249
"0110100110000100": 2320 50

#creating new column with nameOrig_nameDest of each transaction

df = pd.read_csv('E:/joana/tese/Paysim.csv')
df = df.assign(Orig_Dest=lambda x: x.nameOrig + "_" + x.nameDest)

# number of networks with subgraph 0110100110000100 and fraud

filteredNetworks = set()

for network in networks:
    for motif in network.motifs:
        if motif.adjmatrix == '"0110100110000100"' and network.hasFraud ==
1:
            filteredNetworks.add(network)
            break

print(len(filteredNetworks))

50

```

```
#creation of list with Paysim information of the transactions from the  
networks file
```

```
transactions = list()
```

```
for network in networks:
```

```
    networknodes=network.nodes
```

```
    networkTrasactions=network.edges
```

```
    for transaction in networkTrasactions:
```

```
        transactions.append(networknodes[int(transaction.sender) -  
1].name + "_" + networknodes[int(transaction.receiver) -  
1].name)
```

```
#Creation of objects from the file with the subnetworks including  
gtries 3 nodes subgraphs information
```

```
import csv
```

```
networks = list()
```

```
file = open('e:/joana/tese/Paysim3nodes.txt', 'r')
```

```
lines = file.readlines()
```

```
nodesList = list()
```

```
edgesList = list()
```

```
motifsList = set()
```



```

for line in lines:
    line =
    line.replace("\n", "").replace(", ", "")
    row=line.split(" ")

    if row == ['*']:
        if len(nodesList) > 0 and len(edgesList) > 0:
            newNetwork = Network(nodesList, edgesList,
            motifsList,hasFraud)
            networks.append(newNetwork)

        nodesLis
        t=list()
        edgesLis
        t=list()
        motifsLi
        st=set()
        lastNode
        = 0
        lastEdge = 0
        c = -1
        continue

    if c == 0:
        lastNode = int(row[0])
        lastEdge = int(row[1]) +
        lastNodehasFraud = int(row[2])
        c = c + 1;
        continue;

    if c > lastEdge + 2:
        if "Unable" not in line:
            adjmatrix = row[0]
            occ_original = row[1]
            z_score = row[2]
            avg_random = row[3]
            stdev_random = row[4]

            newMotif=Motif(adjmatrix, occ_original, z_score,
            avg_random, stdev_random)
            motifsList.add(newMotif)

    if c > 0 and c <= lastNode:
        nodesList.append(Node(row[1], row[2], row[3]))
    if c > lastNode and c <= lastEdge:
        edgesList.append(Edge(row[0],
        row[1]))
    c = c + 1

```

*number of frequency of subgraphs in all networks/in  
fraudulentnetworks*

```
motifsPresent = set()

for network in networks:
    for motif in network.motifs:
        motifsPresent.add(motif.adjmat
            rix)

for motifMatrix in motifsPresent:
    count = 0;
    fraudCount = 0;
    for network in networks:
        for motif in network.motifs:
            if motif.adjmatrix == motifMatrix:
                count = count + 1
                if network.hasFraud == 1:
                    fraudCount = fraudCount + 1
    print(motifMatrix + ": " + str(count) + " " +
        str(fraudCount))

"011000000": 9030 159
"001100000": 1762 34
"000100100": 454857 5506

#number of networks with and without fraud

filteredNetworksWithFraud = set()
filteredNetworksWithoutFraud = set()

for network in networks:
    if network.hasFraud == 1:
        filteredNetworksWithFraud.add(network)
    else:
        filteredNetworksWithoutFraud.add(network)

print(len(filteredNetworksWithFraud))
print(len(filteredNetworksWithoutFraud))

5509
450677

transactions = list()

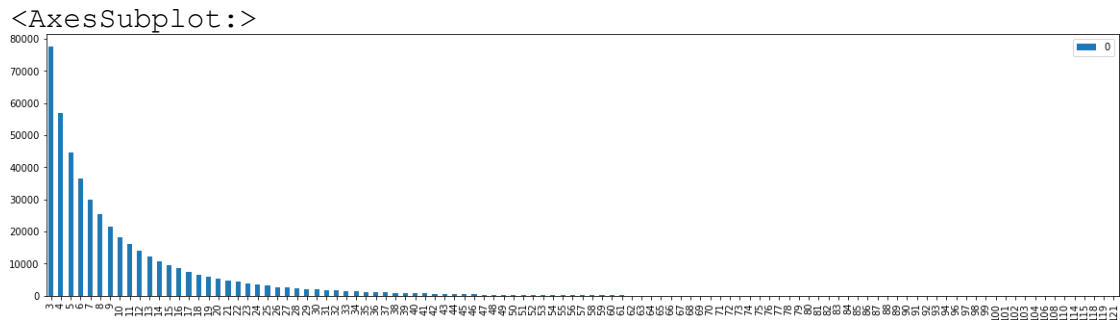
for network in filteredNetworksWithFraud:
    networknodes=network.nodes
    networkTrasactions=network.edges
    for transaction in networkTrasactions:
        transactions.append(networknodes[int(transaction.sender) -
            1].name + "_" +
```

```
networknodes[int(transaction.receiver) -1].name)
```

```
numberOfNetwork=  
list()  
numberOfEdges=  
list() for  
network in  
networks:  
numberOfNodes.append(len(network.nodes))  
numberOfEdges.append(len(network.edges))
```

-----  
-----

```
import pandas as pd  
from collections import Counter  
  
count = Counter(numberOfNodes)  
df10 = pd.DataFrame.from_dict(count, orient='index')  
  
df10.sort_index().plot(kind='bar', figsize=[20,5])
```



```

#creating dataset with information (per transaction) from the
networks file and Paysim original file

#1-transaction between customer and customer
#2-transaction between merchant and merchant
#3-transaction between customer and merchant

#calculating transitivity and degree centrality per transaction
(edge)

begin = time.time()

transitivityListsender =
list()
transitivityListreceiver =
list()
centralityListsender
= list()
centralityListreceiver =
list()
accountsTypeList =
list()
c2=0

for network in
networks:
    if c2==10000:
        break

    c2=c2+1
    networknodes=network.nodes
    networkTransactions=network.edges
    transactions =
list()
    for transaction in networkTransactions:
        transactions.append(networknodes[int(transaction.sender)
-
1].name + "_" +
networknodes[int(transaction.receiver) -1].name)

df2=df.loc[(df['Orig_Dest'].isin(transactions))]

nodesindex=list()
edgeslist=list()
c=1

for n in network.nodes:
    nodesindex.append
d(c)
c=c+1

for e in network.edges:
    edge=[int(e.sender),

```

```

int(e.receiver)]
edgeslist.append(edge)

Paysim_graph= nx.Graph()
Paysim_graph.add_nodes_from(nodesin
dex)
Paysim_graph.add_edges_from(edgesli
st)

centralityArray=nx.degree_centrality(Paysim_graph)
centralityArrayList= list(centralityArray.values())

for transaction in networkTransactions:
    index = networkTransactions.index(transaction)
    if ("C" in
transactions[networkTransactions.index(transaction)]
and "M" in
transactions[networkTransactions.index(transaction)]):
        accountsType = "3"
    elif "C" in
transactions[networkTransactions.index(transaction)]:
        accountsType = "1"
    elif "M" in
transactions[networkTransactions.index(transaction)]:
        accountsType = "2"
    accountsTypeList.append(accountsType)

    transitivitysender = nx.clustering(Paysim_graph,
int(transaction.sender))
    transitivityreceiver = nx.clustering(Paysim_graph,
int(transaction.receiver))
    transitivityListsender.append(transitivitysender)
    transitivityListreceiver.append(transitivityreceiver)

    centralityListsender.append(centralityArrayList[int(t
ransaction.sender)-1])
    centralityListreceiver.append(centralityArrayList[in
t(transaction.receiver)-1])

#####
#####

alltransactions=
list()
transactionsDF=l
ist()

for network in networks:
    networknodes=network.nodes
    networkTransactions=network.edges
for transaction in networkTransactions:
    alltransactions.append(networknodes[int(transaction.sender)

```

```

-1].name + "_" + networknodes[int(transaction.receiver) -
1].name)
c4=0

for transaction in alltransactions:
    if c4==len(centralityListsender):
        break

    c4=c4+1
    index = alltransactions.index(transaction)
    df4 = df.loc[df['Orig_Dest'] == transaction]
    indexdf= df4.index[df4['Orig_Dest'] ==
transaction].tolist()[0]amount= df4.loc[indexdf, 'amount']
    step = df4.loc[indexdf, 'step']
    hasfraud = df4.loc[indexdf, 'isFraud']
    oldbalanceorig = df4.loc[indexdf, 'oldbalanceOrg']
    oldbalancedest = df4.loc[indexdf, 'oldbalanceDest']
    newbalanceorig = df4.loc[indexdf, 'newbalanceOrig']
    newbalancedest = df4.loc[indexdf, 'newbalanceDest']

    accountstype = accountsTypeList[index]
    transitivitysender =
transitivityListsender[index]
    transitivityreceiver=
transitivityListreceiver[index]centralitysender
= centralityListsender[index]
    centralityreceiver=
centralityListreceiver[index]

    transactionDFObject = [amount, step, hasfraud,
oldbalanceorig,oldbalancedest, newbalanceorig,
newbalancedest, accountstype, transitivitysender,
transitivityreceiver, centralitysender, centralityreceiver]
    transactionsDF.append(transactionDFObject)

#####
import csv

transactions_header = ['amount', 'step', 'hasfraud',
'oldBalanceOrig','oldBalanceDest','newBalanceOrig',
'newBalanceDest', 'accountsType', 'transitivitySender',
'transitivityReceiver', 'centralitySender',
'centralityReceiver']

with open('rapidminerDB.csv', 'w')
as file:
    writer = csv.writer(file)
    writer.writerow(transactions_he
ader) for line in
transactionsDF:
        writer.writerow(line)

end =

```

```
time.time(  
)  
print(end-  
begin)
```

### 8.3. Gtries and Subgraphs Detection

**3 nodes command:** -s 3 -d -f simple -m esu -g gtriesinput.txt -oc file.txt

**4 nodes command:** -s 4 -d -f simple -m esu -g gtriesinput.txt -oc file.txt

```
import pandas
as pd import
networkx as
nx
#import matplotlib.pyplot as plt
import
pickle
import
matplotlib
inline
import
warnings
import
subprocess
import os

with open("Paysim3nodes.txt", "r") as networks:
    with open("gtriesallinputs.txt", "w") as gtriesallinputs:
        alreadyWritten=0
        for line in networks:
            if len(line.split()) ==
                2: alreadyWritten=0
                gtriesallinputs.wri
                te(line)
            if '\"' in line and alreadyWritten==0:
                gtriesallinputs.write("*\n")
                alreadyWritten = 1

import
fileinput
import
subproces
s
cmd = r"E:\joana\tese\gtrieScanner.exe -s 4 -d -f simple -m esu
-g gtriesinput.txt -oc file.txt"

gtriesinput.txt"begin = time.time()

with open("gtriesallinputs.txt", "r") as
    gtriesAllInputs:gtriesInput =""
    for line in gtriesAllInputs:
        if
            len(line.split
            ()) == 2:
                gtriesInput+=1
```



```

        line continue
    if '*' in line:
        with open("gtriesinput.txt", "w") as
            gtriesInputTxt:
                gtriesInputTxt.write(gtriesInput)
        process = subprocess.Popen(cmd,
stdout=subprocess.PIPE,creationflags=0x08000000)
        process.wait()
        gtriesInput = ""
        with open("results.txt", "r") as results:
            with open("filteredResults.txt", "a") as
                filteredResults:
                    motif = '\n'
                    for line in results:
                        if line.startswith("1") or
                            line.startswith("0"):
                            if len(line.split()) ==1:
                                motif += line[0:4:1]
                            else:
                                array=
                                re.split("\s{2,}",line)motif=
                                motif +array[0][0:4:1]+'\'\'', '+
                                array[1][0:array[1].index("|")]
                                +', '+ array[2][0:4:1] +', '+
                                array[3][0:4:1]+'\' '+
                                array[4][0:4:1] +'\n'
                                filteredResults.write(moti
                                f)motif = '\n'
                    filteredResults.write("*\n")

end = time.time()
print(f"Total runtime of the program is {end - begin}")

Total runtime of the program is 32119.045303821564

motifs = list()

with open("filteredResults.txt", "r") as filteredResults:
    motif = ""
    for line in filteredResults:
        if "*" not in line:
            motif+= line
        else:
            motifs.append(motif)
            motif = ""

motifsListIndex = 0
motifsCounter = 0

for line in fileinput.input("Paysim3nodes.txt", inplace=True):

```

```

if '\"' in line and motifsCounter == 0:
    if motifs[motifsListIndex] == "" :
        print("Unable to find 4 nodes motifs")
        motifsCounter = motifsCounter + 1
        motifsListIndex = motifsListIndex + 1
    else:
        print(motifs[motifsListIndex], end="")
        motifsCounter = motifsCounter + 1
        motifsListIndex = motifsListIndex + 1
elif '\"' in line and motifsCounter != 0:
    print("", end="")
else:

    print(line, end="")
    motifsCounter = 0

```

## 8.4. RapidMiner's Process and Model

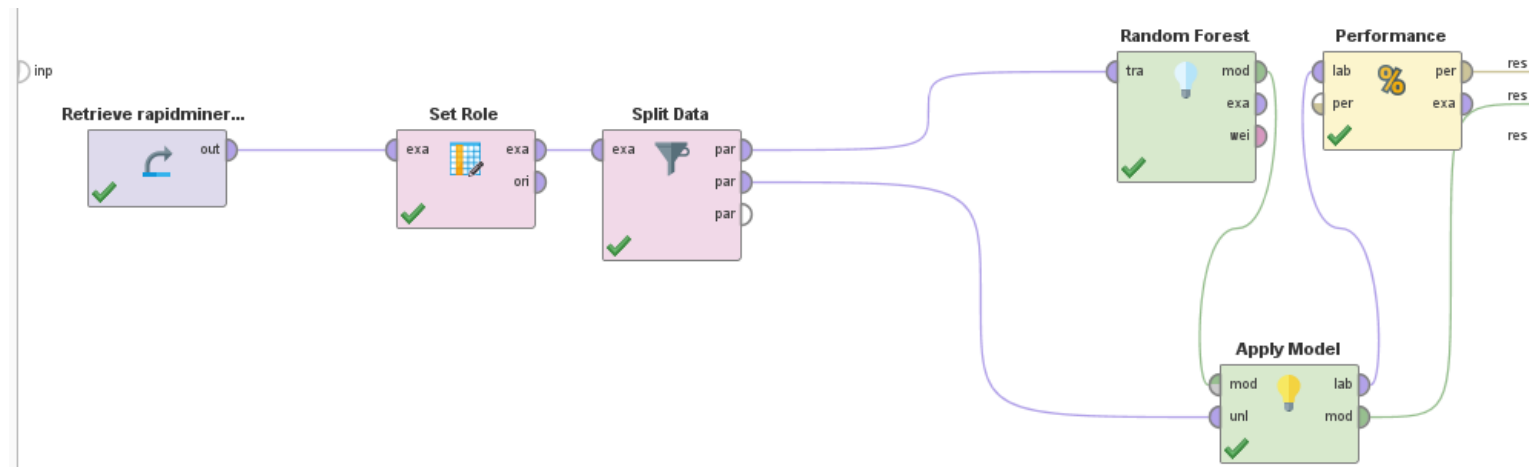


Figure 18: RapidMiner's Process before Under Sampling. Source: the author.

Row No.	hasfraud	amount	step	oldBalanceOrig	oldBalanceDest	newBalanceOrig	newBalanceDest	accountsType	transitivitySender	transitivityReceiver	centralitySender	centralityReceiver
1	0	244486.460	249	8946	526950.370	0	771436.840	1	0	0	0.143	1
2	0	125143.120	234	763	401807.250	0	526950.370	1	0	0	0.143	1
3	0	117748.900	187	103	0	0	117748.900	1	0	0	0.143	1
4	0	203078.890	230	114272	198728.360	0	401807.250	1	0	0	0.143	1
5	0	345504.920	328	0	771436.840	0	1116941.760	1	0	0	0.143	1
6	0	195101.130	372	10132864.150	1116941.760	10327965.270	921840.630	1	0	0	0.143	1
7	0	80979.470	203	12995	117748.900	0	198728.360	1	0	0	0.143	1
8	0	261877.190	231	7596	1126627.700	269473.190	864750.510	1	0	0	0.067	1
9	0	87762.750	277	6019	1304253.670	93781.750	1216490.920	1	0	0	0.067	1
10	0	45690.820	288	25763	1216490.920	0	1262181.740	1	0	0	0.067	1
11	0	162654.690	328	5074171.690	1489567.430	5236826.380	1326912.740	1	0	0	0.067	1

Figure 19: RapidMiner's Dataset for model creation. Source: the author.

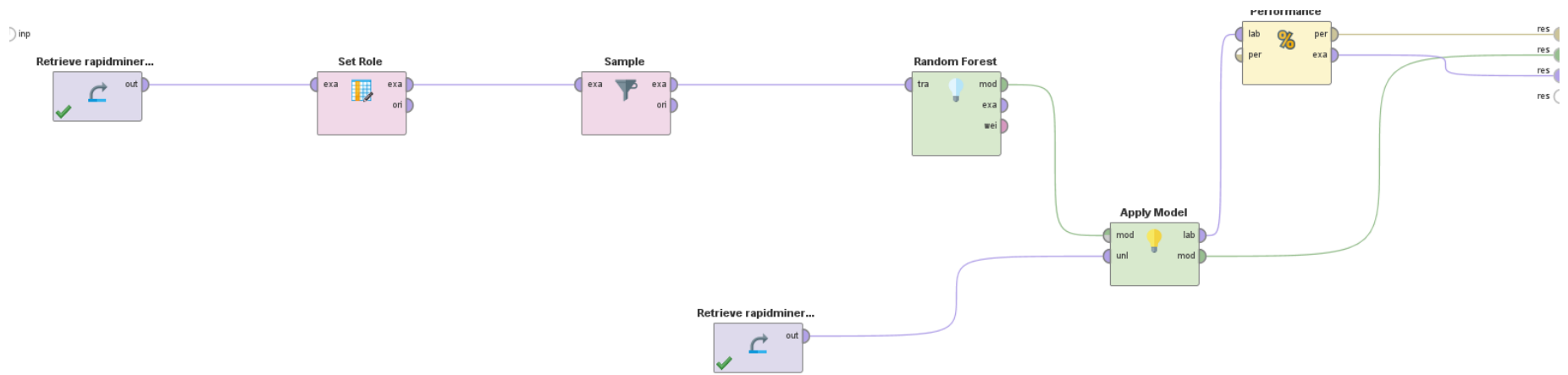


Figure 20: Final RapidMiner process including Random Under Sampling. Source: the author.

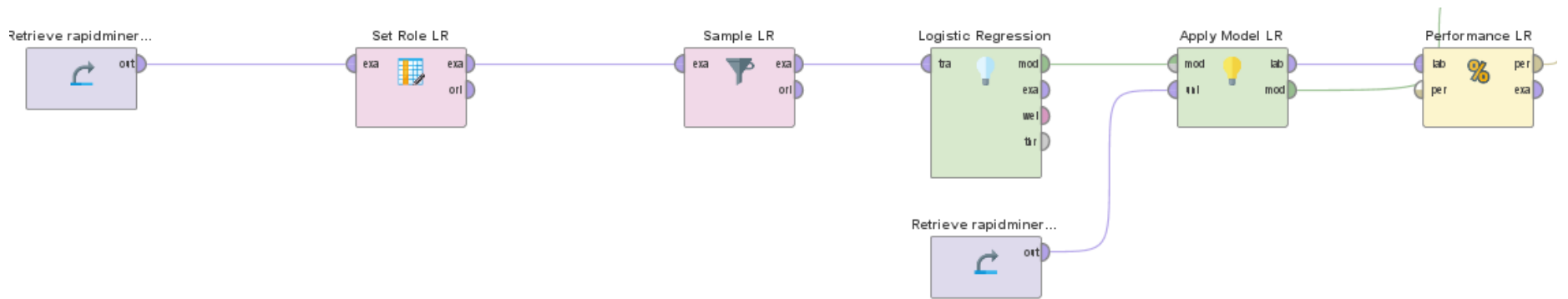


Figure 21: Final RapidMiner process including Logistic Regression. Source: the author.

FACULDADE DE ECONOMIA

