Ecole Nationale d'Ingénieurs de Tunis
Ecole Doctorale ST
U2S Laboratory

Paris Descartes University

Master Thesis

This dissertation is submitted for the obtention of

Master of Research Diploma

Information Processing and Complexity of the Living

Presented by

Rim BACCOUR

Analysis of financial transactions in Bitcoin

Defended on, February 2

President of the jury	**: Mr. Nicolas LOMENIE**
Thesis examiner	**: Mr. Michel SOTO**
Lab advisor	**: Mr. Rémy CAZABET**

Achieved within

Laboratoire d'Informatique de Paris **6**
Université Pierre & Marie Curie

Academic year : 2016 - 2017

Signatures

Approved by Mr. Remy CAZABET

Dedication

I dedicate this modest work ...

To my great, kind, strong, compassionate, good-hearted fighter & friend.
To the one to whom I owe all, to my rare pearl, to my mother.

To my father who never stopped being my great and my faithful idol.

To my little sunshine Mahdi, for all the joy and happiness he brings to my life.

To my unique love, my ultimate refuge, my kind-hearted partner, Amine, for making my life worth-living.

To my great ally and companion throughout this journey, Nadine, for all moments of happiness and sorrows we shared.

To all my teachers and professors, I dedicate this modest work as a testimony of affection and infinite gratitude.

Acknowledgement

First, I would like to express my regards and my gratitude to Complex System department director Mr. Matthieu LATAPY and Complex Networks team leader Mrs. Clemence MAGNIEN who trusted and welcomed me within the Computer Science Lab of Paris 6 (LIP6) at Pierre and Marie Curie university.

Thank you for the chance you gave me to work within Complex Networks unit. Being a member of such an amazing and brilliant team brought me a great benefit at the personal and professional level.

I am deeply thankful, in particular, to my lab advisor Mr. Remy CAZABET at LIP6, who was present throughout the whole project and never stopped guiding me with his precious advices, his knowledge and his support.

I am sincerely grateful for your assistance during both my graduation and master projects.

I would like to express also my sincere gratitude to Mrs. Imene ELLOUMI, who never stopped supporting and encouraging me throughout the development of this project.

I am also indebted to my professors during all my academic path especially Mrs. Mariem Jaidane, the one who inspired me during my journey at ENIT.

I am also thankful to all my family and friends without whom I wouldn't have the chance to live such an enriching and rewarding experience.

Finally, I would like to thank jury's members for assessing my master project.

Preambule

This master research thesis is the follow-up of our six-month graduation internship within LIP6 of Pierre and Marie Curie university in France.

We were interested in examining bitcoin currency, the first cryptocurrency that appeared in 2008, its unusual transaction patterns and use cases. Our work is derived of a more global initiative called iTRAC, involving many industrial partners and research labs, with the major goal of conceiving an efficient framework for supicious and fraudulent behaviours detection.

Delving a big dataset of monatery exchanges occuring between public addresses and anonymous identities enabled us to conceive, through our study, a prototype for de-anonymizing bitcoin users, build a model to characterize nodes' activities, identify patterns of identified users' profiles and inspect patterns of suspicious and fraudulent behaviours.

Abstract

Bitcoin, the first crypto-currency to appear in 2008, has nowadays reached a widespread use and raised an increasing interest for its unusual transactional patterns. As it offers high-level anonymity for its users, this virtual currency is reputed to be highly used for fraudulent activities.

Since, fetching suspicious patterns in bitcoin network has become a serious issue to deal with using mature techniques from network analysis and statistics. Machine learning offers, in turn, a set of algorithms to fulfill such a challenging purpose.

In this project, we propose a prototype for users' discovery in bitcoin network and user profiles identification using parallel and distributed computing environment offered by Spark and a set of machine learning algorithms. Extracted insights are then used to characterize identified users and inspect fraudulent bitcoin users' activities.

Key words: virtual currencies, Bitcoin, frauds pattern, graph analysis, data analysis, Big data.

Contents

List of Figures

List of Tables

List of Acronyms

aNMI **a**djusted **N**ormalized **M**utual **I**nformation
APIs **A**pplication **P**rogramming **I**nterfaces
BTC **B**itcoin
Dash **D**arkcoin
DAG **D**irected **A**cyclic **G**raph
DFT **D**iscrete **F**ourier **T**ranform
DWT **D**iscrete **W**wavelet **T**ranform
ECDSA **E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
ETH **E**thereum
FN **F**alse **N**egative
FP **F**alse **P**ositive
FT **F**ourier **T**ransform
Gb **G**igabyte
HAC **H**ierarchical **A**scendant **C**lassification
HDFS **H**adoop **D**istributed **F**ile **S**ystem
ICT **I**nformation and **C**ommunication **T**echnologies
Id **I**dentifier
IoT **I**nternet **o**f **T**hings
JDBC **J**ava **D**atabase **C**onnectivity
LOF **L**ocal **O**utlier **F**actor
LTC **L**itecoin
ML **M**achine **L**earning
NMI **N**ormalized **M**utual **I**nformation
OS **O**perating **S**ystem
PCA **P**rinciple **C**omponent **A**nalysis
RDD **R**esilient **D**istributed **D**ataset
SSH **S**ecure **SH**ell
SVM **S**upport **V**ector **M**achine
TP **T**rue **P**ositive
WT **W**avelet **T**ransform
YARN **Y**et **A**nother **R**esource **N**egotiator

General Introduction

Context and problematic

Since the dawn of time, monetary exchanges between people have been established on the trust-based model. Financial institutions, playing the role of watch dog and trusted third parties, ensure the supervision of transactions' integrity and the regulation of financial funds. Then, with the advent and the rapid expansion of the internet, the rapid increase of online commerce and the drastic proliferation of e-payments, parties soon realized the drawbacks of centralized transactions. It was mainly the vulnerability of financial institutions and central banks to corruption, the lack of transparency, the rise of swindle and high transactions costs that have led to the inception of virtual crypto-currencies such as bitcoin and its followers which came to support the idea of direct monetary exchanges.

Bitcoin, the first virtual currency to appear in 2008, Ethereum, Litecoin and many other crypto-currencies have the great ingenuity in replacing traditional financial intermediaries by a set of technologies including transactions' timestamping, peer-to-peer (P2P) networks, cryptography and shared computational power along with a consensus algorithm allowing anonymous parties to transact and a public and irreversible ledger to track visibly and continuously the flow of money ownerships. This innovative financial decentralized paradigm has redefined the notion of ownership as bitcoin is imperatively not associated to a fiat currency, coins are simply virtual and are exchanged within a network of anonymous identities that are not tied to real world individuals.

However, the anonymous nature of bitcoin users who transact using a multitude of public addresses on bitcoin network, the rise of bitcoin's purchasing power, the trustless nature of bitcoin transactions and the increasing number of exchanges rate on bitcoin network made it substantially associated to suspicious activities such as money laundering, purchasing illegal products and thefts. All these reasons make the growing record of bitcoin transactions that have ever been performed an interesting big data set to delve. Fetching and identifying suspicious patterns on bitcoin network become actually a major issue which could help limit the proliferation and the widespread of illegal actions.

The current project falls within this perspective. Designing and developing a batch-processing prototype that aims to recover users' activities from a set of anonymous transactions, classify bitcoin users according to a specified set of users profiles and inspect fraud patterns in this peer-to-peer network is the major outcome of this work relying on the cross-disciplinary synergy of big data technologies, machine learning algorithms, time series analysis and graph theory.

Project Framework

LIP6: Hosting laboratory

This project was realized within Complex Networks team, a unit attached to the Computer Science Lab of Paris 6 (LIP6) of Pierre and Marie Curie university, one of the most known universities in France.
Since its launch in 2008, the activity of Complex Networks team members is mainly focused on studying complex graphs and their numerous aspects including their dynamical and static proprieties, their evolution over time as well as their structural features, as they remain one of the best theoretical modelling systems for real-world networks such as social networks, internet topology and peer-to-peer exchanges [1]. This research area represents a real-added value as it brings responses to unsolved questions and provides key insights on essential topics of high interest such as explaining users' behaviours and preferences in a social graph and predicting networks evolution, instants of stability and finally attacks detection [1].

Project Context: iTRAC proposal

Since 2016, Complex Networks team has been part of a strategic project named iTRAC, involving many industrial partners such as Thales Communication and Security and Paymium as well as many other research labs, aimed to investigate different use cases of crypto-currencies which become a potential and a powerful competitor to fiat currency.
This initiative was proposed to respond to the growing concern about the use of virtual crypto-currencies for criminal and fraudulent purposes such as purchasing illegal substances or contraband, cyber-criminality, money laundering and financing terrorists and illegal organizations [2].
Analyzing bitcoin users exchanges and detecting fraudulent behaviours actually falls in the heart of Complex Networks activity as it can be solved at the crossroad of a panoply of disciplines, namely: Graph theory, Machine Learning (ML) and Time Series Analysis.
In the other hand, fetching fraudulent pattern in an exponentially growing data set of financial transactions requires the implementation of a set of efficient algorithms with high calculation complexity. This is where Big Data interferes to offer an efficient computing environment different from the traditional one. Thus, we suggest in this work a solution to identify bitcoin users, quantify their behaviours and study fraudulent activities pattern using a Big Data platform that handles the complexity of such task.

Project objectives

The main objectives of the present project are :

1. Setting up a big data framework to clean bitcoin transactions dataset.
2. Proceeding with the users' discovery process through the implementation of bitcoin protocol's intrinsic heuristics.
3. Setting up an evaluation process to measure each heuristic robustness.
4. Constructing users' transaction graph.
5. Conceiving the feature extraction module.
6. Proceeding with the clustering task to identify different bitcoin users' profiles.
7. Exploring users' wallets evolution and investigating present patterns in the network.

Thesis Overview

The present thesis is the essence of this project and includes three main chapters detailing the work achieved. In the first chapter, we will provide an extensive theoretical study behind our work including virtual crypto-currencies and particularly its trendsetter, the bitcoin currency. Secondly, we will present also a literature review covering main researches about bitcoin currency. The third and final part of this chapter is dedicated to exposing an overview about all paradigms involved in our work. We investigate how techniques derived from these fields perform separately as well as when combined to capture bitcoin user profiles and analyze fraudulent behaviours.
Along the second chapter, we specify the high level architecture of the proposed solution and we describe the detailed design of each module.
As for the third and final chapter, we deal with the implementation of our system and discuss the experimental results of each of the developed modules.
In the conclusion chapter, we provide a summary of our work, its limitations as well as its perspectives.
At last, we support this report by an appendix where we expose in details each framework and each algorithm used during this project.

Chapter 1

Scope of the study

Introduction

This chapter is dedicated to exposing the master project's theoretical segment. Throughout its sections, key concepts about crypto-currencies as well as all technological aspects involved in this project will be highlighted for further reference.
We will also provide a literature review exposing a set of flagship scientific researches which aimed to investigate bitcoin payment system.
All these preliminary elucidations will precipitate the demonstration of the project's design in the following chapter.

1.1 Virtual currencies background

The emergence of crypto-currencies made strides over digital currencies and online payment systems always relying on the same common monetary scheme of cash where mechanisms for establishing ownership, protecting against thefts and double spending are ensured by one same central authority. Nowadays, these virtual currencies are no longer experimental and exclusively reserved for computer science specialists; in contrary, they have reached a widespread use and have raised an increasing interest for their unusual transactions' patterns.

We will detail in the following subsections substantial aspects about these emerging currencies and we will focus particularly on its trendsetter; the bitcoin currency.

1.1.1 Virtual currencies proliferation

The first crypto-currency to appear was bitcoin when Satoshi Nakamoto, presumably a pseudonym for a group of hackers, published a detailed paper about this new decentralized currency protocol and its operational mode on October, 2008 [3]. The idea of deploying a totally decentralized payment system was revolutionary at that time. Bitcoin has then captured attention and achieved large-scale acceptance as the pioneer currency that turned this theoretical concept into a real and an operational direct monetary system. Crypto-currencies are inherently independent of governments and centralized authorities; All trusted third parties' tasks remain certainly in crypto-currencies' scheme but are ensured by technology only [3]. It is about a

combination of a cryptographic proof of work deployed on a peer-to-peer network of users transacting anonymously using pairs of public and private keys generated around a specific encryption algorithm and a public ledger storing the entire history of transactions and currency ownerships.

Virtual currencies have also revolutionized the notion of ownership as they do not exist; they are simply virtual and unbacked by either physical commodities or sovereign obligation [4]. Transactions are performed purely and solely in an electronic form with no intrinsic value.
In fact, fiat currencies have been based conventionally on gold. However, crypto-currencies are totally virtual, there is no physical object or even a digital file to point to representing a specific currency. Apparently, we can talk about someone having virtual coins, but when looking at a particular address in the network, there are no digital coins held in it in the same way that dollars are held in a bank account. Instead, there are only records of transactions between different addresses, with balances that increase and decrease.

1.1.2 Existing virtual currencies

The success of the crypto-currencies' trendsetter ushered in a wave of new virtual currencies inspired by bitcoin. They are collectively called altcoins and presented themselves as modified or improved versions of bitcoin.

The inception of virtual currencies helped provide a more flexible payment system striving fiat currencies' vulnerabilities, namely [4]:

- Low friction e-commerce as transactions imply low fees comparing to charges and expenses imposed by fiat currencies.
- Flexible payment scheme allowing merchants and customers from all over the world to exchange services, goods and funds without the obligation of passing by established banking systems and complicated financial solutions.
- Reduction of frauds related to trusted third parties corruption.
- Quick and instant funds transfer.

The following table (1.1) exposes at-a-glance differences between the four most known crypto-currencies.

Table 1.1: Characteristics of four most known crypto-currencies

	Bitcoin	**Litecoin**	**Darkcoin**	**Ethereum**
Deployment date [5]	03/01/2009	07/10/2011	18/01/2014	30/07/2015
Creator name [5]	Satoshi Nakamoto	Charles Lee	Evan Duffield	Vitalik Buterin
Coin Limit [5]	21.000	84.000	18.900	No official declared limit.
Encryption algorithm for blocks' hash [5]	SHA-256	Scrypt	X11	Ethash
Consensus algorithm	Proof of work	Proof of work	Proof of work	Proof of stake
Mean block validation time [5]	10 minutes	2.5 minutes	2.5 minutes	15 seconds
Initial reward for mining [5]	50 BTC	50 LTC	Ranges from (5 to 25)	5 ETH
Current reward for mining [5]	12.5 BTC	50 LTC	Ranges from (5 to 25)	5 ETH
Exchange value in $ [5] & [6]	1 BTC = $8026.18	1 LTC= $70.87	1 Dash = $443.51	1 Eth= $357.74
Market cap [6]	$133,947,766,496	$3,826,979,966	$3,415,721,532	$34,321,895,466

1.2 Bitcoin overview

Bitcoin represents our study case in this master project. The overall design and structural properties of this crypto-currency are highlighted in the following subsections.

1.2.1 Bitcoin protocol

After Satoshi Nakamoto's paper about bitcoin was published in 2008, the deployment of bitcoin open source protocol took place on January 3th, 2009. Finally, first testing transactions begun on January 12th of the same year to announce that bitcoin payment system has become fully operational [4].

Existing solely in electronic form, bitcoin does not incur legal authorities control and relies on a set of protocol processes and settlements to ensure both the currency sustainability and integrity.

Bitcoin payment scheme can be described as below :

A peer-to-peer network constituted by a set of anonymous nodes (bitcoin users) using a private key and an expandable set of arbitrary aliases or public keys generated instantly with no charges around an encryption system named Elliptic Curve Digital Signature Algorithm (ECDSA).
A bitcoin user receives funds through one of its reusable public addresses. Considering that transactions are signatures, he will certainly need the corresponding private key to prove his ownership of transferred funds in order to sign new transactions and re-assign previously assigned bitcoin amounts to one or more public identities new payees control.
Consequently, we conclude that transactions can have one of the following combinational formats:

- multi-inputs and one output
- one input and one output
- one input and multi-outputs
- multi-inputs and outputs

The transaction's format depends essentially on bitcoin amounts to pay and bitcoin amounts held in each of the payer's public addresses as shown in figure 1.1.

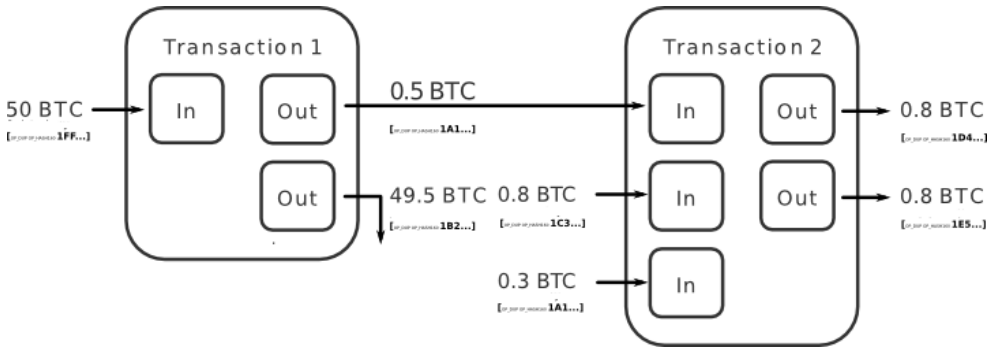

Figure 1.1: Examples of transactions' format

The validity of all performed transactions is ensured by a collective consensus algorithm of the network participants, known as proof of work algorithm, through which every new transaction is appended to a public ledger, called the blockchain, containing history of all previously agreed-upon transactions [5].

The proof of work consensus algorithm involves solving a computational challenging puzzle in order to create new blocks in the bitcoin blockchain. Colloquially, the process is known as mining, and the nodes in the network that engage in mining are known as miners. The incentive for mining transactions lies in economic payoffs, where competing miners are rewarded new bitcoins and a small transaction fee for performing a repeated and a continuous computation of a cryptographic hash of a block containing these four pieces of metadata [3] (see figure 1.2):

- The timestamp.
- The hash of the previous block.
- The proof of work which is a data field called nonce. The nonce will be recalculated continuously in the block until the right format of the block's hash, specified previously by the bitcoin protocol is mathematically proved by the miner.
- The Merkle tree root for the transactions included in this block.

The Merkle tree, also known as a binary hash tree, is a data structure that is used to store hashes of the individual data in large datasets in a way to make the verification of the dataset efficient. It is an anti-tamper mechanism to ensure that the large dataset has not been changed. The word tree is used to refer to a branching data structure in computer science, as seen in the figure below [8].

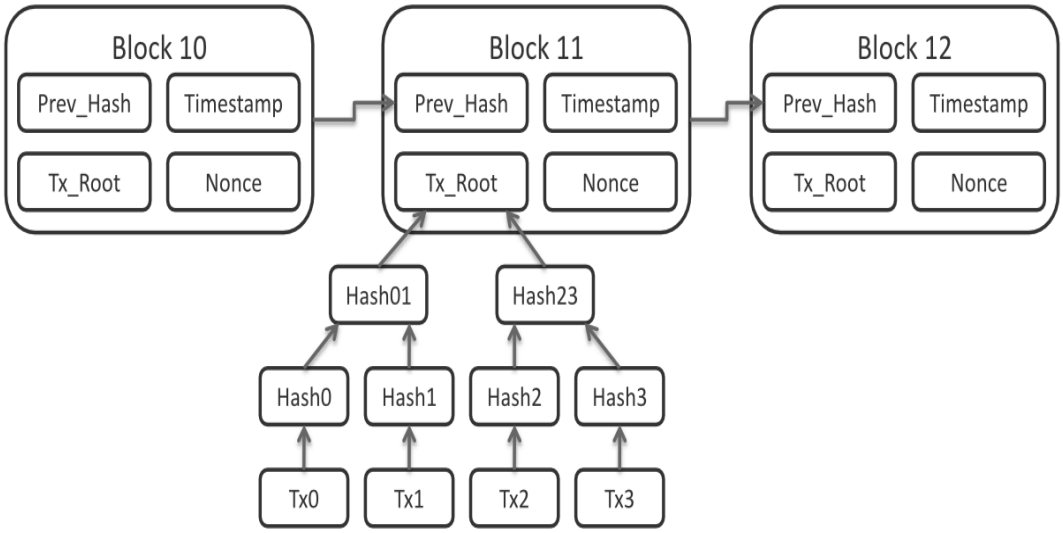

Figure 1.2: Bitcoin transactions' block structure

The mining process refers also to the solution through which bitcoin protocol increase the money supply. In fact, to encourage bitcoin nodes to mine transactions'

blocks which actually requires huge computational power (CPU processing and GPU processing costs), bitcoin protocol offers a reward of newly generated bitcoins and transactions fees to users who succeed in validating new blocks.

Furthermore, bitcoin algorithm is programmed to generate bitcoins in decreasing quantities up to a final amount of twenty-one million bitcoins. Once this threshold is reached, no more bitcoins will be created. This property is highlighted by the increasing computational complexity of blocks hashes' calculation: As bitcoins are getting more and more rare, the validation process of transactions' blocks is getting rougher and bitcoin rewards are getting lesser. Currently, only mining pools gathering millions of nodes and combining computational resources could confront the increasing complexity of mining process. These gatherings generate blocks quickly and therefore receive a portion of the bitcoin block reward on continuous basis, rather than randomly once every few years. The reward will then be split over all nodes involved in the mining pool. The following figure shows the most prominent mining pools over the last years.

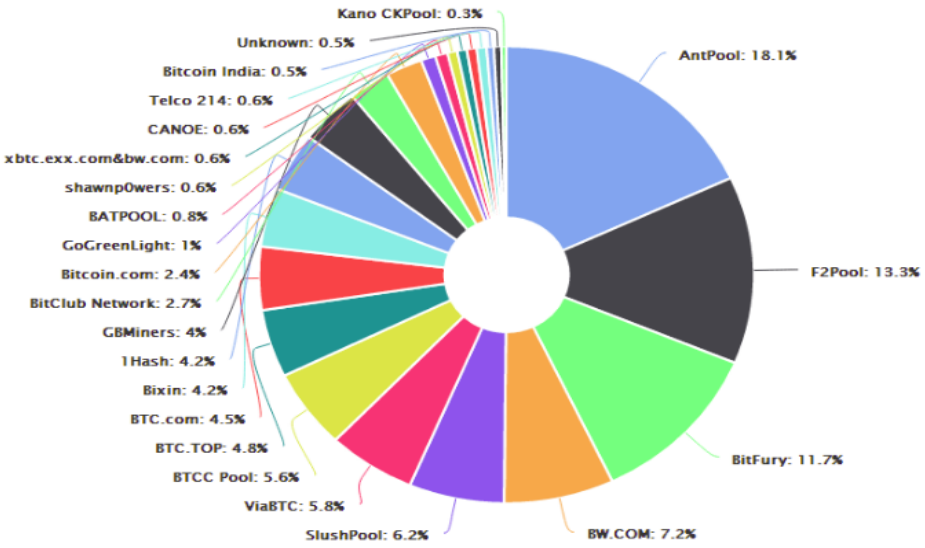

Figure 1.3: Most prominent mining pools of bitcoin currency

Transactions' fees and bitcoin funds are nowadays divided into smaller parts; the smallest divisible amount is one hundred millionth of a bitcoin and is called a Satoshi (1 Satoshi $= 10^{-8} BTC$).

Additionally, double spending is also an issue that had to be considered in the absence of central authorities. Bitcoin protocol mitigated it by the fact that the entire history of bitcoin transactions is publicly available in a distributed database that all nodes share. As a matter of fact, blocks are encrypted using SHA-256

encryption algorithm. Trying to spend an already spent amounts of bitcoins implies cracking the block containing the specific transaction and all posterior blocks chained after it and finally falsifying it. Processing such an important portion of encrypted transactions history demands a huge computational power. As the majority of the bitcoin network nodes are honest and collectively controlling more CPU power than any cooperating group of hackers, double spending is actually not viable [3]. Besides, another reason explaining this option inefficiency is the ever increasing complexity of the encryption algorithm which adjusts its difficulty to maintain approximately the ten minute block creation [3]. All transactions are though non reversible and permanently recorded since the bitcoin origin.

1.2.2 Blockchain

The blockchain is considered arguably as one of the most significant innovations in information technology over the past few years. This public and distributed ledger which came with the advent of bitcoin currency is a specific form or subset of distributed ledger technologies, which constructs a constantly growing chronological chain of blocks, hence the name 'block-chain'. A block refers to a set of transactions that are bundled together and added to the chain at the same time. In the bitcoin blockchain, the miner nodes bundle unconfirmed and valid transactions into a block [9]. Each block contains a given number of transactions. All nodes have a copy of the blockchain, which will be updated each time a new block of transactions is validated and appended to the existing chain.

Satoshi Nakamato explains in his paper the process of appending new blocks to the blockchain at the network level. As bitcoin protocol uses broadcasting to forward new transactions and blocks to add to the blockchain, the probability of generating simultaneously two different versions of next block is quite important due to delays caused by the adopted transfer mode. As a consequence, some nodes will work on these two different versions, constitute and share also two variants of the chain of blocks. Later, nodes will save also the other branch of blockchain in case it becomes longer [3]. The tie will be broken as soon as the next proof-of-work is computed and one variant of the blockchain becomes longer; the nodes that were working on the other branch will then switch to the longer one [3].
This scenario can be expanded to a tuple or even more simultaneous blocks. The following figure (1.4) illustrates the scenario explained where blocks 4 and 2 as well as 3, 7 and 8 where created simultaneously. Finally, the longest version of blocks carries out and all informal variants will be then deleted.

The blockchain is one of the most valuable components of the bitcoin protocol as it holds complete information about all addresses in the bitcoin network from the genesis block to the most recently completed block.

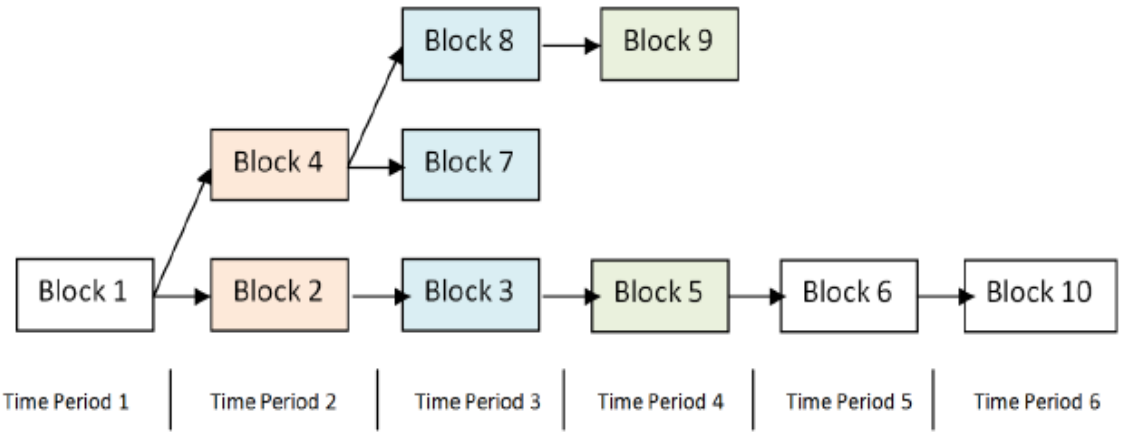

Figure 1.4: Three simultaneous variants of the blockchain

1.2.3 Frauds in bitcoin transactional system

The upward proliferation of decentralized payment paradigms and the rapid growth of crypto-currencies, especially bitcoin's market price, as shown in figure 1.5, have certainly raised high suspicions and doubts as they offer an open source framework for financial exchanges between anonymous and untraceable entities.

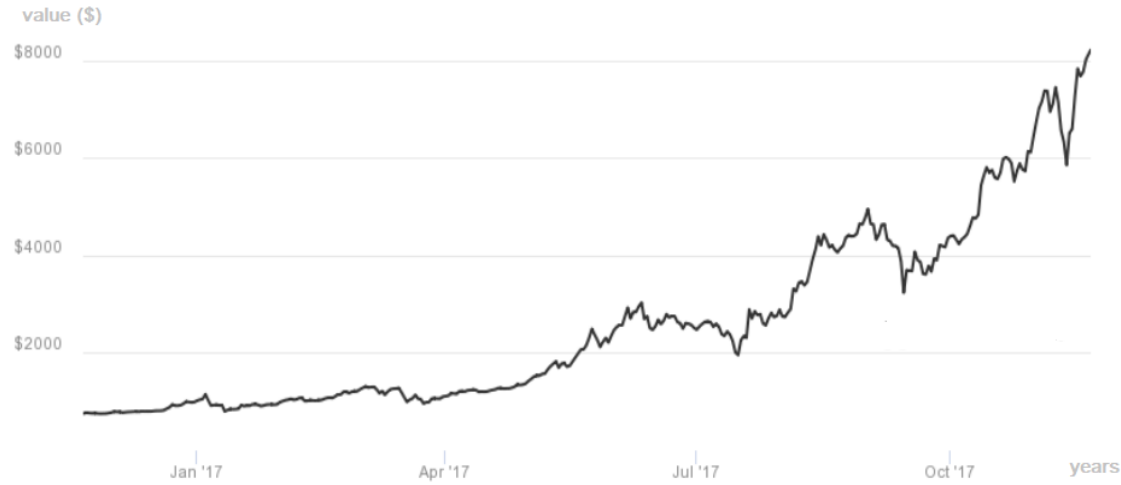

Figure 1.5: Bitcoin market price evolution [10]

Legal authorities and governments, totally uninvolved in this trend, announced profound concerns about the use of virtual currencies which are prone to the spread of suspicious purchases and criminal activities. These worries are explained by the fact that bitcoin design enables funds' transfer occur without any identity verification.

Additional thoughts are triggered by the widespread of numerous mechanisms among bitcoin users promoting nodes' anonymity such as multiplying the number of public keys controlled by one user which makes it difficult for analysis efforts to recover and retrace the user whole activity. Another example is using TOR wallet deployed essentially in the dark web to upgrade the bitcoin user privacy level.

Then, worries are substantiated by real cases:

- A prime example of bitcoin being used in dishonest and illegal trading is Silk road. It is an online marketplace launched in 2011 to facilitate the exchange of illegal substances such as drugs and illicit purchases such as weapons.
- Bitcoin is widely used in the dark web and enables consequently the proliferation of corrupted groups of hackers developing illegal business such as child pornography, narcotics trafficking and identity thefts.
- It also became an appealing tool for drug dealers interested in money laundering actions which can be easily accomplished through mixing services, detailed in [11].

Besides, bitcoin is nominated as the official dark web currency and is strongly used in marketplaces such as Silk Road and Agora for purchasing illegal substances and weapons. In addition, several reports and investigations about bitcoin confirm the preponderance of illegal activities on the bitcoin network given the absence of sanction authorities and delve into the impact imposed by its usage across the dark web [12].

All of these reasons make the bitcoin transactions' history an interesting big dataset to parse. Modelling exchanges using graph theory and studying transactions and users activities using Big data frameworks and Machine Learning techniques can lead to surprisingly meaningful discoveries.

1.3 Literature Review

Virtual crypto-currencies came to found the new paradigm of decentralized financial exchanges where no need to trusted-third parties intermediation. This innovation relies exclusively on technological means; a combination of efficient cryptographic regulation system and a distributed peer-to-peer network infrastructure to substitute social coordination and economic exchanges [13].

Such a great feat, upholding clearly the implied idea of power relation redistribution and decentralization had numerous repercussions affecting many fields. As a consequence, many surveys and analysis were conducted to study the potential impacts of virtual currencies and particularly bitcoin from economic, social, institutional and technological perspectives.

Numerous studies such as [13] and [14] were conducted to analyze the eminent political and institutional changes triggered by the rapid rooting of virtual cryptocurrencies which broke down conventional power politics. [13] presented first a brief history of bitcoin currency since its inception in 2008. Then, De Flippi inquires how far bitcoin payment system obeys the structural architecture of a decentralized governance. The author explains through the example of bitcoin block size crisis the drawbacks of removing a control point and the real challenges a decentralized system governance faces. As for [14], authors introduced cyrpto-currencies as an alternative for outdated and old financial systems currently mismanaged by governors. [15] discusses cryptocurrencies' issue from an economic point of view. In his paper, Harwick presents cryptocurrencies as an illustration of free banking theory and discusses its future in the absence of intermediation.

[7] and [16] investigated another aspect of the bitcoin phenomenon and studied an important protocol dimension. In fact, they focused on the anonymity issue and tried to provide an answer to one of the most relevant questions about bitcoin : To what extent can bitcoin protocol ensure user anonymity ?
Reid and Harrigan wrote an explanatory paper [7] about bitcoin currency, focused on main differences between electronic currencies and virtual cryptocurrencies to introduce in the most efficient manner how bitcoin payment system operate without an issuer or a central authority. They explained how bitcoin transactions occur and the transition from transaction's graph level to user's graph level: two ultimately important components to question the anonymity in the bitcoin protocol. Authors insisted on the fact that bitcoin protocol internal information can only help build a partial user directory when associating public keys to its own controller and the need of external and/ or publicly available data such as personal information of mixing/wallet services, forums to de-anonymize users. [16] investigated the efficiency of bitcoin protocol intrinsic heuristics in de-anonymizing users and focused also on the fact that tracking users using their IP addresses could be done unless bitcoin clients are not using an anonymizing proxy such as TOR.
Added to that, [4] is considered as one of the best reference analysis about bitcoin users activities' because it uses labelled data of bitcoin network. Through this study, researchers carried out transactions with a variety of most known compositional services of bitcoin network in order to label addresses and finally investigated illegal activities occurring in this peer-to-peer network using traffic flow tracking techniques.
[17] as well presents an innovative semi-automatic library to parse the blockchain

data, identify users in the bitcoin network and proceed with users activity analysis. The writer exposes the implementation details of this library and finally presents experiments and study cases on which he tested the deployed solution.
As for [11], researchers highlighted the implication of mixing and anonymizing services for the limited traceability of money flows and presented effective simulations of several possible scenarios of money laundering actions that can be performed using those platforms.
[18], [19] and [20] present emergent outliers detection approaches to detect illegal activities in bitcoin network. In fact, researchers in these surveys considered that executing fraudulent transactions in the bitcoin network is an abnormal behaviour. They applied consequently many anomaly detection techniques such as the LOF, measuring the local deviation of a given data point regarding its neighbours, network analysis techniques such as power Degree and densification laws which generally detect the presence of irregularities when its curves' shapes do not obey a specific law [18] and finally unsupervised SVM and Mahalnobis distance based method to identify outliers [19].

Our work is mainly based on all this existing literature and falls precisely at the intersection of technological and social perspectives... How bitcoin technology which ensures anonymous exchanges, free of charge and traditional third parties regulation could reinforce the proliferation of fraudulent activities in our society?

1.4 Paradigms involved in the project

1.4.1 Graph theory

The field of graph theory owes its beginnings to a mathematical problem solved by Leonhard Euler, a Swiss mathematician who had drawn Koenigsberg, a Prussian town, as a network of line-connected points, where a line replaces a bridge connecting two islands. This gave birth to the first written graph theory paper in 1736. Since then, graph theory has been considered as an active and substantial area for researchers to design virtual and real-life issues [21].

This domain is nowadays used in many fields as large-scale and highly-interconnected networks pervade our society and nature around us, including internet, social networks, genome and scientific databases, medical and institutional records [21].
Theoretically, a graph is a pair of sets (V,E) where V represents the set of vertices (or nodes) and E is the set of edges connecting couples of nodes. We can distinguish directed graphs (figure 1.6), with oriented edges such as Twitter social graph, and undirected graphs in which the edges represent a two-way relationship, such as friendship in Facebook social graph.

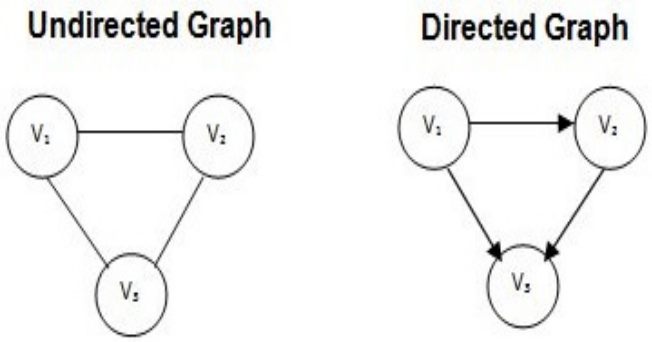

Figure 1.6: Undirected and directed graphs

1.4.1.1 Graph theory for transactional system modelling

Since vertices and edges represent real-life objects (humans, devices, cities...) and relationships between these entities, they usually need to be described through a set of attributes which bring information about nodes and relations' strength and weight in the graph. Graph theory has recently been the most widely adopted solution to which researchers resorted to model and visualize transactional networks data and turn it into valuable insights. Modelling and processing transactions using graph-based computations include two main approaches.

In fact, transactional graphs evolve over time. We can state as an example the variability of currency flows between nodes in financial graphs. These variations induce several changes of nodes' proprieties. We can thus study such graphs from a dynamic perspective and analyze the overtime-evolving properties of nodes, flows and communities. Another option is to freeze time, target a specific date and study the graph elements' proprieties statically.

Finally, whether it is about the dynamic or the static approach, in order to analyze a network, researchers resort to mathematical operations and measures called graph metrics. Use cases include determining important nodes, detecting huge traffic flows and modelling abnormal behaviours and activities.

1.4.1.2 Graph metrics

In transactional graphs, measuring the structural properties is essential to understand hidden patterns embedded in it. This subsection discusses selected measures for this project.

Degree centrality: refers to the number of links a node has. It can be interpreted as the possibility for a vertex to be influenced by the change occurring in the graph [22].

Closeness centrality: measures the minimum steps to take in order reach nodes starting from a particular one. This metric reflects autonomy. Mathematically, closeness centrality for a vertex x in a network of V vertices is defined as [22] :

$$C(x) = \frac{V-1}{\sum_{j \neq x} d(j,x)} \tag{1.1}$$

Betweenness centrality: This measure highlights important nodes; intermediary ones between other vertices. Although it was first defined for undirected graphs, betweenness centrality can be generalized for directed graphs. Mathematically, betweenness centrality for a vertex x in a network of V nodes is defined as [22]:

$$B(x) = \sum_{s \neq x \neq t \in V} \frac{\sigma_{st}(x)}{\sigma_{st}} \tag{1.2}$$

Where σ_{st} refers to the number of shortest paths from node s to node t and $\sigma_{st}(x)$ refers to the number of edges that pass through x.

Clustering Coefficient: The local clustering coefficient measures the tendency of a node's neighbours to form a cluster. Mathematically, the local clustering coefficient is given by the probability that two randomly chosen nodes are connected [21].

Finally, it is important to note that graph metrics are based on algorithms with tremendous computational complexity which will increase enormously when applied on large-scale networks. The following table provides an insight about computational complexity of these algorithms in term of space and time.

Table 1.2: Complexity of centrality metrics algorithms [22]

Algorithms	**Space**	**Time**
Betweenness centrality	$O(V)$	$O(\|E\| \times \|V\|)$
Closeness centrality	$O(V)$	$O(n(\|E\|))$
Shortest path Dijkastra	$O(V^2)$	$O(s \times \|E\| log \|E\| + \|V\|)$
Shortest path Bellman Ford	$O(V^2)$	$O(s \times \|E\| \times \|V\|)$

Note that s refers to the number of sources, n refers to the number of vertices to calculate, V refers the number of vertices in the graph and E refers to the number of edges in the graph. In order to calculate those metrics, we have resolved to big data technologies since they offer distributed processing methods detailed in the next section.

1.4.2 Big Data overview

The integration of digital technologies into everyday life aspects in the new information technologies era has abetted to the explosion of data emanating from various

sources (social networks, mobile phone, computers, transactional data generated using credit cards history, security systems ...).

This massive generation of data along with new opportunities it provides for discovering new insights and solutions to many unsolved problems has given rise to a new concept, commonly referred to as Big Data.

This field takes full advantage of the continuously increasing complexity of available information along with Information and Communication Technologies (ICT) maturity to solve real-life issues. Nowadays, big data is no longer a theoretical approach and helps getting better informative decisions in any sector where data has the potential to be mined and parsed [23].

1.4.2.1 Big Data fundamentals

Big data offers the possibility to explore and analyze abundant amounts of structured, semi-structured and unstructured data coming from heterogeneous sources. This newly created paradigm emerging from the crossroads of old disciplines : statistics, computer science and mathematics [24] can be summarized in five dimensions defined as big data 5Vs (figure 1.7):

- Volume: This dimension emphasizes on the huge amounts of available data and its increasing generation. This is a direct consequence of the rapid evolution of ICT and the recent emergence of a new concept known as the Internet of Things (IoT) where all moral and physical entities are interconnected. Data volume is presumed to pass from Petabytes to Zettabytes in the few coming years.

- Velocity: This measure highlights how fast and massive data flows are produced and processed. Social networks are the best example illustrating the rapidity of data generation. In the other hand, processing data as soon as it is collected become essential to detect financial frauds and cyber security attacks. In this perspective, analyzing processes have evolved from batch to near real-time and even real-time in order to decrease the output's delay.

- Variety: This aspect captures the variety of data types including audio messages, videos, texts, images and formats covering structured, semi-structured and unstructured data. All those formats should be considered, stored and processed to get a complete view about the target issue to solve.

- Veracity: This V indicates the importance of data sources reliability and credibility as well as the process of data cleaning. Analytics should be conducted on clean, pre-processed data in order to eliminate noise and avoid accumulative errors in the analytical process.

- Value: This dimension refers to valuable insights, answers and knowledge we can extract when parsing big datasets.

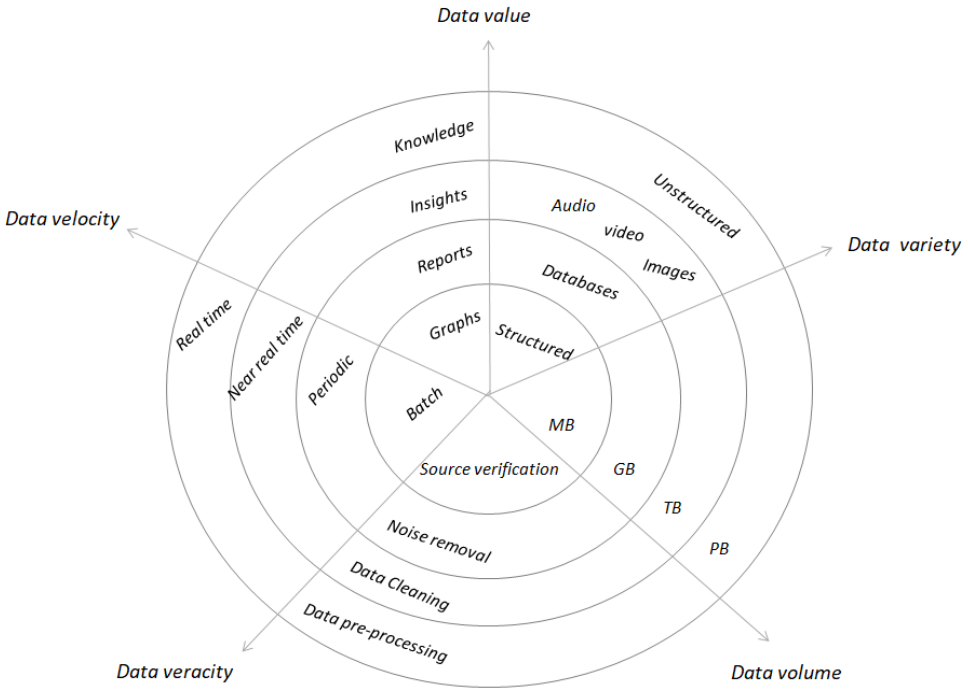

Figure 1.7: The 5Vs that define Big data

Technically, processing big datasets is nowadays ensured by a panoply of frameworks. Hadoop is one of the fundamental frameworks for complex and parallel big data storage and implementation. This pioneer open source big data solution relying on distributed files system and Map Reduce programming paradigm ensures mainly batch processing. Later, with the evolution of big data community needs, Spark; the new processing environment has been created to effectively offer the possibility to manage and analyze huge datasets in both batch and streaming modes [25].

1.4.2.2 Spark: Big Data processing framework

Spark is an open source framework for big data processing developed and managed by Apache Software Foundation since 2014. This project was built to offer an easy to use computing engine extending the MapReduce model and executing parallel distributed processing with APIs for scala, java, python and R programming languages. It was conceived to further extend Hadoop's power and perform batch processing, real-time workloads, interactive queries and machine learning tasks [26].

Spark is a cluster-computing framework that doesnât have its own distributed file system but can actually use Hadoop Distributed File System (HDFS). The overall

design of Spark engine, as shown in figure 1.8, is a stack of different superposed layers where the bottom layer represents Spark core responsible for distributed low level data processing and top layers are Spark own libraries enabling high level tasks.

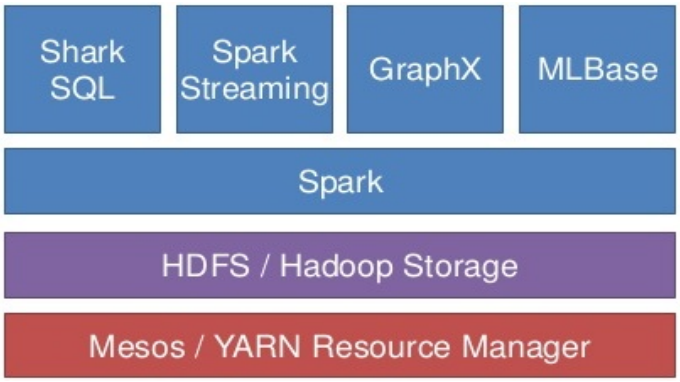

Figure 1.8: Spark framework ecosystem [26]

This stack represents the benefit of higher layers inheriting improvements of the bottom layer; optimization of one of Spark Core APIs will speed streaming, SQL, ML and graph processing libraries detailed in appendix A.
This architecture actually ensures the must-have properties of any big data processing engine which are:

- Fault tolerance
- Scalability
- Performance

1.4.3 Machine Learning overview

Machine Learning (ML) is a field that focuses on building softwares that can learn from past experiences [27]. This is achieved by extracting knowledge from labelled or unlabeled data or both combined. ML is divided into four categories: supervised learning, unsupervised learning, reinforcement learning and transfer learning, all explained in the next paragraphs.

- Supervised learning: In supervised learning, we have an initial information regarding belonging of N observations to different populations. The goal is to build a classification method that predicts to which populations new observations belong. Supervised algorithms deal with labelled data.
- Unsupervised learning: Unsupervised learning and refers to methods deployed aiming to detect patterns for N observations described by p features. This

implies that, although they were collected through the same experience, observations can belong to K classes. With unsupervised algorithms, the objective is to determine for each observation the class to which it belongs.

- Reinforcement learning: This subarea of ML is inspired of behaviourist psychology which thereby makes it a branch of artificial intelligence. It allows machines and software agents to automatically determine the ideal behaviour and reaction within a specific context in order to refine and enhance their performance.

- Transfer Learning: called also inductive learning refers to the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.

1.4.4 Time Series

Time series data is a fundamental data structure that can be defined as set of repeated measurements of a system status or activity over time [28]. This collection of sequential observations helps describing and understanding the temporal evolution and thus hidden and complex dynamics of several real-life processes.

Time series data type is related to a variety of real-life aspects from physiological time series to financial, industrial and ecological time series... [29]. Its interdisciplinary nature made it a relevant research area. So far, time series related researches have focused on two main axes: time series similarities and dissimilarities quantification and analysis techniques including basic time distribution, frequency and wavelet transforms...

However, latest time series analysis approaches have been coupled with ML methods to automate human learning and aid the understanding of its dynamical patterns. This is said, mining time series data has always been a complex issue to deal with due to its continuous and numerical nature [30]. Added to that, this sequential data is also characterized by high dimensionality as it is inconceivable to consider and analyze each time point separately. Time series are also characterized by non linear relationship between its elements [29]. All these reasons fostered researchers to deal with the need of preprocessing this data in order to analyze it. In this context, choosing a specific approach to represent time series is a real issue to deal with in order to achieve substantial dimensionality reduction and preserve at the same time important characteristics of the original data. The choice of which features to use in order to characterize time series is subjective and non-systematic as there is no best representation; the most useful representation form depends actually on the data and the questions being asked of it [28].

1.4.4.1 Time Series representation

1- Time domain representation
Time series data reduction to a manageable size includes a variety of time domain representation techniques. The challenge of all techniques is to preserve relevant features of the original data and remove random and useless noise. Time domain representation techniques, highlighted in figure 1.9, include:

1. Sampling the original data. The choice of the sampling frequency is critical. Low frequency will result in the original shape distortion and the loss of pertinent information, in the contrary, high frequency will put off the problem of high dimensionality as it is [30].

2. Identification of **P**erceptually **I**mportant **P**oints **(PIP)** is a summarization technique of a 1D signal which consists of representing it with N perceptually significant points [30].

3. **P**iecewise **A**ggregate **A**pproximation **(PAA)** consists of splitting the signal into a fixed number of non-overlapping segments and fitting local models to each of them. Two variants of piecewise approximation exist. Each segment length can be determined using the quantification of homogeneity of the aggregated data, another piecewise approximation variant uses constant length for all segments [29].

2- Frequency domain representation
Frequency domain representation focuses on projecting the original time series into the frequency domain using Discrete Fourier Transform (DFT) or into tiling of the time-frequency plane using Discrete Wavelet Transform (DWT) [31]. While time-based representation focuses on capturing local and global trend changes of the time-evolving signal, frequency transformations can help capture relevant and discriminative frequency patterns. Both the FT and the WT express the signal as coefficients in a function space spanned by a set of basis functions. The basis of the FT contains only the complex exponential function. The basis of the WT consists of infinitely scaled and shifted versions of a mother wavelet function.

$$c_f = \frac{1}{\sqrt{(n)}} \sum_{t=1}^{n} f(t) exp(\frac{-2i\pi ft}{n}) \tag{1.3}$$

The Discrete Wavelet Transform (DWT) measures frequency at different time resolutions and locations.

$$\Psi_{j,k}(t) = 2^{\frac{j}{2}} \Psi(2^j t - k) \tag{1.4}$$

where ψ can commonly be the Haar wavelet with the mother function [31].

$$\Psi_{Haar}(t) = \begin{cases} 1, \text{if } 0 < t < 0.5 \\ \text{-1, if } 0.5 < t < 1 \\ 0, \text{otherwise} \end{cases} \tag{1.5}$$

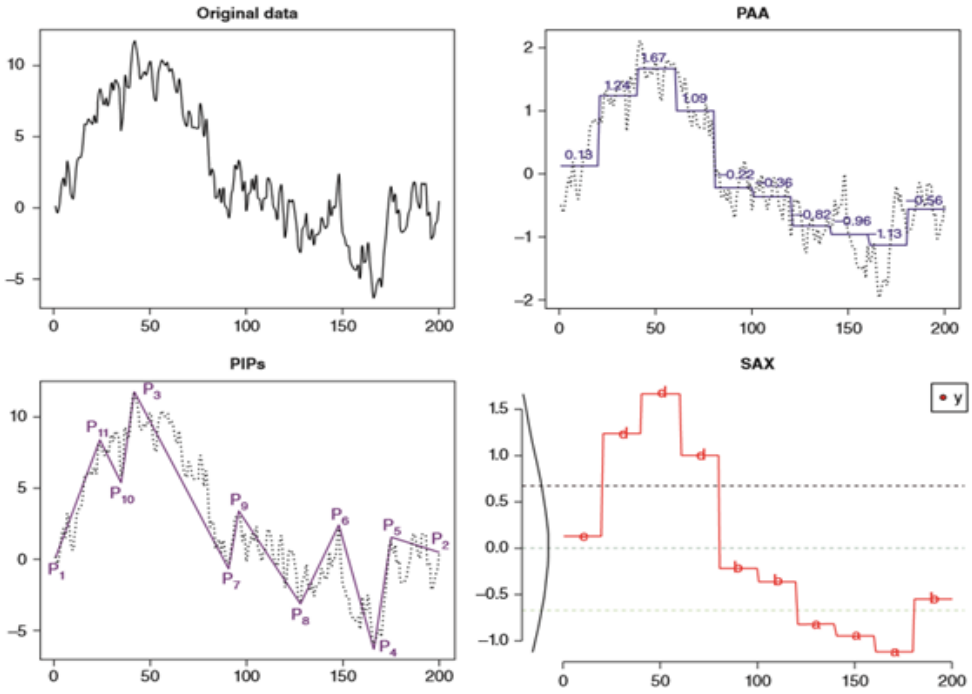

Figure 1.9: Time series representation by PAA, PIPs and SAX [29]

3- Features based representation

Another common alternative to represent time series is to summarize the whole time interval of measurements into interpretable features that quantify best temporal patterns and properties of the data. Extracted features distill temporal dynamics and can provide insights about the generative processes [28]. Finally, global features allow a time-series dataset to be represented as time series (rows) × features (columns) data matrix, which can form the basis of traditional ML methods. The projection of a signal from the time domain into the frequency domain using FT will result in coefficients written using the following form.

Some of the major informative time series features:

- Entropy measures are derived from information theory. Most known entropy measures are Approximate Entropy (ApEn) [32] defined as the logarithmic likelihood that the sequential patterns of the data (of length m) that are close to each other (within a threshold, r), will remain close for the next sample, (m + 1), Sample Entropy which a modification of ApEn [33] and Permutation Entropy (PermEn) [34] which is defined as the Shannon entropy of the distribution of ordinal patterns in the time series.

- Periodicity captures different segments that are periodically repeated through

time series [35].

- Trend indicates the increase and the decrease in the time series over a period of time [36].
- Seasonality detects time series fluctuations occurring over each time period (year, month...). This can provide an insight about regular up and down fluctuations patterns [35].
- Statistical features such as the average or mean, the maximum, the minimum of time series magnitude.

1.4.4.2 Time Series analysis

1. Forecasting is one of the most known time series analysis. It focuses on fitting a statistical model to the data and then simulating it forward in time to make future predictions [28].
2. Query by content refers to the process of searching and retrieving time series data that are similar to a well known and specific pattern of interest in time series database [29].
3. Anomaly detection is the task of detecting abnormal and unusual subsequences in a time series data set [28].
4. Motif discovery is the identification and the detection of commonly repeated and recurrent time series subsequences in database.
5. Clustering refers to the activity of grouping and organizing different time series data into similar groups.
6. Classification refers to the activity of distinguishing and discriminating different labelled classes of time series from each other [28].

Conclusion

This first chapter expounded the rudimentary theoretical concepts recurrent over this thesis. The first part was dedicated to present the decentralized payment systems paradigm and specifically bitcoin currency. Then we presented a literature overview exposing flagship scientific researches about bitcoin payment system.
Finally, over the following sections key concepts involved in the realization of our project such as graph theory, big data, machine learning and time series.
Hereby, these elucidations will be useful to comprehend the project's implementation phases. Therefore the coherent continuation would be expounding the project's design. That's what the next chapter will deal with.

Chapter 2

Proposed approach and solution design

Introduction

The purpose of the present chapter is to outline the general architecture of our solution as well as the various approaches and technical considerations adopted for the development of our system. First, a high level system design is introduced. Then, a detailed description of the developed modules is provided.

2.1 High level system design

As any typical big data analytics solution, the general architecture of our system obeys the analytics systems' common scheme. However, we deployed additional blocks to satisfy our system's requirements. The global system is modular and consists of six main modules:

- Data cleaning block
- User discovery block
- User graph construction block
- Features building block
- Data processing block (Data mining block)
- Time series analysis block

All modules constituting our prototype are expandable to respond to the growing number of bitcoin transactions.
Figure 2.1 summarizes the interaction between these different blocks.

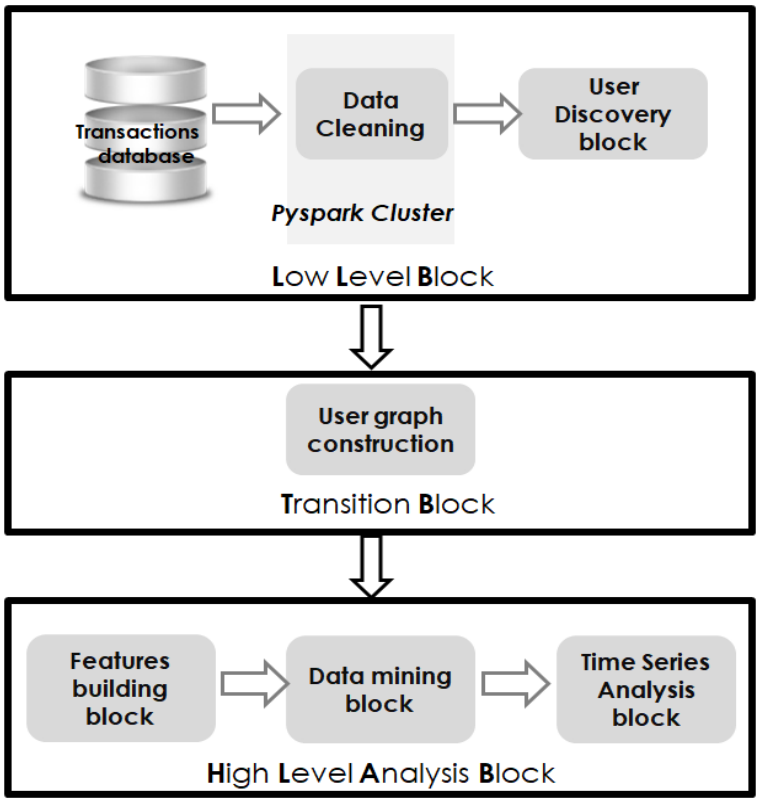

Figure 2.1: High level system design

2.2 Blockchain data cleaning

After the collection of the blockchain dataset which was actually ensured by our partner Thales Group, a cleaning task is performed. This module is implemented in order to ensure the steps below:

- Handle missing fields and remove noise from our dataset.
- Extract important and relevant parts from the transactions' dataset.
- Unify data structure.

2.3 User discovery block

The process of user discovery i.e. discovering the set of addresses belonging to a same user, is one of the most interesting challenges in the bitcoin network analysis. This key step is crucial to go further and study users' behaviours and their wallets' evolution. This task enables gathering addresses controlled by one same user and mainly relies on applying intrinsic properties of bitcoin protocol structure and usage.

At this point, it is necessary to recall what is the address control. As explained in [4], the controller of an address is the entity (or in some exceptional cases multiple entities) that is expected to participate in transactions involving that address. So, considering that transactions are signatures and the knowledge of the private key corresponding to an address is necessary to realize a successful and a valid bitcoin exchange, the address control is associated to the knowledge of the signing private key. However, [4] exposes other cases such as wallet services and bitcoin exchange services where the knowledge of private keys is not sufficient to define address control; for instance, wallet services own all the addresses it generate even though bitcoins detained in these addresses are owned by the wallet services' clients. Hence, the variety of bitcoin uses confirms the exceptional character of this currency as it defines differently the notion of ownership.
In this context, the state of the art provides many heuristics. [4] and [17] present in their work two of the most efficient heuristics used to associate addresses to their appropriate owner.

2.3.1 Heuristic 1: Multi-input transactions

The first heuristic we implement to proceed with users discovery focuses on public keys used in multi-input transactions. In fact, we state that all input addresses of multi-input transaction belong to the same user. This is explained by the fact that using a public key implies the knowledge of its signing private key. So likely, we can presume that these keys are controlled by one same user.

Indeed, the scenario justifying this heuristic seems to be quite logical:
When a bitcoin user controlling multiple addresses, through each of which he had obtained an amount of coins, tries to perform a payment and the payment amount seems to be higher than bitcoin values available in each of the controlled addresses, the user will utilize a subset of his addresses to aggregate bitcoin amounts corresponding to the payment cost. Seemingly, this manner of transacting is widely spread through the bitcoin network as it avoids proceeding with multiple transactions which can induce high transactions fees and it therefore confirms the safety of this first heuristic [17].
Besides, the importance of this heuristic is meaningfully highlighted when observing specifically its transitive effect at a large scale. The next figure (2.2) illustrates the first heuristic.

If we consider many multi-input transactions in the bitcoin graph: Note A and B as inputs of one first transaction and tag another transaction with B and C as input addresses:
***Then, we can safely assume that A, B and C belong to the same user* [4].**

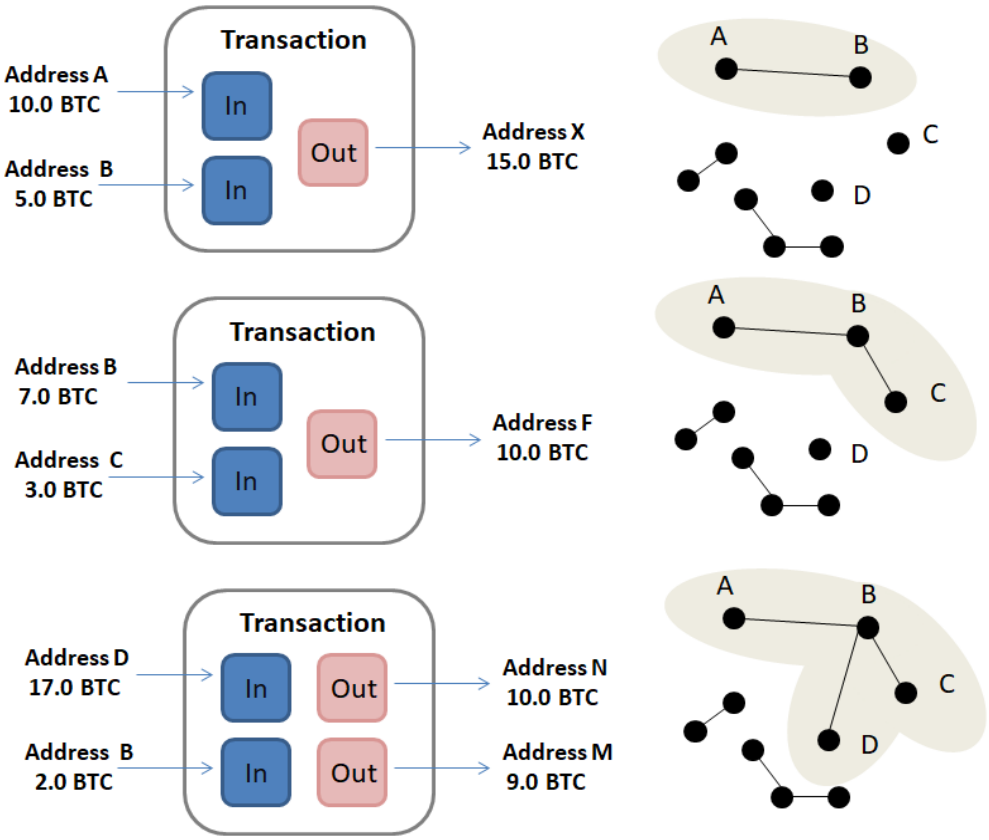

Figure 2.2: The process of linking input addresses to its controller

This property enables proceeding with partial but a crucial step of user discovery and getting an increasingly precise graph of transactions, the one illustrating exchanges between users rather than addresses. Using notions about bitcoin transactions and addresses graphs, we can summarize the first heuristic formally as below:

Let t1 be a multi-input transaction.
Then, let inputs(t1) be the input set containing all addresses involved as inputs of this transaction;
So, we assume that all pk(t) ∈ inputs(t1) are controlled by the same user.
If t2 is another multi-input transaction and we consider inputs(t2) as the set of all input addresses.
If inputs(t1) ∩ inputs(t2) ≠ 0
Then we can affirm that the addresses set of inputs(t1) ∪ inputs(t2) is controlled by the same user.

2.3.2 Heuristic 2: Change addresses

We have seen in the previous section how heuristic 1 can associate addresses in input of a transaction to its controller. Despite its role in unanonymizing bitcoin users, applying only this first heuristic is not sufficient to reconstitute a fully accurate scheme of users' transactions.
The second step of user discovery implemented in our work relies actually on another intrinsic property of bitcoin protocol. In fact, Bitcoins cannot be divided only when spent [4]: in another words; users cannot proceed with partial spending; if they perform payments where input amounts exceed the payment value, they have to send back to themselves the difference between the sum of input values and bitcoin amount to pay (see figure 2.3).

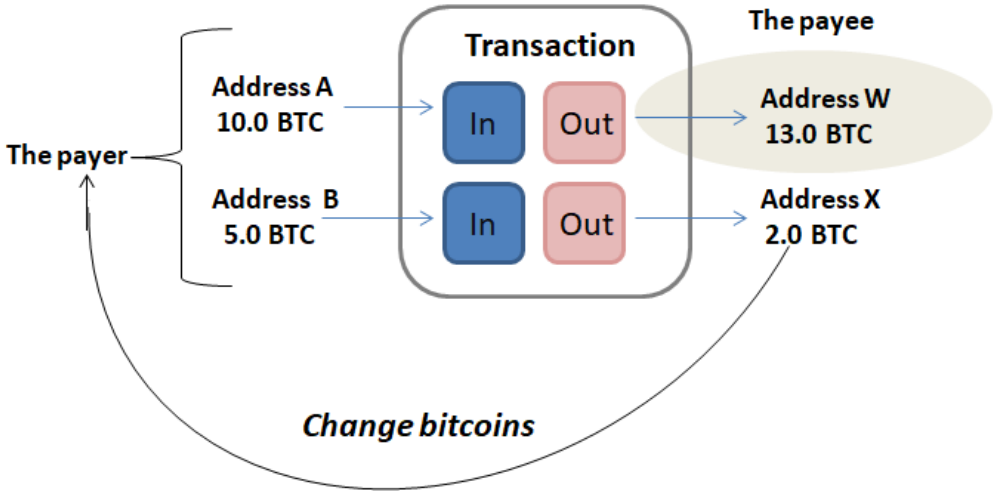

Figure 2.3: Identification of one-time change address

In order to improve the user discovery process, we need to associate input addresses with one âor severalâ output addresses that belong to the same user, these addresses are called change addresses or shadow addresses as mentioned in [17].

Back to literature, several authors have proposed different versions to discover which output address plays the role of a change address in each transaction. In fact, among all existing versions associated to this heuristic, the most efficient one is mainly searching for one-time change address. Indeed, this is explained by the fact that it is a widely common practice to receive the change on a newly generated address, that will probably never be reused [4]. Consequently, this heuristic can be used to improve addresses clustering and add to the set of already involved addresses in multi-input transactions, the one-time change addresses for each user.

We use in our work an improved version of heuristic 2 as it was introduced in [17]. It can be formally presented as follow :

First, we should consider the transaction dataset ordered chronologically.
Let t be a transaction :
Let inputs(t) and outputs(t) be the sets of input addresses and output addresses involved in the transaction.
If t is not a mining transaction and outputs(t) contains only one address appearing for the first time.
If user in input is not the same user in output,
i.e there is no pk_out ∈ output(t) belonging to addresses set of input user.
Then, the address appearing for the first time is actually one-time change address detained by the input user.

So far, we illustrated in this section one of the key modules of our solution. It certainly does not break entirely the anonymity ensured by the bitcoin protocol as the implemented process does not trace public keys to real-world identities but it definitely helps a better investigation of users' behaviours as it collects almost all the keys associated to users' activity.

2.4 User graph construction

After performing an initial clustering of public keys, we defined new entities which are users controlling one or a huge set of addresses to transact through the bitcoin network. Even though users are still anonymous and addresses clustering result is not ultimate, the transition from address-to-address level to user-to-user level is quite interesting; we can carry out now a true relational scheme between tangible and real entities. This step consists mainly in rewriting the whole bitcoin transaction history at a user-to-user level; the process consists of considering each transaction as well as the set of addresses in input and those in output, fetching in the user discovery output about the user corresponding to each address and finally note the new transactions' record. The attractiveness of this graph resides actually on the relevant overlap of increments and superposed layers we can construct over it.
Finally, we should avow that drawing and visualizing users' transaction graph is quite interesting as it helps understand how interactions occur, detect the most central nodes and define where the most important bitcoin flows circulate.

2.5 Features building block

The idea behind this module is building descriptive variables outlining and describing users' activity and behaviour in the bitcoin transactional system.
Actually, these features are the result of translating the bitcoin exchanges pattern into a social graph. Indeed, extracted variables will help quantifying users' activity and will serve later as input for the data mining task. So, at this level, users will be called observations. Each feature is called variable and represents a vector component of an N-dimensional space where N is the total number of variables. Figure 2.4 highlights the mathematical and formal representation of our dataset.

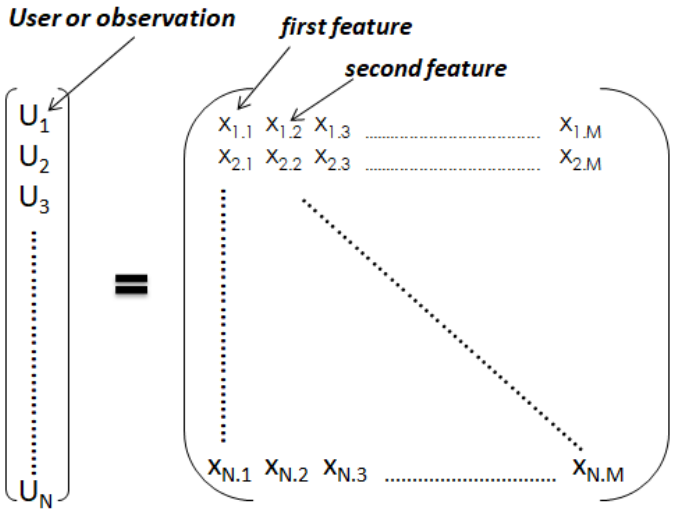

Figure 2.4: Features' matrix

Since studying the bitcoin financial system and analyzing the user-to-user transaction flow can be inspected from a social and a graph modelling perspectives, we chose to combine both and extract a panoply of features to surround in the most efficient manner all the indicative and significant aspects of users' activities.

2.5.1 Graph metrics

This part is based on users' transaction graph; so we will consider a directed graph where the source node (S) refers to the sender and the reception node (R) represents the receiver. The edge connecting these nodes represents the transaction operation between (S) and (R) and can be also described by an attributed weight which represents in this case the transaction value.

As mentioned in [22], many graph metrics can be of great interest in revealing users or nodes importance in a graph. We chose to implement in this module, the following five metrics detailed in section 2.3:

1. ***Indegree***: The indegree of a vertex is the number of incoming edges connected to it. In users' transaction graph, this feature represents the number of incoming transactions or the number of payments the bitcoin user had received.

2. ***Outdegree***: The outdegree of a node is the number of outgoing edges connected to it. In users' transaction graph, this feature represents the number of outgoing transactions or the number of payments the bitcoin user did.

3. ***Closeness Centrality***: This metric tells about the mean distance from a vertex to other vertices. Since vertices are here users, distances are calculated on the basis of bitcoin payments between users.

4. ***Betweenness Centrality***: This measure calculates the number of shortest paths passing through each node.

5. ***Clustering coefficient***: This feature measures the average probability that two neighbours of a same node are themselves neighbours. Consequently, this metric calculates an approximate average of whether two bitcoin users who transacted with one same common bitcoin client had already transacted with each other.

2.5.2 Behavioural features

Like graph metric variables, behavioural features can bring up meaningful insights about users' comportments. We relied in this part of features extraction on the exhaustive set of variable implemented in [19] and we considered mainly eighteen features we explain below:

1. ***Number of public keys owned by a user***: This feature refers to the number of public addresses or keys a user is controlling in the bitcoin network.

2. ***Number of mining transactions***: The number of mining operations the user performed.

3. ***Bitcoins owned through mining***: Amounts of bitcoins a user gained through mining.

4. ***Number of transactions between same user addresses***: The number of exchanges between the same user addresses.

5. ***Bitcoin amounts sent between same user addresses***: Sum of bitcoin amounts transferred between the same user addresses.

6. ***Average of bitcoin sent***: Average of bitcoin amounts sent to other users. We consider in our calculation only external exchanges and eliminate transactions between the same user addresses.
7. ***Average of bitcoin received***: mean of bitcoin amounts received by a user.
8. ***Maximum of bitcoin sent.***
9. ***Maximum of bitcoin received***.
10. ***Minimum of bitcoin sent***.
11. ***Minimum of bitcoin received***.
12. ***Average time interval between in-transactions***: The average time interval between transactions where the user receives bitcoin amounts.
13. ***Average time interval between out-transactions***: The average time interval between transactions where the user sends bitcoin amounts.
14. ***Average in-transactions***: The average of bitcoins the user receives.
15. ***Average out-transactions***: The average of bitcoins amounts the user sends.
16. ***Mean of degrees of users with whom a specific user has transacted***: This feature reveals the average of degrees of one user's neighbours (average of in and out transactions of one user's neighbours).
17. ***Variance of degrees of users with whom w a specific user has transacted***: This feature refers to the variance of degrees of user neighbours (variance of in and out transactions of one user's neighbours).
18. ***Life duration***: This feature refers to the time interval during which the user has been active on bitcoin network.

Before moving to the next step, we precede the data mining task by an analysis of extracted features.

2.6 Data mining block

With the purpose of extracting hidden patterns from our cleaned, transformed and prepared data, we conceived this module to effectively disclose meaningful insights about bitcoin users' activities. Its implementation is preceded by a crucial step which is selecting the task and algorithms to apply. To do so, we should at this point recall the main goal of this survey which is identifying the pattern of fraudulent and illegal transactions and outline illicit bitcoin users profiles.

When exploring previous researches [18], [19] and [20], mentioned earlier in section 1.3, which tried to address this same issue, we found that they all dealt with it using outliers detection approaches. All these trials were quite interesting as they parsed bitcoin data which is a tough task in itself due to the arguably significant complexity of this transactional data. However, they all succeeded in detecting some suspicious activities but hadn't led to a detailed pattern specification and identification of every fraud type occurring in the bitcoin network.
One of the reasons that led to the imperfection and incompleteness of these surveys is the fact that they were all conducted using unlabeled data as bitcoin users are anonymous and no evidence can tie a bitcoin client to a real-world identity. This in itself is one of the biggest challenges in ML field due to the absence of an obvious and a universally agreed upon testing approach in this case [18].
When further analyzing the causes of these surveys inefficiency, we discovered that outliers detection paradigm and the process of identifying irregularities that deviate from the rest of the data patterns in transactional networks rely mainly on the assumption that legitimate behaviours define the standard and the ordinary state of the network and that illegal activities will appear as a significantly distinct intruder behaviours [36].

So, applying outliers detection approaches to bitcoin data implies consequently that the standard routine of bitcoin network is carrying out normal and legal purchases, which is not the case. In fact, many surveys such as [4] declare that mining actions, online games purchasing and exchange services corner about 95% of bitcoin network traffic. Considering all facts outlined in chapter 1 section 1.2.3 we do believe that outliers detection paradigm may not be the most suitable approach to identify frauds pattern. Though, we suggested in our study a different approach to deal with frauds detection. Considering that illegal activities are the most widespread actions, we conclude that it is important as a first step to identify activities' types as well as users and propose to perform clustering to identify different users' profiles. As a consequence, we resolved to clustering-based algorithms to proceed with users profiles identification. Finally, as we have no prior knowledge about the number of existing communities and the validity of our result, we will be testing, as a next step, our implemented system with [4]'s labelled data in order to confirm our findings. Finally, reporting and data visualization are key components of our data analytics process as they enable grasping what the processed data have to tell. We will be visualizing results of the data mining task.

2.7 Time series analysis block

The final step of our work consists of investigating and inspecting the activity and wallets' evolution of identified users' profiles from a dynamical perspective as the bitcoin network structure and users' behaviours are perpetually evolving over time. In order to study time series data, we tried to extract meaningful features which will

serve later as input for ML models and algorithms. We focused mainly, as a first step, on simple and interpretable statistical features and abandoned shape-based features , namely:

1. ***Maximum of time series:*** This feature indicates the maximum of bitcoin amounts a user had on his public addresses.

2. ***Minimum of time series:*** This attribute indicates the lowest bitcoin amount a user had during his activity on bitcoin network.

3. ***Mean of time series:*** This feature measures the average of bitcoin amounts a bitcoin client has using the aggregation of all his public addresses' accounts during his activity period.

4. ***Standard deviation:*** This measure is used to quantify the amount of variation or dispersion of bitcoin amounts a user had compared to the mean value.

5. ***Shannon entropy:*** This feature refers to the amount of disorder or uncertainty in a time series data. In other words, time series entropy measures the quantity of information or surprisal of the sequential data. Time series with low-probability value will carry more information than more predictable time series with a high-probability value.

6. ***Number of peaks:*** This feature measures sudden and unexpected dynamics or wallet's changes through users' transactions.

7. ***Average duration between peaks:*** This attribute quantifies the average duration between sudden increases of bitcoin amounts each user controls. This duration is measured in hours.

8. ***Duration between nearest peaks:*** This item measures the lowest duration between two sudden increases of bitcoin amount a user holds. This duration is measured in hours.

9. ***Duration between farthest peaks:*** This item measures the longest duration between two sudden increases of bitcoin amount a user holds. This duration is measured in hours.

10. ***10 most prominent frequencies:*** These features quantify the dynamics of time series data. High prominent frequencies indicate how bitcoin user's transactions go rapidly and how bitcoin exchanges' amounts vary significantly. However, low prominent frequencies inform about transactions monotony and the low variance of bitcoin amounts exchanged.

All these features will be tested on [4]'s labelled users to respond to this relevant question: How far can dynamical features extracted from users' time series represent and classify bitcoin network key components ?

Conclusion

This third chapter was allotted to put the light on the proposed solution's model. Therefore, as we have an explicit design of our prototype, we can continue to delineate its implementation steps as well as the experimental results on the fourth chapter.

Chapter 3

System implementation and experimental results

Introduction

In this chapter, we move to the implementation of our solution. We will expose the development environment details, the description of the implemented modules and finally experimental results and interpretations.

3.1 Setup of work environment

Our work was realized using a server, a laptop, linux operating system and a set of softwares. In this section, we will be briefing about the deployment and the configuration of our development environment including hardware and software tools.

3.1.1 Hardware work environment

The hardware tools on which we have implemented this project are a server machine and a laptop. We detail below the laptop main characteristics:

- Disk: 1 Terabyte.
- Memory: 16 G.
- Processor: Intel Core i7-6600U.
- Operating System: Linux.

Considering the computational complexity of our work, we used this machine to work locally and access the lab's remote server using SSH sessions in order to carry out intensively complicated calculations (see figure 3.1).
The server's main characteristics are:

- Disk: 5 Terabytes.
- Memory: 64 Gigabytes (Go).
- Processors: 4 CPUs, each one is an Intel Xeon Quad-Core 2.93 GHz, which makes a total of 16 computing cores.
- Operating System: Linux.

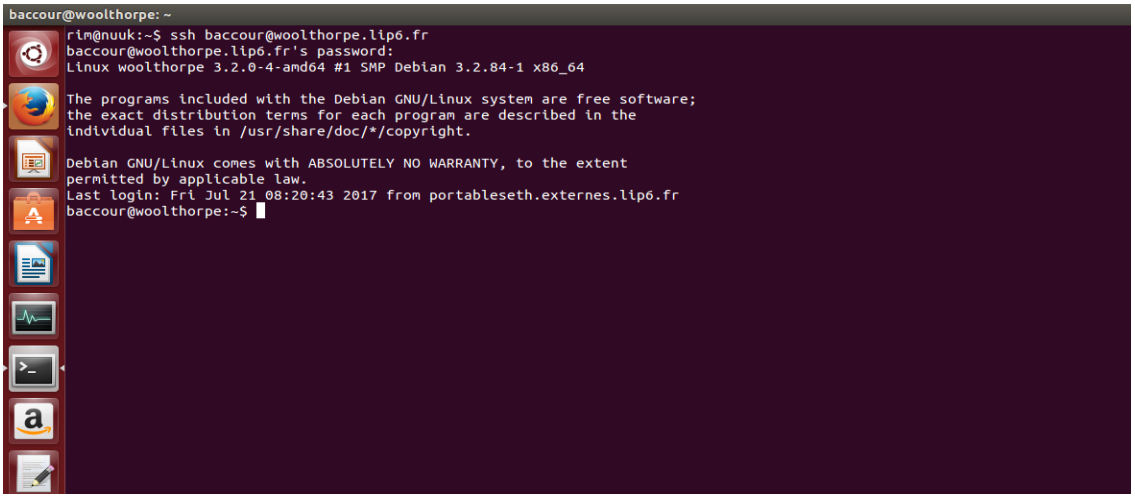

Figure 3.1: Secure Shell to access Woolthrope

3.1.2 Software work environment

During this project, a set of tools softwares were used in its different stages. In the upcoming paragraphs, we provide a description of these tools.

Gephi: is an open source and free software for all kind of networks (dynamical, static, small world, directed, undirected...) and especially large networks exploration and visualization. It offers a variety of filtering, clustering and modelling options. It offers also a framework to extract and compute meaningful network features such as centrality metrics, degrees...[37]. This helped us visualize transactions, addresses and users graph.

PyCharm: is an Integrated Development Environment (IDE) used in programming, specifically for python language. It offers code analysis and specifically graphical debugger. We chose this development environment as it is a cross-platform that can be deployed on many operating systems and offers a well documented framework.

Python: is a high-level programming language written in C and whose design philosophy emphasizes on code readability. Scripts are first compiled or translated to byte code than interpreted by the python interpreter. All scripts were mainly written with python as it offers a variety of interesting packages for graphs processing, machine learning and data visualization.

NetworkX : This python package is dedicated to graph creation, manipulation and analysis. It is dedicated for studying the structure and dynamics of complex networks. It offers a set of flexible functions conceived to perform calculations which

characterize graphs such as distances between nodes, centrality measures, degree metrics and connected components. We used this package to benefit from its simple functionalities.

Snap: is a high performance python library for large networks analysis. Snap was developed by a group of Stanford researchers with C++ programming language. It was optimized for maximum performance and compact graph representation to process large graphs with millions of nodes and billions of edges. This tool helped us manipulate the huge users transaction graph [38].

Scikit learn: is an open source python library containing various methods for data mining and data analysis tasks. We used this library to perform all ML related tasks.

PySpark: represents the spark python API. This big data framework actually converts the Spark programming model initially based on scala to python and enables batch and streaming data processing.

Our choice of PySpark rather than Hadoop for performing big data processing tasks is based on a benchmark study summarized in the following table.

Table 3.1: Benchmark study of Spark and Hadoop

	Spark	**Hadoop**
Speed	×	
Ease of use and presence of high level APIs	×	
Recovery and fault tolerance	×	×
Large-scale graph processing packages	×	

PySpark framework is more adapted to our needs as it is compatible with our initial programming language and is able to process streaming data as we intend to evolve later our prototype for real-time blockchain data processing.

GraphX: is Apache Spark library for graphs and graph-parallel computation.

PySpark setup

The installation of PySpark requires :

1. Installing java and python as well as python interpreter.

2. Downloading the chosen version of Apache Spark (2.1.0 in our case).

3. Creating directory for Spark applications.

4. Adding environment variables to .bashrc file including Py4J python package which helps python interpreter to dynamically access the Spark object from the JVM (see figure 3.2).

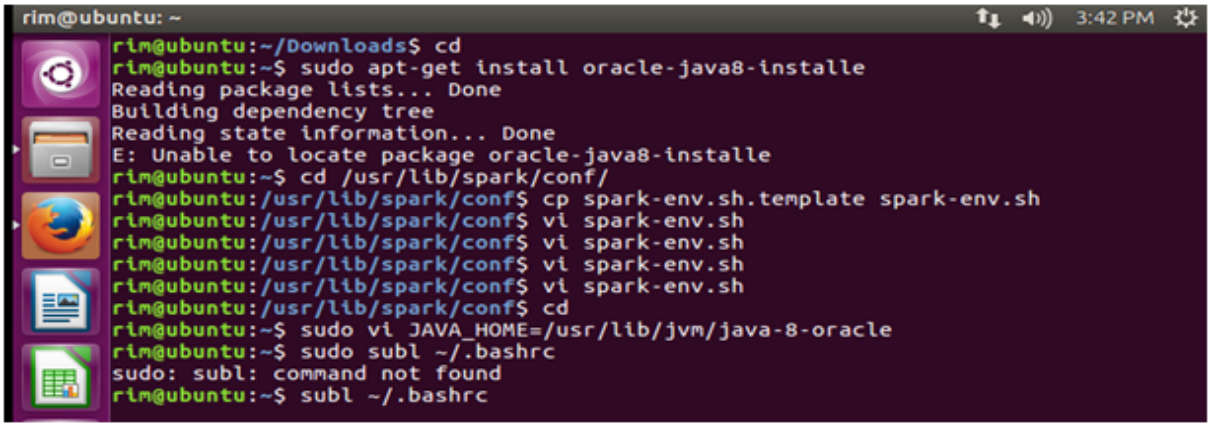

Figure 3.2: Access .bashrc file for adding environment variables

5. Installing the Simple Building Tool (SBT) which is used to build Spark.

6. Accessing finally the PySpark interactive shell using ./bin/pyspark command line.

3.2 Implemented modules

Before discussing the main implementation stages, it is necessary to mention that the project is actually divided in two main steps. In fact, we first started working on a small dataset containing 757 714 transactions and covering a time interval of two years and a half from January 2009 to June 2011. This dataset is composed of 457 949 exchanges between public keys, 129 765 mining transactions and involving 621 428 unique addresses. In this first step, we needn't a big data framework to process data. All scripts were written with python. Then, we just got transactions that took place from 12/01/2009 to 10/08/2015. As expected, the number of transactions increased exponentially to reach 78 762 324. We can thus identify 78 394 505 exchanges between 88 171 000 public keys and 367 819 mining operations. The following figure 3.3 shows the rapid increase of bitcoin transactions' number. At this step, cleaning and processing such a huge dataset was not feasible with simple python scripts. We needed to upscale our algorithms to process this big transaction set using a big data framework, which is in our case Pyspark.

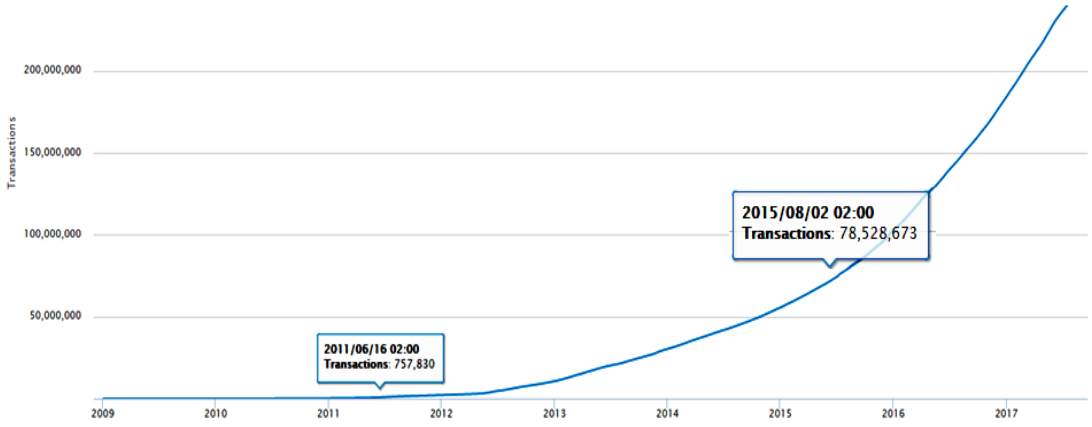

Figure 3.3: Evolution of the total number of transactions [10]

All along this chapter, we will present, discuss and justify our choices of the tested technological solutions in both stages.

3.2.1 Data cleaning block

Before enumerating different data cleaning tasks implemented throughout the project, it is necessary to provide first a detailed description of our target dataset to ensure good assimilation of the cleaning and the other system modules processes.

3.2.1.1 Data set description

Each transaction is composed of six information blocks:

- transaction Identifier (txid): refers to the suite of characters identifying the transaction in the blockchain.
- timestamp: corresponds to the exact date and time at which the block containing the transaction was validated by nodes in the bitcoin network.
- exchange rate : refers to the price for which bitcoins can be exchanged for another fiat currency at that timestamp.
- total value: indicates the bitcoin amount transferred throughout the transaction.
- transaction inputs: is a list of N input elements with N the number of involved input addresses, each element consists of four main components:

  1. the public key identifier (address)

2. the hash of the previous transaction (hashPrevOut) through which the input public key obtained the bitcoin amount it tries to spend through the present transaction.
3. the output index (indexPrevOut) referring to the exact indicator pointing at the bitcoin transfer operation of the previous transaction used by the input public key in the current transaction.
4. value or the bitcoin amount to send which is equal to the bitcoin gained in the previous transaction.

- transaction outputs: is a list of M output elements with M is the number of receiving addresses, each element consists of three main components:

1. the output index (indexOut).
2. the receiving public key identifier (address).
3. the value which is the bitcoin amounts transferred to the considered output address.

These transactions represent bitcoin exchanges between public keys. The total value is always equal to the sum of bitcoin amounts transferred from input to output addresses. Furthermore, the bitcoin record contains mining activities which can also be represented as transactions with no input and an output which corresponds to the total retribution for solving a block (the current bitcoin reward and the sum of transaction fees of all transactions).

3.2.1.2 Implementation

As we worked first on the first sample with 757 714 transactions, filtering the data was feasible with python scripts. We could simply browse chronologically ordered transactions and get involved input and output addresses. Later, when moving to 2009 to 2015 transactions, handling addresses which lengths vary from 27 to 35 characters was a rough task which demands a huge computation power and a great memory capacity. At this stage, we made use of PySpark and implemented the following steps:

1. Adapt mining operations to bitcoin exchanges format by adding a fictive input address Id which is 0 referring to the source of all mining operations and transaction fees as well as this following block $\{hashPrevOut = 0, IndexPrevOut = 0\}$. These transformations helped us fasten our algorithms as they helped unify transactions format. We still though differentiate mining operations as address Id 0 doesn't exist in the original transaction dataset.

2. Order chronologically exchanges and mining operations.

3. Get an overhead set of all addresses involved in all transactions and associate a unique integer Id to each address. Building this intermediate dictionary was helpful because we will use from now on integers which are encoded on 32 bits rather than huge number of long strings, each of them requires at least $27 \times 128 = 3546$ bits to be encoded in memory. Thus, this intermediation helps reduce largely memory space required to manipulate addresses.

4. Get input addresses Ids involved in all transactions. This result will serve as input for the next bloc.

5. Establishing edges between input addresses for each transaction.

3.2.2 User discovery process

This section details the overall user discovery process.

3.2.2.1 First heuristic

As explained in section 2.3.1, this heuristic is based on multi-input transactions.
It gathers all addresses serving as simultaneous inputs in these transactions and associates it to one same user.

Implementation

As we worked first on a small sample of transactions, we used at this level, NetworkX python library for graph processing. In a bid to ensure this initial addresses clustering, we perform the calculation of connected components.
In fact, this operation consists in creating an undirected graph where transactions' input addresses are vertices and edges link addresses figuring in the same multi-input transaction. This functionality takes full advantage of transitivity to concatenate all addresses of the same user when browsing all the transactions record. Connected components are consequently induced sub-graphs, each of them corresponds to the set of one user addresses. The next step was trying to apply this same functionality of NetworkX on our second huge graph. Our trials totally failed due to NetworkX 's method for graph storage which makes it unsuitable for processing massive networks with billions of nodes and edges.
In fact, all graph objects including nodes and edges are represented in NetworkX with hash tables called dictionaries in python which require large memory space. This offers maximum of flexibility for the library users at the expense of performance as it imposes performance overhead in terms of a slower speed and a larger memory footprint than the other alternative representations.

Our next technological choice failed also in processing such a large graph. Computing connected components with Spark build on top package for graph processing which is GraphX was unsuccessful. When fetching about job abortion causes with

Spark, we learned that processing 150 Gigabytes with graph frames of GraphX on standalone mode is not feasible and that Spark's high level packages still suffer from bugs and instability problems.

Results and validation

The output of this first heuristic is a CSV file containing three columns. We register randomly attributed but unique IDs for users, the set of addresses IDs owned by each user and the count of these addresses. User discovery's first step enabled us reduce the graph size; the output holds 35 762 738 clusters of users from an initial dataset containing 88 171 000 public keys.
Since, there is no ground truth data available to validate our findings, we contacted [4]'s researchers and obtained the labelled transaction and user sets gathered in their survey. To verify the consistency of our algorithms, we considered the set of transactions corresponding to the starting and ending dates of [4]'s survey and applied our first heuristic to find the same number of users identified in [4] which is equal to 6 595 564 distinct users. This test proved the uniformity of our algorithms and results with other surveys. To go further in the validation process, we computed five indicators detailed in Appendix B and which are:

- Precision
- Recall
- F1 measure (harmonic mean of precision and recall)
- NMI (Normalized Mutual Information)
- aNMI (adjusted Normalized Mutual Information)

These measures compare two clusterings ([4]'s ground truth clustering and our first heuristic output). If our first heuristic output uniformly tends to group all addresses belonging to a same user and results are therefore identical in both clusterings then Recall=Precision =1 and F1 measure tends towards 1 as well.
Similarly, NMI measure will increase as the mutual dependence between both results increases and decrease if both clusterings do not share mutual information.
However, if we identify users, which are actually equivalent to connected components, with only a subset of their controlled addresses and other distinct connected components possessing the rest of addresses belonging to the first same user then this does not affect Recall but actually decreases the Precision; we conclude that the clustering does not contain errors, it just delivers an incomplete result.
Finally, if test clustering tends to merge clusters of addresses belonging to different users and identified it as one connected component then this output will decrease Recall as it delivers "real" errors and not just "forgetfulness".
The following table provides an insight about recall, precision, F1, NMI and aNMI measures calculated upon our first output.

Table 3.2: Measures evaluating the first heuristic output

Measures	Precision	Recall	F1	NMI	aNMI
First heuristic	0.98	0.77	0.86	0.89	0.65

As expected, Precision score for this heuristic is very high which means that when two addresses are grouped together, they nearly always belong to the same user. Both clusterings deliver identical results with same number of connected components and same gathering of addresses. By manually investigating one of the errors, we found that addresses mistakenly grouped together belonged to two companies, one being in fact a branch of the other. Surprisingly, Recall score is already high for this heuristic which indicates that our clustering includes several connected components pointing to one address which totally true.

Limits

The first heuristic of user discovery process presents promising results as it collapses efficiently addresses' graph as shown in table 3.2. So far, the main limitation is that this first heuristic works under the assumption that owners do not share their private keys. However, this intrinsic property doesn't hold anymore in bitcoin network with the continuous change and mutation of bitcoin usages and protocol structures.
Added to that, this heuristic doesn't identify totally users. So, its final output CSV file contains imaginary bitcoin clients and needs to be refined by further carrying out user addresses identification.
This will lead us to move to the second heuristic implementation.

3.2.2.2 Second heuristic

Although heuristic 1 already yields a useful clustering of users, we chose to further collapse users, using the second heuristic. Detailed earlier in section 2.3.2, heuristic 2 focuses on the mechanism through which users tend to get the rest or their money back as bitcoin amounts to be paid are not always equal to bitcoin amounts held in the payer public addresses.

Implementation

First, we applied our improved version of the second heuristic algorithm on our first target dataset: the two year transaction sample. This algorithm helped us refine once again users identification: We passed from a set of 455 332 users to 292 986 users. The next step was to upscale this process on our big dataset. The process involves the whole transaction history (transactions occurring from 2009 to 2015), the output of the first heuristic and the intermediate dictionary constituted of addresses and their integer IDs.

Before starting the computation process, we tried to take a look at inputs' sizes as shown in table 3.3.

Table 3.3: Second heuristic input files' size

Inputs	Size (Go)
Transactions history	120
CSV file (output of the first heuristic)	1.3
Dictionary of addresses and associated integer IDs	4

Considering the computational complexity of the algorithm as well as the huge memory space required to process all input data, we tried to estimate approximately time duration needed for this algorithm execution and presumed that it will take us about three weeks with an entire monopolisation of the lab server computational resources. Consequently, we decided to limit transactions time interval to the duration covered by [4]'s transaction dataset and apply our algorithm to decide after whether it is worth trying on the whole dataset or not.

Results

When applying the second heuristic algorithm on transactions from 2009 to 2013, we got a CSV file containing less users than the first heuristic output; 6 595 564 users passed to 4 384 687 users which is not identical to [4] finding. This can be explained by the fact that we used a different code as we tried to improve the algorithm of this second heuristic to save the refinement effort accomplished by [4]'s researchers. The following figure shows an example of identified addresses of Mt'GoX, a well known bitcoin exchange plateform, using heuristics 1 and 2. Blue edges link addresses used as inputs of same transactions and Red edges correspond to links between change addresses. To further evaluate our findings and decide whether to conduct or abandon this highly complex algorithm on 35 762 738 users, we once again proceeded to the computation of measures evaluating this heuristic output.

Table 3.4: Measures evaluating the second heuristic output

Measures	Precision	Recall	F1	NMI	aNMI
Second heuristic	0.09	0.83	0.16	0.47	0.15

By adopting this second heuristic which creates larger clusters, Recall score raises as shown in table 3.4. It actually regroups more addresses together, however, this gain was made at the cost of an important decline in Precision resulting in lower overall scores of F1, NMI and aNMI. Considering our lack of a labelled dataset up

to 2015 which could helped us identify even a small fraction of true positives (TP) and remove manually false super-clusters corresponding to different users and identified as one same user, we had finally convinced ourselves of both the uncertainty of Heuristic 2 and its ineffectiveness attested by the very low indicators computed in table 3.4.
All these reasons, added to the huge resources required to execute this algorithm on users set identified earlier after the first heuristic, we decided to abandon this second heuristic and consider only the output of the first one.

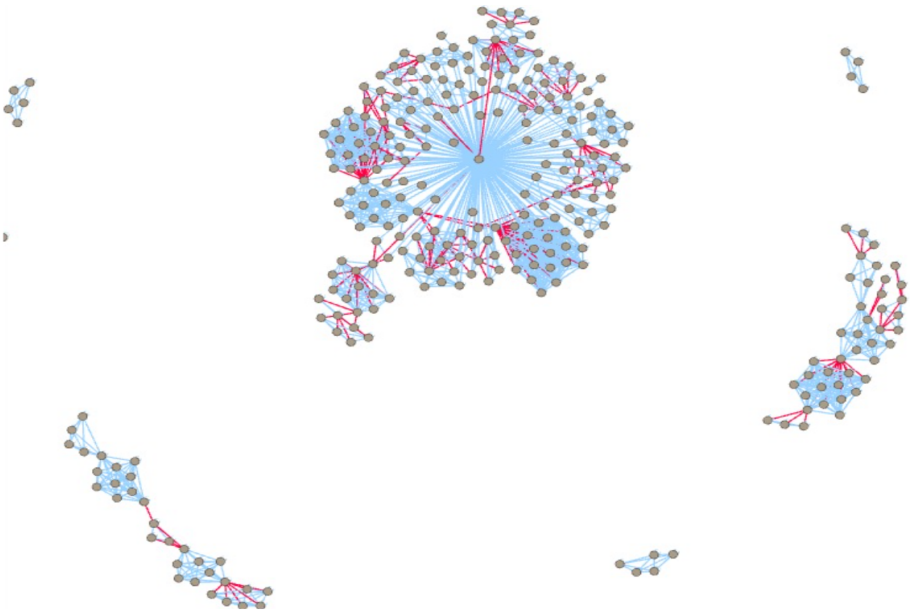

Figure 3.4: Linking Mt'Gox addresses

Limits

The main limitation of this heuristic is the fact that it takes advantage of an idiom of use, rather than an inherent property of the bitcoin structural protocol as heuristic 1 does. It consequently does lack robustness in the face of changing (or adversarial) patterns in the network.

3.2.3 User graph construction

The transition from address-to-address level to user-to-user level despite the potential anonymity of bitcoin users remains quite interesting.
Rewriting the whole transaction history at user-to-user level helped us obtain a CSV containing the timestamp, the sender ID, the receiver ID and amounts of bitcoins exchanged. The following figure shows a subgraph of bitcoin users' exhanges.

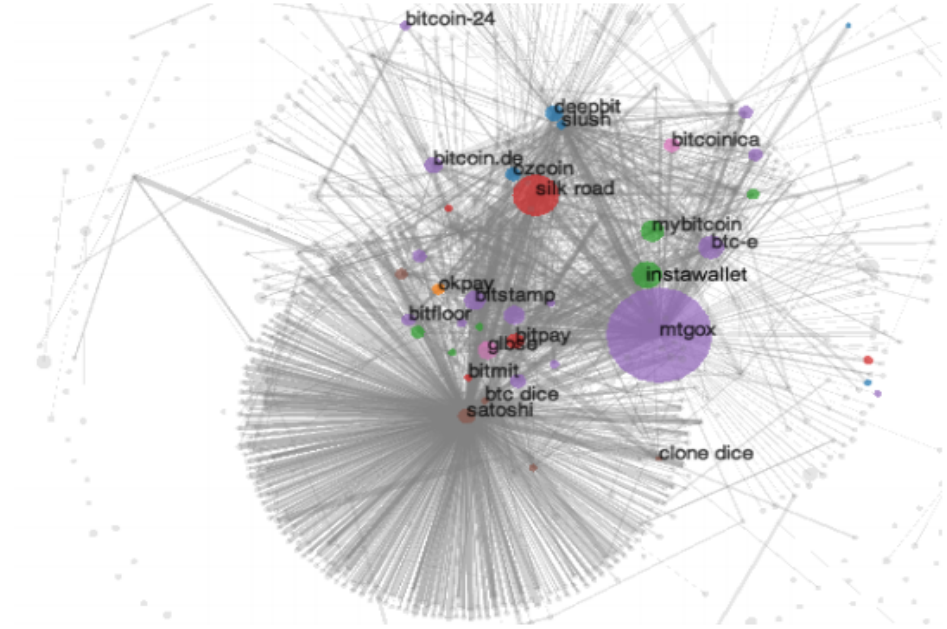

Figure 3.5: Users' exchanges graph

3.2.4 Features Building

This section is about extracting behavioural and graph related features which will help us shed considerable light on the bitcoin users activity.
The calculation of features related to the two year users of bitcoin network was quite simple and algorithms run locally on the laptop and delivered a rapid output.
When moving to features of 35 762 738 identified users, things are getting rougher. As to the issue of computing graph metrics, we consider at this level transactions between users containing about 35 762 738 vertices and 44 194 300 edges.
Referring to table 1.2, we presumed that calculation of centralities in such a huge graph will take years to be done. Considering resources limitation, we decided to abandon complex graph features and work only with degree centralities along with behavioural features which are relatively easy to compute.

3.2.5 Data mining task

The data mining task consists of three principal phases:

1. Data exploration
2. Features selection
3. Model implementation & result visualizations

3.2.5.1 Data exploration

The first step of data exploration was to visualize statistical distribution of extracted features. As expected, they all fit the power law distribution since they describe dynamics which are users activities over a network that evolve over time. We chose to represent in the next figure (3.6) the log-log distribution of the outdegree.

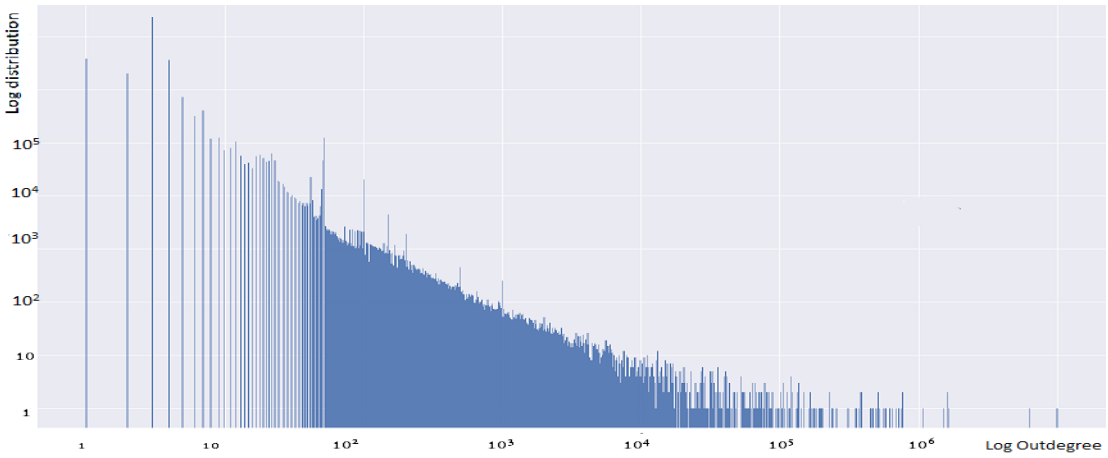

Figure 3.6: Log-log of statistical distribution of the outdegree

As K-Means like all ML clustering-based algorithms are sensitive to differences in scale of selected input variables, we opted to proceed with a useful pre-processing phase which is features scaling. This practice reduces the effect of those features with broad ranges of values on cluster assignment. In fact, after visualizing statistical distributions of features and a benchmark study of different scaling methods detailed in Appendix C, we eliminated the most common scaling method which is the standard deviation technique as it stands only for input data that fit the standard normal distribution and opted only for the following two methods:

1. Min-Max scaling
2. Log min-max scaling

The next step was applying the Principal Component Analysis (PCA) method to decide about the most suitable scaling method describing best relations between our data set variables. Graphical output of PCA includes variables correlation circle and histogram delivering percentages of variance explained by each retained dimension. The following three figures summarize the results of applying PCA on datasets resulting from the two scaling methods (see figures 3.7 to 3.9).
The factorial plan is made up of the two first dimensions which explain more than 90 % of the total inertia for the first data set and a little less, about 65 % of the total inertia for data on which we applied log min-max scaling (see figure 3.7).

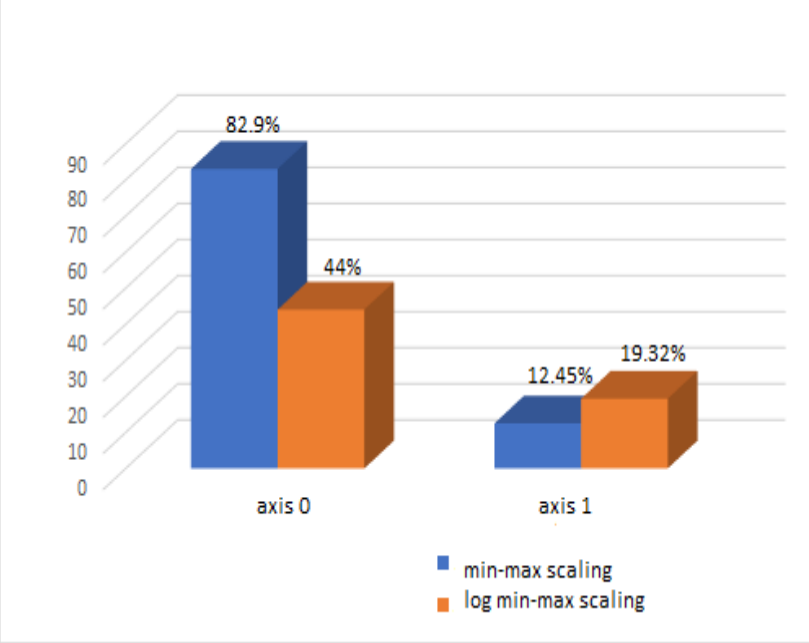

Figure 3.7: Percentages of variance explained per component

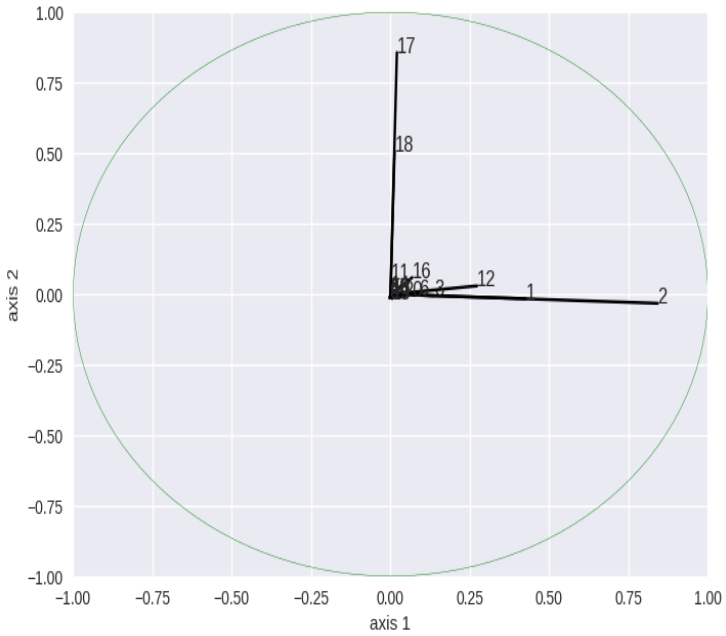

Figure 3.8: Circle of features correlation [Min-Max scaling]

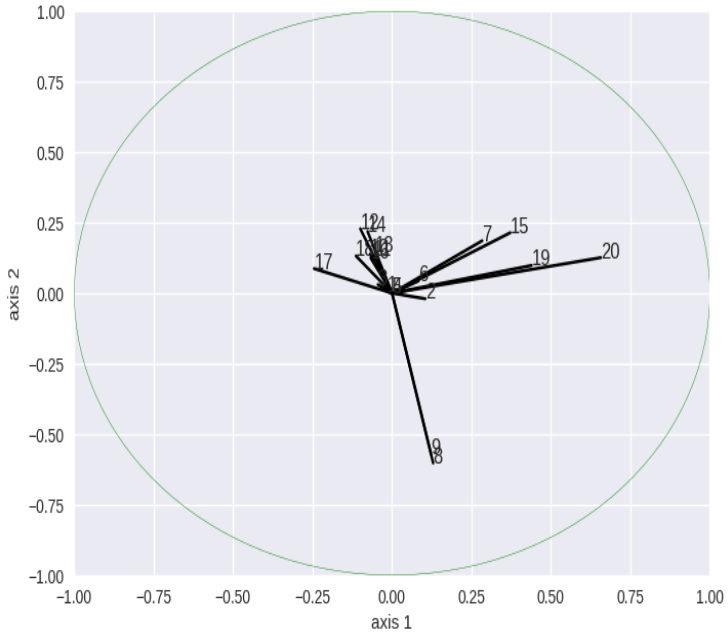

Figure 3.9: Circle of features correlation [Log min-max scaling]

Figures 3.8 and 3.9 show the variables factor map focusing on correlation between features for both cases. The first figure actually highlights the extremely high correlation between features represented by the same range of numbers as mentioned in the subsection 2.5. This is actually highlighted by the high adjacency of features upon the first axis. As for the second input dataset, we note that the correlation circle is more clear. Time related features 15, 19 and 20 are adjacent and consequently high correlated. In the other hand, 8 and 9 referring to the variance and mean of neighbours degrees (users with whom a specific user has transacted) are negatively correlated with the user's behavioural features which seems totally reasonable.
This can be explained by the fact that well known services such mining pools or wallet services which are characterized by high in and out degrees make continuous exchanges with their clients or normal users with relatively poor transactional history which results in low degrees and vice versa.
Though, we conclude that the second input dataset offers a better modelling for relations between users' behavioural features. This is explained by the fact that logarithmic transformation actually helps remove non-linearity and represents value ranges uniformly.

3.2.5.2 Features selection

Features selection is a key step in every data mining task. In this part, we relied on the training data set with labelled users of [4]'s researchers. As the training set con-

tains only few examples representing different bitcoin users' classes, we applied the random forest algorithm which is based on bootstrap aggregation or bagging technique to construct a multitude of decision trees at the training time using randomly sampled data and outputs the mode of each class. One of this algorithm outputs is an insight about variables importance in a classification problem. Generally, feature relative importance is measured using a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance is (figure 3.10). Finally, retained features are those with rel-

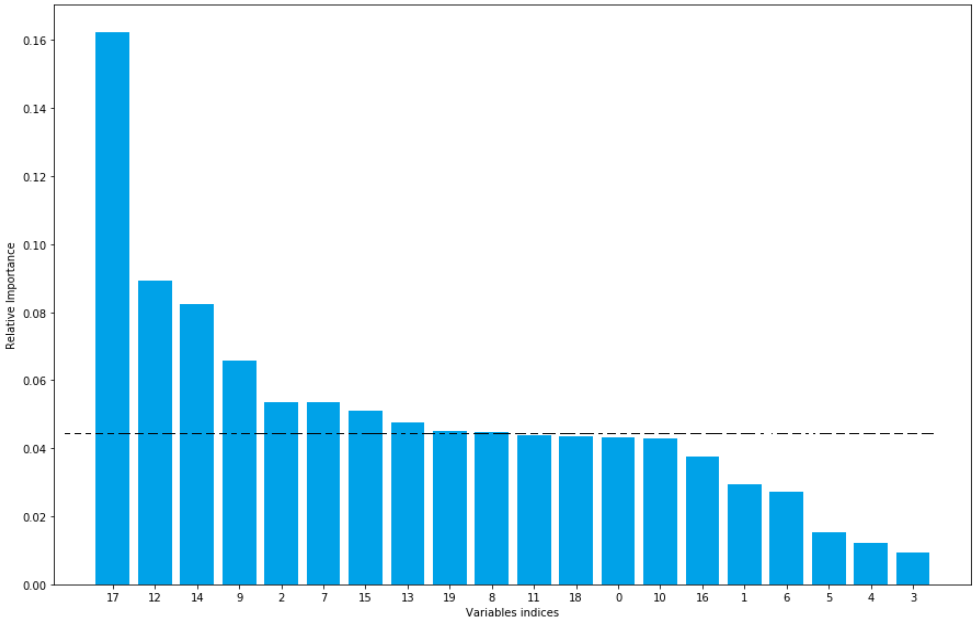

Figure 3.10: Features importance

ative importance higher than 0.05. The next list highlights features in descending order of importance:

1. OutDegree
2. Average of bitcoins received
3. Minimum of bitcoin sent
4. Time average between outgoing transactions
5. Time average between ingoing transactions
6. Variance of bitcoin on all user's transactions
7. Mean of neighbours' degrees

8. Variance of neighbours' degrees
9. Maximum of bitcoin received
10. Maximum of bitcoin sent
11. Average of bitcoin sent
12. activity duration

3.2.5.3 Model implementation & result visualizations

In this section, we try to provide a comparison between two approaches. In the first model, we tried to implement the elbow method which consists in determining visually the appropriate number of clusters (see Appendix C). Then, we applied k-means algorithm to cluster data samples using all extracted features.
Figure 3.11 shows the result of the elbow method. 7 is the chosen number of clusters justifying approximately 70% of variance explained. Indeed, the next figure highlights the result of k-means algorithm with pair-plots of most discriminating features.

In the second model, we tried to implement the hierarchical ascendant clustering method to build a hierarchy of clusters and present the dendrogram. This tree diagram helps determining the number of clusters k which coincides with the greatest inter-classes inertia loss. This parameter will be used later as an input for k-means algorithm. The next step was to use the same clustering algorithm with only most important features determined in the previous section.

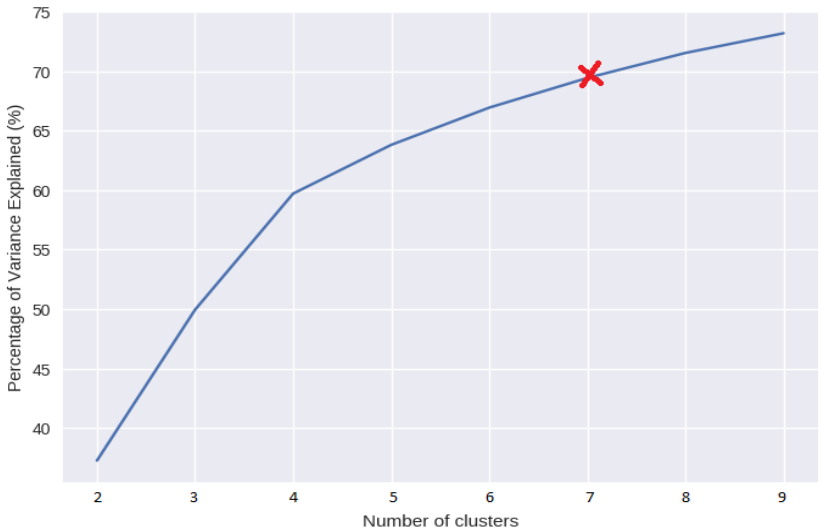

Figure 3.11: Elbow method for determining the number of clusters

As it appears in figure 3.12, our dataset is split into 7 different clusters which are highly adjacent. This actually reflects the low between-cluster variance and is totally justified by the fact that only excessively active bitcoin users are considered in our study. Actually, most dominant bitcoin components such as exchange services, mining pools and gambling sites operate similarly; they transact with millions of users on a daily basis thus they have similar features' values. Approximately all combinations of features among most discriminating ones are robust and represent well users' classes.

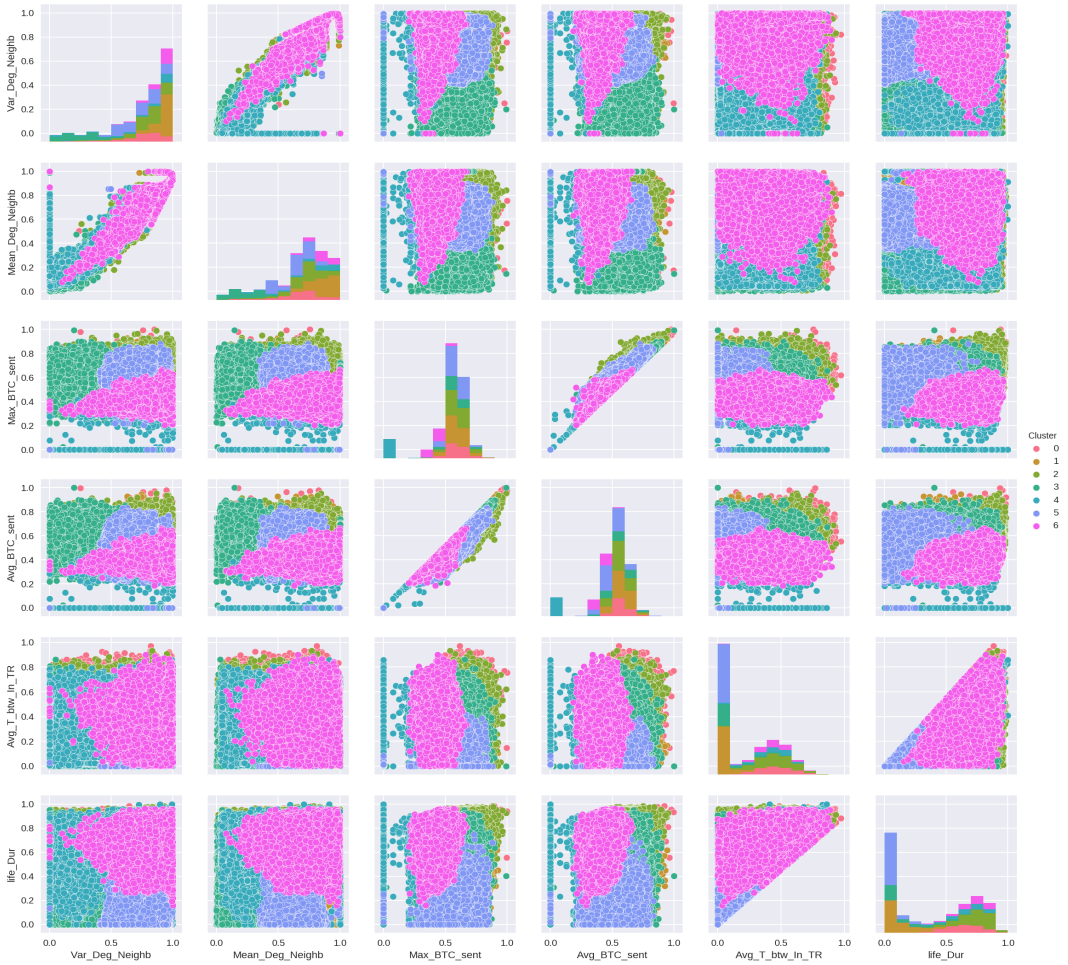

Figure 3.12: Clustering result with seven most discriminating features

The next figure (3.13) shows the resulting dendrogram through which we were able to decide that 8 is the number of clusters as it indicates the most important inertia loss.

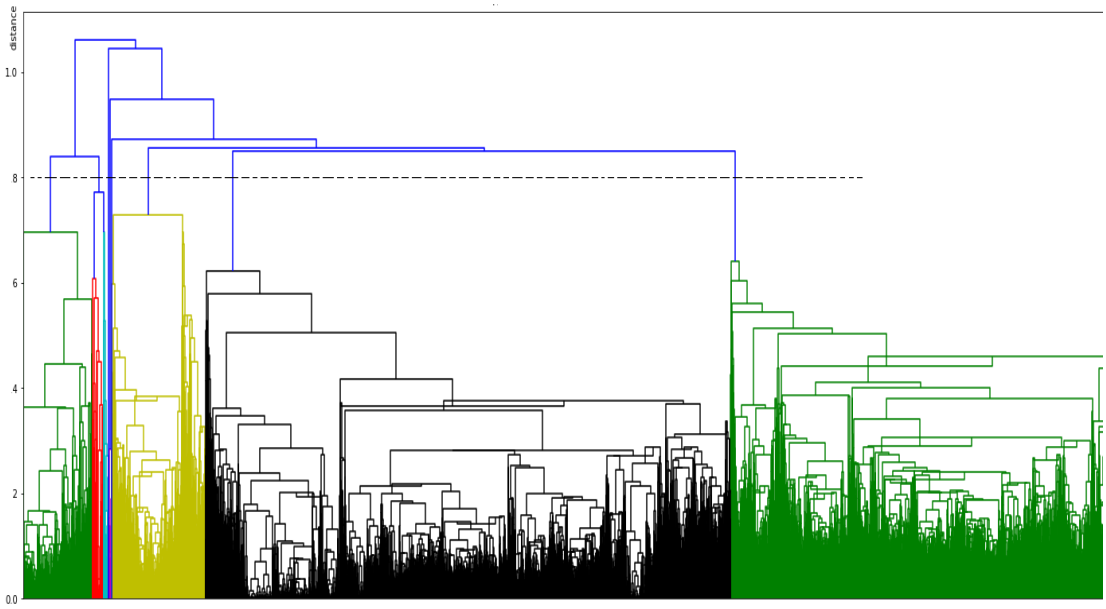

Figure 3.13: Dendrogram of bitcoin users population

After performing the clustering task, we tried to evaluate the performance of each model. As we worked with unlabelled data, we couldn't afford a real mechanism to validate our findings. Consequently, we relied at this point on [4]'s ground truth data to find out to which user category each cluster refers to and which model builds a more compact and stable clusters. When performing the cross-validation of users' clusters with our first model's findings, we were able to identify only three users classes namely: vendors, miners and exchanges and wallet services with high error rates, around 40%. However, the result of second model presented less error rate and seemed to be quite promising; using only important features as input for our clustering algorithm enhanced clustering and enabled us to define activity patterns of all users' profiles present in [4]'s data which are bitcoin network key components, namely; miners, exchange and wallet services, gambling sites and finally vendors (see confusion matrix in figure 3.14).
The following table highlights the repartition of samples among 8 clusters of the second model.

Table 3.5: Samples' repartition among clusters

Clusters	**Cl 0**	**Cl 1**	**Cl 2**	**Cl 3**	**Cl 4**	**Cl 5**	**Cl 6**	**Cl 7**
Percentages (%)	**29.8**	**18.24**	**6.53**	**7.14**	**13.58**	**4.85**	**8.85**	**11.3**

We should note that Clusters 1, 4, 6 and 7 are identified respectively as Miners' cluster, vendors, Exchange and wallet services and finally gambling sites' cluster.

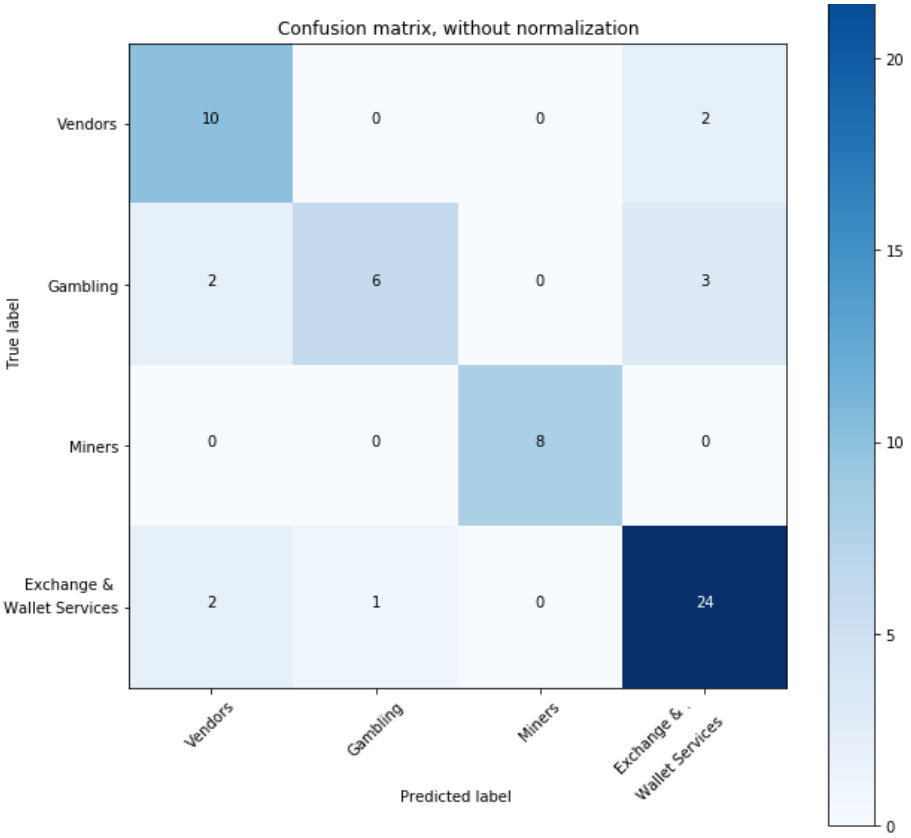

Figure 3.14: Confusion matrix after variables' selection

However, we were unable to identify to which users' profile refers all the rest of non-identified clusters due to the lack of labelled samples especially those pointing to fraudulent bitcoin users. Certainly, one of the non-identified classes points to traders who are interested in financial transactions with bitcoin currency known for its high speculation. Indeed, fraud makers profile may suit one or more of the non identified clusters.
When trying to manually label fraudulent users, we searched through bitcoin forums such as bitcointalk.org & blockchain.info/ tags looking for addresses associated to thieves, scammers or defunct services. Although these sources are less reliable than [4]'s labelled users, we opted to consider this sample in order to test our clustering results and find out if our model can gather fraudulent users and associate them to one cluster.
So, we considered the new data set; the result of merging [4]'s users with newly collected fraudulent samples to identify which class represents fraudulent users and need a more attentive inspection and we were able to notice that the majority of identified fraudulent users belong to one same cluster; Cluster 0 which detains the majority of our dataset samples (29%). This result might be promising. However,

we should be testing on well-known fraud cases gathered from more reliable data sources to gain some confidence.

3.2.5.4 Time series analysis block

Gathering all addresses controlled by each user using the bitcoin protocol heuristics helped us aggregate addresses' activities and retrace users' wallet evolution over time. This is done by ordering chronologically addresses' transactions over time and capturing ingoing and outgoing bitcoin amounts for every user.

When trying to manually inspect and compare activities of suspicious users and non fraudulent bitcoin clients, we concluded that thieves usually use new addresses to recover stolen bitcoins and never reuse them in multi-input transactions with other addresses, if they got any. In this way, they will probably be identified as normal users and without the label, researchers will classify this activity as a normal one. We conclude also that a glance to users' activities only is not sufficient to detect suspicious events and that we should always resort to a manual inspection of the blockchain and specifically transactions that followed thefts to identify patterns of suspicious activities. When considering aggregations while inspecting some thefts, we noticed that many addresses which were not clearly associated to thefts were used in next transactions to split and clearly to clean the stolen money. However, considering each user activity who controlled these addresses does not reveal abnormal patterns.
Finally, as we were unable to go further on inspecting suspicious users on the previous step, we thought about enlarging our knowledge and our understanding of already identified bitcoin users' profiles assuming that knowing perfectly what does not represent fraudulent users helps us also seize better what could characterize them. So, the question we tried to answer in this section is how far is this data valuable ? Can we characterize already identified users' profiles by their wallet activity over time ?

Time series Exploration

The first step was about an exploratory visualization of time series in a bid to discover and disclose their in-common properties. The first thing we noticed is that time series representing users' wallets evolution are not stationary; in other words statistical properties of these processes such the mean, variances and covariances evolve over time. Consequently, as non-stationary data is unpredictable and cannot be modelled or forecasted, we abandoned the idea of modelling our identified users' time series using a forecast model such as ARIMA or ARMA models which could helped us distinguish the category of time series data that does not fit our models and classify it as belonging to suspicious users. The next figure represents the rolling mean and variance of the log of a time series belonging to an exchange service.

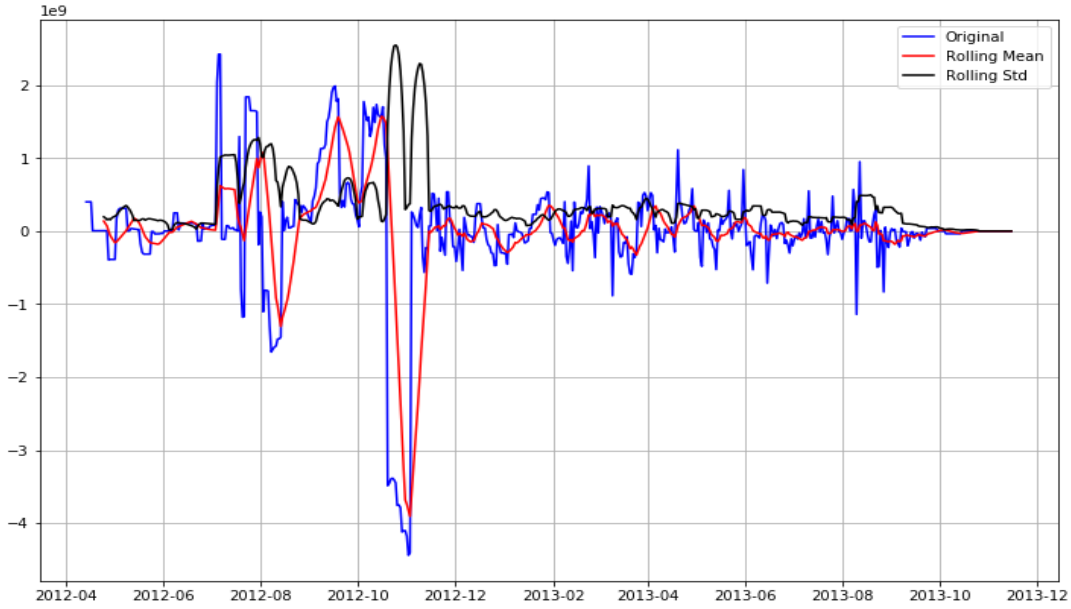

Figure 3.15: Rolling mean and standard deviation of time series log

When delving further our time series data set and trying to visualize simultaneously those belonging to one same class users, we concluded that users' activities do not represent a common and visual repeating pattern over time. Added to that, time series of same class users do not represent a common deterministic long-term increase or decrease trend nor a seasonal pattern. This can be explained by the fact that the majority of each class samples appeared in a different and a specific stage of bitcoin network existence. Thus, every sample's underlying dynamics and actions justifying every transaction occurrence and globally every wallet evolution over time are different. We finally opted to proceed with features extraction step.

Time series classification

Before proceeding with features' extraction, we begun by some crucial pre-processing tasks including resampling our time series data. In fact, users' wallets evolution were measured using all addresses aggregation on instants of ingoing or outgoing transactions only and were not captured on a periodic basis. So, this pre-processing phase helped us get equally spaced time series. We fixed the measures between-interval to ten minutes which corresponds to the minimum duration that separate validation moments of two successive blocks of the blockchain. The next step was about de-noising time series data using a filter in order to remove unnecessary signal fluctuations and get a smooth time series plottings.

We extracted all time series-based features listed in section 2.7 and we relied on [4]'s labelled data to train a decision tree and test the discriminating potential of all extracted features for the classification of four key components of bitcoin protocol present in our labelled data set. After training the decision tree with 70% of our labelled data (equal to approximately 51 samples), running the predictive model on the test set gave us results summarized in the following figure.

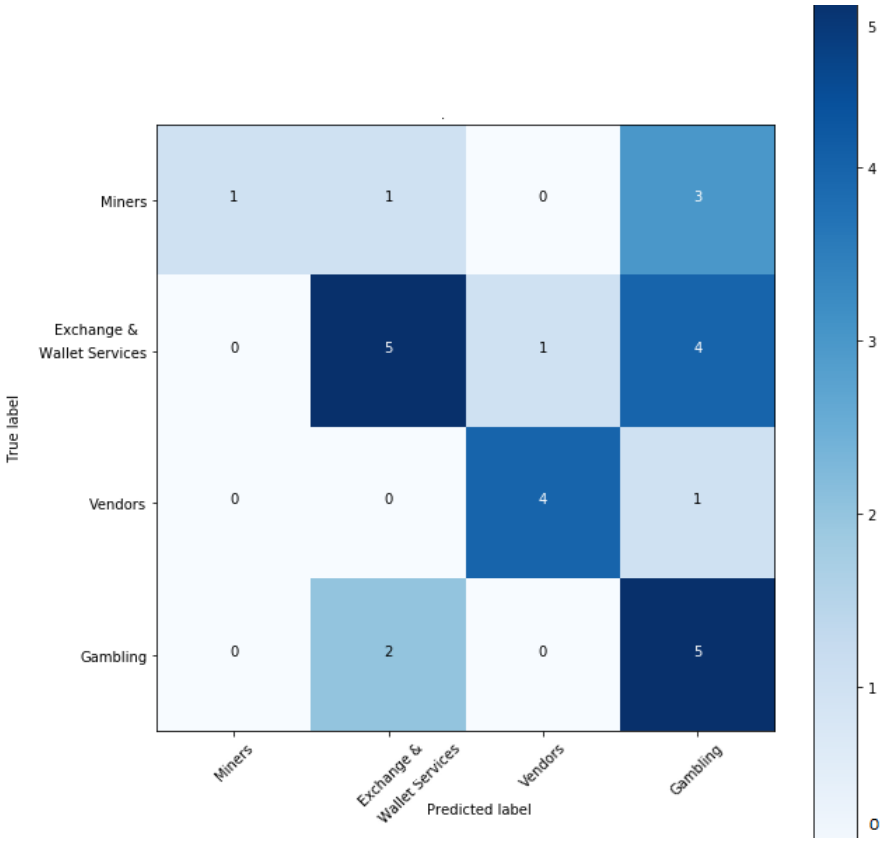

Figure 3.16: Confusion matrix with time series based features

We can conclude that extracted features of our model can distinguish and classify better Gambling sites and Vendors. However, our predictive model is less efficient when trying to predict miners and exchange and wallet services classes. This can be explained by the fact time series data alone is not sufficient to describe the overall behaviours of bitcoin network components; other information in addition is crucial to build a comprehensive and global insight about users' activities.
To sum up, we were able in this work to identify different users' profiles and build models to characterize their activities' patterns on bitcoin network. However, the lack of labelled data was a real handicap to test our solution and go further in the study of fraudulent pattern in bitcoin network.

Conclusion

We covered in this chapter the implementation of the proposed system for users profiles identification on the bitcoin network. We started by describing development environment and technological choices adopted in this project. We presented then developed modules' implementation, testing results and highlighted also every task complexity and main limitations.

Conclusion and perspectives

Throughout this project, we attempted to design and implement an analytical solution to identify, characterize efficiently different users' profiles of bitcoin currency and try to inspect suspicious patterns. Our target dataset holds about 78 762 324 transactions involving 88 171 000 public keys used to represent anonymous bitcoin clients.

Our main objective was first to proceed with the user discovery process, define secondly activity patterns of key bitcoin network components and finally provide at-glance insight about fraudulent users exchanging this virtual currency for illegal purchases. This was actually achieved by a set of big data tools, graph theory related algorithms, time series analysis techniques and ML clustering and classification-based models namely: k-means, hierarchical ascendant clustering, decision tree and random forests.

Processing such a huge dataset required a parallel and distributed computing environment to handle the computation complexity. That's why we resolved to Big Data technologies implemented on Spark one-node cluster to proceed with data batch processing. Then, as input data uniquely defines address-to-address transaction level, the next challenge was to reduce users' anonymity and associate addresses to its appropriate controller using intrinsic heuristics of bitcoin protocol which are mainly related to mutli-input transactions and change addresses. We set up also a validation process to verify the consistency of adopted heuristics in the user discovery process. It consists on computing a set of measures, namely, aNMI, NMI, Recall, Precision and the F1 measure which compare our results to ground truth data and provide a feedback about their accordance.

Then, after the redefinition of real entities which are users, the next step of our project was features building. We considered, at this point, a set of behavioural and graph related features that describe and quantify users' activities. Computing these features was actually carried out with a study specifying the complexity of such a task in term of time. By this step, therefore, starting from a huge dataset of transactions, we created a dataframe of anonymous individuals described and quantified by N features.

At this level, we moved to the data mining task through which we compared two approaches for users' clustering. [4]'s ground truth data helped us evaluate the accuracy of our two models' clusters and we found that features' selection enabled us get a more accurate matching users profiles to each cluster. Through this validation procedure we finally identified four among key bitcoin network components which are:

1. Miners
2. Exchange and wallet services
3. Vendors
4. Gambling sites

Although we were unable to identify suspicious users due to the lack of labelled samples pointing to this specific category of users, recognizing popular categories among bitcoin users is quite promising regarding the high complexity of this issue. To further enlarge our study, we tried to inspect dynamic patterns and wallet evolution of already identified bitcoin usersâprofiles assuming that knowing perfectly what does not represent fraudulent users can help also seize better what could characterize them. We tested in our work how far time series data can characterize identified users.

At this level, we can conclude that the initially set objective was achieved. We were able to reduce users anonymity, quantify users activities with behavioural and graph related features and finally proceed with data mining task.

However, the main limitation of our work is the lack of labelled data referring to suspicious users which could helped us build a more efficient model. At this point, further tasks could be developed to improve our solution, namely:

- Label data pointing to suspicious users which will allow the implementation of supervised ML algorithms to set up a detection mechanism of suspicious activities.
- Merging features that highlight the evolution of users activity over time and static features in order to boost the accuracy of our users profiles identification block.
- Implementing the same process in a real-time fashion. Parsing streaming flows of bitcoin transactions can be easily done using Spark.

Throughout this prototype, we could confirm the complexity of the high anonymity bitcoin provides to its users and also the power of big data, graph theory and Machine Learning to parse even anonymous exchanges of virtual currencies.

Bibliography

[1] *https://www.lip6.fr/recherche/team.php?acronyme=ComplexNetworks*, (visited on August 2017).

[2] *iTRAC technical sheets, Systematic Paris-Region, pages 2-5, November 2016.*

[3] S.Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System Bitcoin: A Peer-to-Peer Electronic Cash System*, Online, December, 2008.

[4] S. Meiklejohn et al., *A fistful of bitcoins: characterizing payments among men with no names*, In Proceedings of the 2013 conference onInternet measurement conference, pp. 127 -140, December, 2013.

[5] *https://www.coindesk.com/price/, visited on November 2017.*

[6] *https://coinmarketcap.com/*, (visited on November 2017).

[7] F.Reid, M.Harrigan, *An analysis of anonymity in the bitcoin system*, In Security and Privacy in Social Networks, Springer pp. 197-223, May, 2013.

[8] V. Buterin, *A next-generation smart contract and decentralized application platform*, White Paper, Online, September, 2014.

[9] N. Christin and R. Safavi-Naini, Eds., *Financial Cryptography and Data Security*, vol. 8437. Berlin, Heidelberg: Springer Berlin Heidelberg, July, 2013.

[10] *https://blockchain.info/fr/stats*, (visited on November 2017).

[11] M. Moser, R. Bohme, D. Breuker. *An inquiry into money laundering tools in the Bitcoin ecosystem.* eCrime Researchers Summit (eCRS), April, 2013.

[12] D. Ron and A. Shamir, *Quantitative analysis of the full bitcoin transaction graph*, in International Conference on Financial Cryptography and Data Security, pp. 6-24, July, 2013.

[13] P. De Filippi and B. Loveluck, *The invisible politics of bitcoin: governance crisis of a decentralized infrastructure*, Internet Policy Review, Vol.5, pp. 54-59, October, 2016.

[14] S. Ahamad, M. Nair, and B. Varghese, *A survey on crypto currencies*, in 4th International Conference on Advances in Computer Science, AETACS, pp. 42-48, May, 2013.

[15] C. Harwick, *Cryptocurrency and the Problem of Intermediation*, Spring, Issue of The Independent Review, 2015.

[16] E. Androulaki, G. O Karame, M. Roeschlin, T. Scherer, S. Capkun, *Evaluating user privacy in bitcoin*, International Conference on Financial Cryptography and Data security, pp 34-5, Springer, July, 2013.

[17] M. Spagnuolo, F. Maggi, S. Zanero. *Bitiodine: Extracting intelligence from the bitcoin network*, In the International Conference on Financial Cryptography and Data Security, pages 457-468. Springer, November, 2014.

[18] T. Pham and S. Lee, *Anomaly Detection in the Bitcoin System-A Network Perspective*, Information Security, November, 2016.

[19] Thai T. Pham, Steven Lee. *Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods*, Cryptography and Security, November, 2016.

[20] J. Hirshman, Y. Huang, and S. Macke, *Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network*, Technical report, Stanford University, April, 2013.

[21] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*, New York: North Holland, 1976.

[22] Mark Newman. *Networks, an introduction*, pages 482-490, 2010.

[23] K. Sravanthi and T. S. Reddy, *Applications of Big data in Various Fields*, Int. J. Comput. Sci. Inf. Technol. IJCSIT, vol. 6, no. 5, pp. 4629-4632, October, 2015.

[24] Govindaraju, Raghavan, Rao. *Big Data Analytics*, 1st Edition, Jul 2015.

[25] M. Sassi Hidri. *Big data course*, National Engineering school of Tunis, Dept. ICT, Tunisia, September, 2016.

[26] *http://spark.apache.org*, (visited on September, 2017).

[27] Et.Alpaydin, *Introduction to machine learning*, pages 1-5, Massachusetts Institute of Technology, 2010.

[28] B. D. Fulcher, *Feature-based time-series analysis*, Monash Institute for Cognitive and Clinical Neurosciences, Monash University, Victoria, Australia, October, 2017.

[29] S. J. Wilson, D*ata representation for time series data mining: time domain approaches: Data representation for time series data mining*, Wiley Interdiscip. Rev. Comput. Stat., vol. 9, no. 1, p. e 1392, January, 2017.

[30] T. Fu, *A review on time series data mining*, Eng. Appl. Artif. Intell., vol. 24, no. 1, pp. 164-181, February, 2011.

[31] F. Morchen, *Time series feature extraction for data mining using DWT and DFT*, Univ., November, 2003.

[32] S. M. Pincus. *Approximate entropy as a measure of system complexity.* Proc. Natl. Acad. Sci. USA 88, 2297, March, 1991.

[33] J. S. Richman and J. R. Moorman, *Physiological time-series analysis using approximate entropy and sample entropy*, Am. J. Physiol. Heart Circ. Physiol. 278, H2039, June, 2000.

[34] C. Bandt, B. Pompe, *Permutation entropy: A natural complexity measure for time series*, Phys. Rev. Lett. 88, 174102, February, 2002.

[35] P. Esling and C. Agon, *Time-series data mining*, ACM Comput. Surv., vol. 45, no. 1, pp. 1-34, November, 2012.

[36] T. Lane, *Machine learning techniques for the domain of anomaly detection for computer security*, Purdue Univ., July, 1998.

[37] *https://gephi.org/features/*, (visited on August, 2017).

[38] *https://snap.stanford.edu/snappy/*, (visited on August, 2017).

[39] R. Cazabet, R. Baccour, M. Latapy, C.Magnien *Tracking bitcoin users activity using community detection on a network of weak signals*, LIP6, Pierre and Marie Curie University, Complex Networks, Lyon, November, 2017.

[40] H. Abdi, J. Lynne Williams, *Principal Component Analysis*, 2010 John Wiley Sons, Inc, Volume 2, July, 2010.

[41] D. Fer, *Cluster Analysis*, Extension Chapters on Advanced Techniques, pages 553-567, October, 2014.

[42] *http://spark.apache.org*, (visited on September, 2017).

[43] W. Ayadi, *Computational Intelligence course*, National Engineering school of Tunis, research master TICV, Tunis, November, 2016.

[44] A. Messaoudi, *Employee engagement and retention through Big Data*, Graduation project Thesis, National Engineering school of Tunis, September, 2016.

[45] R. Baccour *Analyse des tranasctions financieres en BITCOIN*, Graduation project Thesis, National Engineering school of Tunis, September, 2017.

Appendix A

Spark fundamentals

Spark is composed of the CORE layer and a set of up layers SQL, ML and graph processing libraries detailed below:

- Spark streaming is conceived for real-time data flow processing.
- Spark SQL and Dataframes expose Spark datasets via Java Database Connectivity (JDBC API) and run SQL queries using traditional viewing tools. Spark SQL allows to extract, transform and load data in different formats (JSON, Parquet, database) and expose them for ad-hoc queries.
- Spark MLlib is a machine learning library that contains all classical learning algorithms and methods such as classification, regression, clustering, dimension reduction, in addition to a set of pre-processing approaches for data scaling and noise reduction.
- Spark GraphX is graph processing and graph parallel computation API. It extends data' formats by introducing graph dataset which is an oriented multigraph with properties attached to nodes and edges. GraphX presents a set of basic operators as well as an expanding collection of algorithms and builders to simplify graph analysis tasks.

Spark Core creates Resilient Distributed Datasets (RDDs) which are distributed collections of elements or objects parallelized across the spark cluster holding datasets from any external storage system such as shared file system, HDFS or any data source offering a Hadoop Input Format. Processing RDDs actually obey the Lazy evaluation mode :
In fact, the Driver program written through the spark interactive shell will be converted into a Directed Acyclic Graph (DAG) which is an ordered sequence of spark transformations and actions. The last vertex is the final action which will trigger the DAG execution. In figure A.1, mapValues, groupBy, map and filter are all transformations updating the DAG. Parallelize is the action that launched the execution of the operation sequence.
At the execution time, a DAG scheduler takes in charge the coordination of assigned operations at the spark context level. It actually pipelines DAG operators together and splits them into stages of tasks. A stage is comprised of tasks based on partitions of the input data. The final result of a DAG scheduler is a set of stages which will passed on to the Task scheduler. The task scheduler then launches tasks via

cluster manager (Spark Standalone/Yarn/Mesos) which is responsible of allocating resources and instructing slave worker nodes to execute jobs. The following figure (A.2) shows the overall architecture of Spark and involved parts in jobs execution.

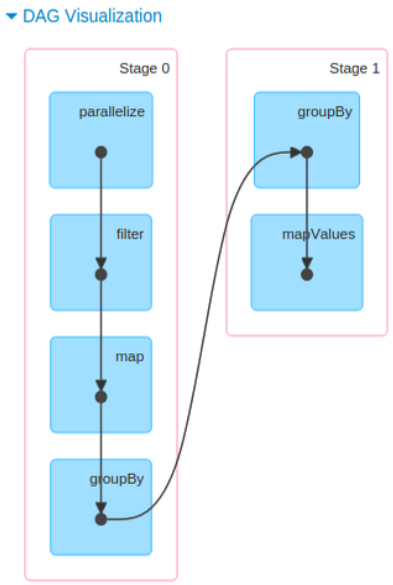

Figure A.1: An example of DAG

Executors of slave worker nodes are responsible of reporting the job status to the cluster manager. If one node crashes while executing a specific operation, the cluster manager assigns another node to continue the processing. The new node will operate on the particular partition of the RDD and refer to the DAG to determine the series of operation that it has to execute. This will actually ensure both high performance and fault tolerance.

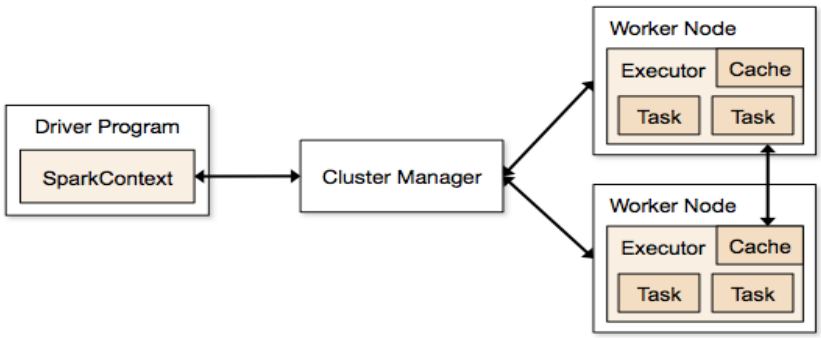

Figure A.2: Internals of Job execution in Spark [26]

Furthermore, Spark in-memory calculation approach makes it a very fast large-scale data processing engine as it avoids multiple disk inputs and outputs which heavily lower performance and increase processes' latency.

Spark supported modes

We can operate Spark in one of three modes, namely :

- Standalone: running as a single scala process, once downloaded, Spark is by default configured in a standalone mode [26].
- Mesos: is a centralized cluster manager, designed for distributed computing environments. It provides resource management and isolation, scheduling of CPU & memory across the cluster [26]. Mesos enables joining multiple physical resources into one single point which boost the system performance.
- YARN: makes Spark benefits from Hadoop next generation where functionalities of resource management and job scheduling and monitoring are spelt into separate daemons [26].

Appendix B

Measures for user discovery heuristics evaluation

Precision, Recall and F1 measure

Precision and Recall are typically used to assess the quality of classification tasks. They can also be used to evaluate the quality of clustering: Precision corresponds to how often elements are correctly grouped together, while Recall evaluate how often elements belonging to a same cluster are labelled as such in the tested partition.

More formally, for each pair of addresses, we check if they are True Positive (TP), False Positive (FP) or False Negative (FN):

- TP: two addresses with the same label are in the same cluster.
- FP: two addresses with different labels are in a same cluster.
- FN: two addresses with the same label are in different clusters.

Then Precision and Recall are defined as usual:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

The F1 score is the harmonic mean of Precision and Recall, and summarize the quality of a clustering in a single value.

NMI and aNMI

The Normalised Mutual Information is frequently used to assess the correspondence between clusterings or community partitions. It owes its definition to information theory field and corresponds to the mutual information between the two compared partitions normalized between 0 (no similarity) and 1 (identical partitions). Several variants of the normalisation can be used; we use the following definition:

$$NMI = \frac{I(U,V)}{\sqrt{(H(U) \times H(V)}}$$

With I(U,V) the mutual information between partitions U and V and H(U) the entropy or the quantity of information of partition U.

Despite being often used in community detection evaluation, NMI suffers from known limitations, in particular that random partitions have expected values higher than 0, that can even be relatively high in some cases. A commonly used variant is the adjusted for chance NMI, called aNMI, which conserves the normalisation but correct for the effect of chance.

Appendix C

Algorithms and techniques used in data mining task

Principle Component Analysis [44]

PCA is a descriptive statistical method used to analyze and observe datasets that involve more than one descriptive variable at time. This multivariate procedure is based on orthogonal transformation to convert a set of I observations described by a J possibly correlated variables into a set of new linearly uncorrelated features.

Mathematically, PCA is an orthogonal linear transformation. In fact, it creates a new coordinate system where the first coordinate, defined as the first principle component, holds the greatest variance(i.e. inertia), the second coordinate holds the second greatest variance under the constraint of being orthogonal to the first one, and so on.

Geometrically, values of these new variables, called factor scores, are interpreted as the projections of observations onto principal components. Pattern of observations' similarity, as well as original variables, are then displayed as points projected in the factorial plane. A factorial plane is a coordinate defined by two of q retained components.

The examination of different factorial plans enables the visualization of the correlation between variables as well as identifying groups of individuals that have close values regarding one or more variables. Components are retained using two criteria :

- Kaiser criterion : only components with an eigenvalue higher than 1 are retained.
- Elbow criterion : we choose the maximum number of components so that an additional component won't provide better data modelling.

K-means [43]

K-Means clustering method takes as input N observations described by m features and a second parameter k which is the number of clusters. The algorithm partitions

these observations into k clusters based on their similarities. This similarity is actually defined as closeness computed using a specific distance usually the Euclidean distance.

The algorithm starts by randomly selecting K first centroid locations for each cluster, from the input points. At this stage, the algorithm repeats, until it converges, the two following steps:

- Cluster Assignment : each observation is assigned to its closest cluster characterized by a single centroid.
- Update Centroids : Once all input observations are assigned to clusters, the mean of each cluster is computed and chosen as a new centroid.

Finally, each observation is assigned to one cluster. Clusters are build in a manner to decrease the variance intra class (dissimilarity between one class's observations) and increase the variance inter classes (dissimilarity between classes).

Elbow Method

Elbow method is one of the most known techniques for determining the appropriate number of clusters in a dataset. This method is based on the computation of the percentage of variance explained as a function of the number of clusters. We define the variance explained as the between-group variance or in another words the measure of dissimilarity between different clusters.
So, the percentage of variance explained is the ratio of this between-group variance to the total variance.
After plotting the resulting curve, the most convenient number of clusters is chosen according to the elbow criterion; we choose the highest number of clusters so that the following number won't provide a significant hop of the explained variance.

Scaling techniques

K-means algorithm is sensitive to scale difference of input features. This means that if one of the features has a broad range of values, the distance will be governed by this particular feature. Consequently, features will not contribute proportionately in the partitioning process. A widely common practice is to minimize the effects of scale on the cluster assignment by applying a scaling technique on the target dataset.

1. Min-Max technique tends to rescale features range between [0,1] and more generally between [-1,1]. In our case, the target range is [0,1] as behavioural features such as the active duration of a user, the indegree and the outdegree can't be valued negatively. The formula of this scaling technique is :

$$x' = \frac{x - min}{max - min}$$

where x' is the new normalized value, x the ancient value, min refers to the minimum of the series' values and max is the maximum of the series' values.

2. Standard deviation technique is used to rescale numerical series in a manner to have values with standard normal distribution. New values are centred around 0 with a standard deviation of 1 ($\frac{1}{4}$=0 and =1) where $\hat{\text{I}}\frac{1}{4}$ is the mean and Ï is the standard deviation from the mean. The formula of this scaling method is given as :
$$x' = \frac{x - \mu}{\sigma}$$
where μ and σ are respectively the ancient mean and standard deviation of the mean. x is the initial value extracted from the numerical series.

3. Log min-max technique consists on applying the log function on all the features' set before using the min-max technique described in 1. The formula is given :
$$x'' = \frac{x' - min}{max - min}$$
where x'=Log(x) and x" is the scaled value.

Hierarchical Ascendant Clustering [44]

Hierarchical clustering is a cluster analysis method based on building hierarchy of clusters. The results are presented in a tree diagram called dendrogram.
This method can be implemented using two approaches :

- Agglomerative : each observation is assigned to its own cluster. Then, as we move up in the hierarchy, closely-related pairs of clusters are merged until forming the classification tree. HAC algorithm is based on this approach.
- Divisive : all observations are considered as a single cluster. They are then splitted into different clusters as we move down the hierarchy. Partitioning methods use this approach.

Implementing HAC implies definition of the method to use to calculate distance between individuals. In this work, we chose to use minimize the Euclidean distance between observation to construct clusters.

Decision Tree

Decision tree is a classification or regression model in the form of a tree structure. It splits a data set into two subsets using the most discriminative feature on each iteration. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches however, a leaf node represents a decision. The topmost decision node in a tree which corresponds to the best predictor called root

node. Decision trees are usually used to handle categorical data. Decision tree algorithm Pseudo-code can be summarized as below:

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

Random forest

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees. In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.