



Investigating Subscribers, their Wealth and their Social Ties using Mobile Network Call Detail Record Analysis

Conor Donohue

B.E Electronic & Computer Engineering Project Report

April 2017

*College of Engineering & Informatics,
National University of Ireland, Galway*

Supervisor: Mr. Liam Kilmartin

Co-Assessor: Prof. Gearóid Ó'Laighin

ABSTRACT

The aim of this project is to use Mobile Call Detail Records (CDR) containing solely USSD requests and responses to investigate social relationships between subscribers. The research conducted in this project will also examine if there is any connection between these social interactions and the wealth of users. For this to all be completed, a framework must be created to house the data, provide statistical analysis and to generate prediction models to classify wealth amongst subscribers in the mobile dataset. To allow this to happen techniques and software tools must be studied to see which can be applicable to CDRs. The data must then be analysed using the tools and techniques that were found to be most appropriate.

The rest of this project will explore how users can be divided into different communities based on who they contact and how these social ties in the different communities can be analysed. Predictive models will be shown which will predict the amount of times a user tops up their credit. These users will be divided into wealth categories based on this interaction with the credit top up service.

STATEMENT OF ORIGINALITY

I declare that this thesis is my original work except where stated

Date: _____

Signature: _____

ACKNOWLEDGEMENTS

Over the course of this project, I have received advice, guidance and feedback from a group of people who I wish to thank for their help and support. Firstly, I would like to express my gratitude to my supervisor, Mr. Liam Kilmartin, for all his advice and guidance throughout the project. I would also like to thank Professor Gearóid Ó'Laighin in his capacity as co-assessor. Mr. Martin Burke and Mr. Myles Meehan must also be thanked for their help and patience over the past academic year. Finally, I thank my parents, John and Patricia Donohue, for their continuing support over the past four years.

GLOSSARY

CDR	Call Detail Records, this is information recorded by telecommunications equipment in a network regarding calls, SMS and USSD transactions made by subscribers. Example of details recorded would be the date, time and number of subscriber.
MSC	Mobile Switching Centre, part of the 2G mobile network, used for switching and functionality such as call hang up.
USSD	Unstructured Supplementary Service Data, is a protocol used in telecommunications networks to offer different services such as credit balance checks and mobile data bundles.
MSISDN	Mobile Subscriber ISDN Number, this is used to identify a subscriber in a telecommunications network.
JSON	JavaScript Object Notation, this is a file format which saves objects in an easily readable format.
CSV	Comma Separated Value, this is another file format which separates file attributes by placing commas between them.
PCA	Principal Component Analysis. This is a technique which is used to perform dimensionality reduction on data.
CMB	‘Call Me Back’ request, a USSD code. The user who invokes the code attaches another subscribers number in the code message. Then, the USSD service sends the other subscriber a text asking them to call the initiator of the code.
IMSI	International Mobile Subscriber Identity, used to identify a user in a telecommunications network.
HLR	Home Location Register, a database of subscriber information found in a telecommunications network. Contains information such as credit balance, last MSC connected to, is the subscriber active, etc.

TABLE OF FIGURES

Figure 1: A small portion of the raw data.....	2
Figure 2: Example of USSD Codes in Ireland [1].....	4
Figure 3 : User interaction with USSD Gateway [2].....	5
Figure 4. A sample MySQL table [10].....	10
Figure 5 : An example of a MongoDB Document.....	11
Figure 6 : Example of Graph Database, Neo4j [16].....	12
Figure 7: Example of looping through user specified directory.....	14
Figure 8: Hashing of MSISDNs for ethical reasons.....	14
Figure 9 : Screenshot of the unique USSD codes without any vouchers	16
Figure 10 : Function which collects all USSD codes	16
Figure 11 : Write Response that doesn't contain an error and its code to a file.....	17
Figure 12 : Screenshot of file with all code functionality	17
Figure 13 : Percentage for the most popular codes	18
Figure 14 : Code used to monitor directories	19
Figure 15: Function to allow Multithreading	21
Figure 16 : A sample 'Call Me Back' request.....	23
Figure 17 : The Network Decay as threshold on edge weight increased	25
Figure 18 : Communities in the network.....	26
Figure 19 : Average Amount of 'Call Me Back' Requests based on Communities.....	27
Figure 20 : Average Amount of Credit Top Ups based on Communities	27
Figure 21: Power Law Distribution of 'Call Me Back' Requests	29
Figure 22: Split Power Law.....	30
Figure 23 : Power Law for Credit Top Ups.....	31
Figure 24 : PCA Graph.....	32
Figure 25 : Monthly Credit Tops.....	34
Figure 26 : Monthly 'Call Me Back' Requests	34
Figure 27 : Distribution of Credit Top Ups	35
Figure 28 : Wealth Distribution.....	36
Figure 29: Correlation and SVM accuracy.....	38

TABLE OF CONTENTS

Abstract.....	ii
Statement of Originality	iii
Acknowledgements	iv
Glossary.....	v
Table of Figures.....	vi
Table of Contents	vii
1. Introduction	1
1.1. Project Overview	1
1.2. Chapter Overview	2
2. Background Review	4
2.1. USSD Codes	4
2.2. Information Source	5
2.3. Literature Review	7
3. Framework for Analysis.....	9
3.1. MySQL	9
3.2. NoSQL	10
3.2.1. Document Orientated Databases	10
3.2.2. Graph Based Databases	11
3.3. Selection of Database.....	12
3.4. Data Pre-Processing.....	13
3.4.1. Parsing of Raw Data into MongoDB	13
3.4.2. Isolating Unique USSD codes	15
3.4.3. Correlating USSD codes to responses	16
3.4.4. Discovering USSD code usage.....	18

3.5. Real Time Database	18
3.6. Techniques employed to handle big data.....	20
4. Analysis.....	23
4.1. Social Analysis	23
4.1.1. Network Visualisation and Analysis	23
4.1.2. Fitting of Power Law to Data for Analysis	28
4.2. Analysis of Wealth and its distribution.....	32
4.2.1. Principal Component Analysis	32
4.2.2. Monthly Distribution	33
4.2.3. Categorisation of Users into Wealth Brackets.....	35
4.2.4. Predicting Subscribers wealth	37
5. Conclusions and Future Work.....	39
References	41

1. INTRODUCTION

This chapter gives an overview of the project and a detailed breakdown of how the project works. A chapter overview will also offer a quick synopsis to each chapter in the order they will be read.

1.1. PROJECT OVERVIEW

The field of Mobile Data Analysis is an area that is growing year on year. Once it was realised the wealth of information that mobile CDRs contained, the field began to see a substantial increase in the amount of research completed.

This project attempts to use a CDR dataset from the African country of Chad to investigate the wealth and social interactions of users in the network. The data begins at the tail end of February 2016 and concludes in August 2016. Data is recorded from two different MSCs. The dataset does not contain any calls or SMS messages belonging to users, which would offer social interactions between two subscribers. It is a dataset focusing completely on USSD codes. Hence the normal social relationships which exist in a telecommunications network are not available for research on this project. This means that users mainly interacted with a USSD service instead of another subscriber, i.e. a person to machine interaction.

The dataset was handed over as text files, each with a unique transaction on every separate line. The size of the data must be noted, the magnitude of the raw data was recorded as 8Gb, zipped. The huge volume of data that had to be analysed would prove to be one of the biggest obstacles faced in the project.

The first portion of this project was to discover what each part of a USSD transaction stood for and to research a suitable database to hold the dataset. Figure 1 shows an example of what was contained within the raw text files. The provision of real time access to the data was also a requirement.

```

000410020303020602020401020005,17,2,2016-03-01,00:51:03,1nzZljfAr,233262241205,11477618,Your balance will be
sent to you via sms.,0,0

000410020303020605040503060809,17,1,2016-03-
01,00:51:03,1nzZljfAE,233265453689,0060153869480,11477631,*100#

000410020303020602050506020509,17,2,2016-03-01,00:51:03,1nzZljfAx,233262556259,11477624,*125#,-1,108

000410020303020608030406040107,17,1,2016-03-
01,00:51:03,1nzZljfAF,233268346417,0060150904597,11477632,*125#

```

Figure 1: A small portion of the raw data

A wide selection of databases were reviewed, ranging from My-SQL to a selection of No-SQL, ‘Not Only SQL’, databases. This concept will be reviewed in detail in Chapter 3.2. After testing, MongoDB was selected. One of the reasons for selecting this No-SQL database was a third-party library called PyMongo. This library allows MongoDB commands to be integrated into Python scripts. This allowed the raw data to be parsed into the database using Python scripts to separate each transaction and add the correct headers to each attribute. After the data had been parsed correctly and without errors, deciphering the functionality of each unique USSD code needed to be completed.

When knowledge of what each USSD code performed was acquired, a more in-depth analysis of the data could be completed. Using software tools such as Gephi, MatLab and Sci-Kit Learn, investigations regarding the number of times users topped up their credit and initiated the ‘Call Me Back’ service were conducted. The network and its subscribers were visualised using the open source graphing software tool, Gephi. Gephi allowed for the retrieval of statistics such as average degree and average weighted degree much quicker and easier than using a simple Python script. Sci-Kit Learn was used to implement different machine learning algorithms to predict attributes of a subscriber in the network.

1.2. CHAPTER OVERVIEW

This section will give a brief overview of this project and offer a quick insight into each chapter.

Chapter 2, Background Review, explains what a USSD code is and demonstrates applications of it that are found in Ireland. It also looks at published papers and how they

relate to Mobile CDR analysis as well as possible uses. It will also explore how these papers can relate to this project. Furthermore, Chapter 2 will discuss the source of information, i.e. the dataset of USSD codes. It will explain what each transaction can contain and how they can be used.

Chapter 3, Framework for Analysis, will look at how the data was saved and processed. Firstly, this section will explore the different types of databases and how these databases are suitable for the storage of this CDR dataset. Following on, the processing of data will be looked at in detail and how the text files can be successfully converted from text files to document objects in the selected database. Lastly the unique USSD codes will be explored and deciphered to conclude this chapter.

Chapter 4, Analysis, will discuss the experimentation and exploration of the dataset. It will look at both social aspects and the wealth of subscribers. Chapter 4.1, Social Analysis, looks at the social ties in the network. Software tools to visualise and analyse the network will be discussed. The characteristics of the network will be explored and discussion of trends will occur in this chapter. Some of these characteristics include the degree distribution of a subscriber, the network decay and the division of the network through the creation of communities. Chapter 4.2, Analysis of Wealth and its distribution, will explore the wealth of users through USSD codes and attempt to create a distribution of wealth by using the data that has been provided for this project. This section will use techniques such as PCA and Linear Regression to predict a subscriber's wealth and place them into a wealth bracket.

Chapter 5, Conclusions and Future Work, will look at areas of interest that could be looked at if work was to continue on this project. Some of these ideas which will be discussed could be developed into completely new projects that could contribute to the findings in this project. Chapter 5 will also summarise the findings of this report. It will look back at each aspect of this project and look at how each one contributed to the end result and conclusions of this project.

2. BACKGROUND REVIEW

This chapter looks at and explains the raw data in detail. It goes through each attribute of a USSD request and response. The next section of this chapter looks at published papers that contain relevant information to the project and others that look at different applications of Mobile CDRs.

2.1. USSD CODES

USSD stands for unstructured supplementary service data. In Ireland, it is mainly used to check mobile balances for SMS, minutes left and mobile data. Figure 2 demonstrates the services available to a bill-pay user in a mobile network in Ireland. The below figure demonstrates the USSD short code that are available in the 48 network.

From the phone, type in the following, then press "send" or "call"	Details returned
*100#	Your account balance
*100*1#	Remaining land line minutes
*100*2#	Remaining mobile calls allowance
*100*3#	Remaining texts allowance
*100*4#	Roaming minutes left (Outgoing)
*100*5#	Roaming minutes left (Incoming)
*100*6#	Roaming texts left
*100*7#	Get TU Me balance
*100*8#	How much internet you have left

Figure 2: Example of USSD Codes in Ireland [1]

In countries where smartphone penetration is low, there have been an abundance of USSD applications created to fill the void of enhanced features offered by smartphones through mobile applications. In Chad, there are USSD applications for banking, purchasing insurance as well as the standard mobile services. The difference between the two countries is clearly evident as the wealth of mobile applications and the high smartphone penetration rate in Ireland ensures that there is no need for the development of USSD services to the scale found in Chad.

When a USSD code is initiated, this transaction is routed through a USSD gateway to a USSD application server. Based on the code that was entered by the subscriber, the appropriate service will be found and a response will be generated. As can be seen in Figure 3 there are a wide variety of services that can be selected from and used, such as mobile banking and credit top up.

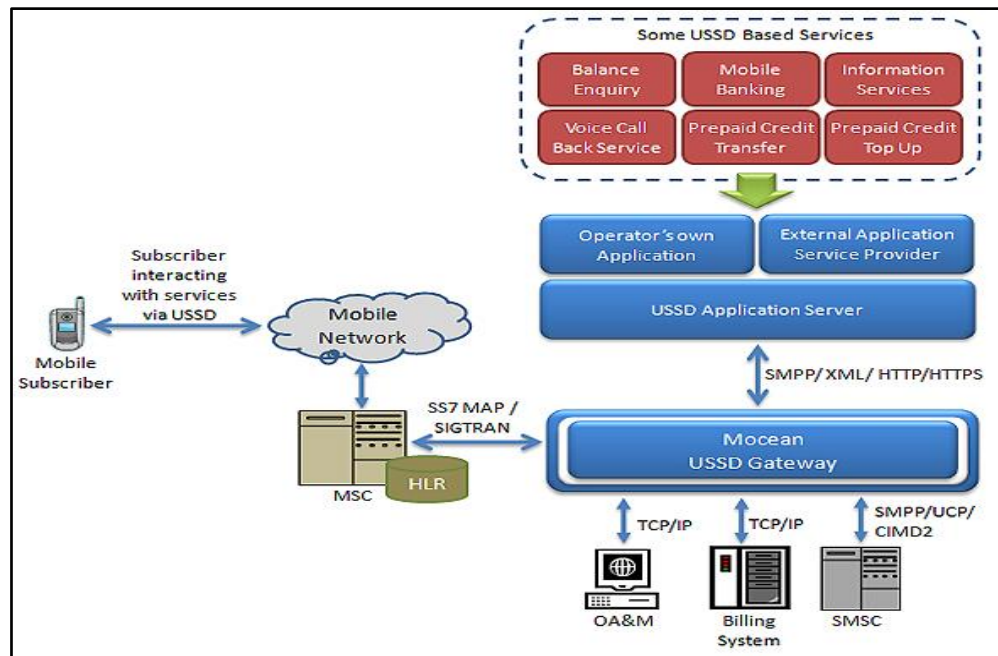


Figure 3 : User interaction with USSD Gateway [2]

An advantage of USSD is that there are no logs saved on the subscriber's phone of the transactions that occurred. When dealing with balances, especially bank accounts, this is essential in providing security to users. Another advantage of USSD is the fact they can be used while roaming. This is because all USSD transactions are routed through the subscriber's HLR. Therefore a subscriber, while in a roaming network, can access every service they have access to in their home network.

2.2. INFORMATION SOURCE

The raw data that was handed over at the beginning of this project consisted of 10,029 text files with over 393,000,000 transactions in total. Each file consisted of one hour of USSD requests and responses. In some cases, the maximum size of the log files would be reached and some data would then spill over into the next file. Each line in each file contains a single

USSD request or a response. Within each transaction there are certain headers. USSD requests contain ten attributes, while USSD responses contain eleven different attributes.

The attributes are

1. Subscriber ID - Contains the users number padded with zeros
2. Service ID - Contains the service id, i.e. indicates which service was used
3. Transaction ID - Indicates whether the transaction is a request or a response
4. Date
5. Time
6. Correlation ID - Links responses and requests together
7. MSISDN - The subscribers number
8. IMSI - Specific only to requests, used to identify subscribers
9. Session ID - Links responses and requests together
10. USSD Content - Contains USSD codes used or the response to those codes
11. Status - Specific only to responses, indicates if there is an error with the transaction but does not specify what the error is.
12. Error Code - Specific only to responses, explains which error is associated with the transaction, i.e. Network Time out error

Each attribute is comma separated, which makes parsing the data into the database significantly easier. It should be noted that not every attribute was stored in the database, some headers were of no value and ignored to reduce space. The Subscriber ID contained the user's MSISDN except there are extra zeros padded between each digit. The Service ID always contains the value "17", this is to indicate that ID number associated with Tango Telecom [3], the company who provided the data. Finally, it was suspected that the Session ID and the Correlation ID performed the same function, to link requests and responses of a 'conversation' together. This was confirmed using a Python script, the Session ID was subsequently removed from the database to reduce space. The remaining attributes were Date, Time, MSISDN, USSD Content, CDRID, Status and the Transaction ID.

2.3. LITERATURE REVIEW

CDR datasets allow a glimpse into the life of subscribers. They can offer an insight into where mobile phone owners go, when they go, who they contact and for how long depending on the data contained within the dataset. The data for this project offers a different view into the life of subscribers. Since it contains only USSD requests and responses, it instead permits an idea of the services which are an intrinsic part of a subscriber's daily life and how these services can interact and influence a subscriber's use of other USSD services.

The area of mobile CDR analysis is one that is becoming ever more popular. There are a vast array of papers and research conducted on the topic. The applications of investigating mobile CDRs varies with each dataset. Mobile CDR datasets are a wealth of information and depending on the dataset, they can offer insights into social interactions and mobility patterns. This introduces the possibility of being able to predict taxi patterns to reduce commuter costs or predicting crime hotspots [4]. The annual Net Mob conference is an example of how popular and useful mobile CDR dataset are becoming.

In reference to this study, one application that would be more applicable would be the creation of a credit rating system where someone who wished to receive a loan could have their credit score calculated by using their mobile credit history. This idea has already been deployed in Chile [5]. This solution offers people who might not have the wealth to deal with banks on a regular basis to apply and successfully receive a loan, potentially giving financial institutions a significant number of new customers and offering capital to people who will go on to build their own enterprises. This is the kind of problem that can be solved through the analysis of mobile data.

Turning to more of a social analysis, Blondel *et. al.* [6] explore numerous social connections between subscribers and how they can affect a wide range of issues. These problems can span from duration of calls, to the amount of average purchases made depending on the region of the country a subscriber is in. Blondel *et. al.* look at different variables and how each one can contribute to the analysis large datasets, i.e. time, location, etc... Each one of these forms of social interaction can provide more insight into the characteristics of a network and its users.

In 2016 there was a mobile penetration rate of 65% with it expected to rise 7% by 2020 [7]. The amount of data that is available to study is increasing every year as the mobile phone penetration rate continues to increase. This will hopefully lead to more studies being performed with new and more innovative solutions being found to problems, like the credit rating system rolled out in Chile. The area of mobile CDR analysis is an exciting area that will lead to surprising developments and trends being discovered [8].

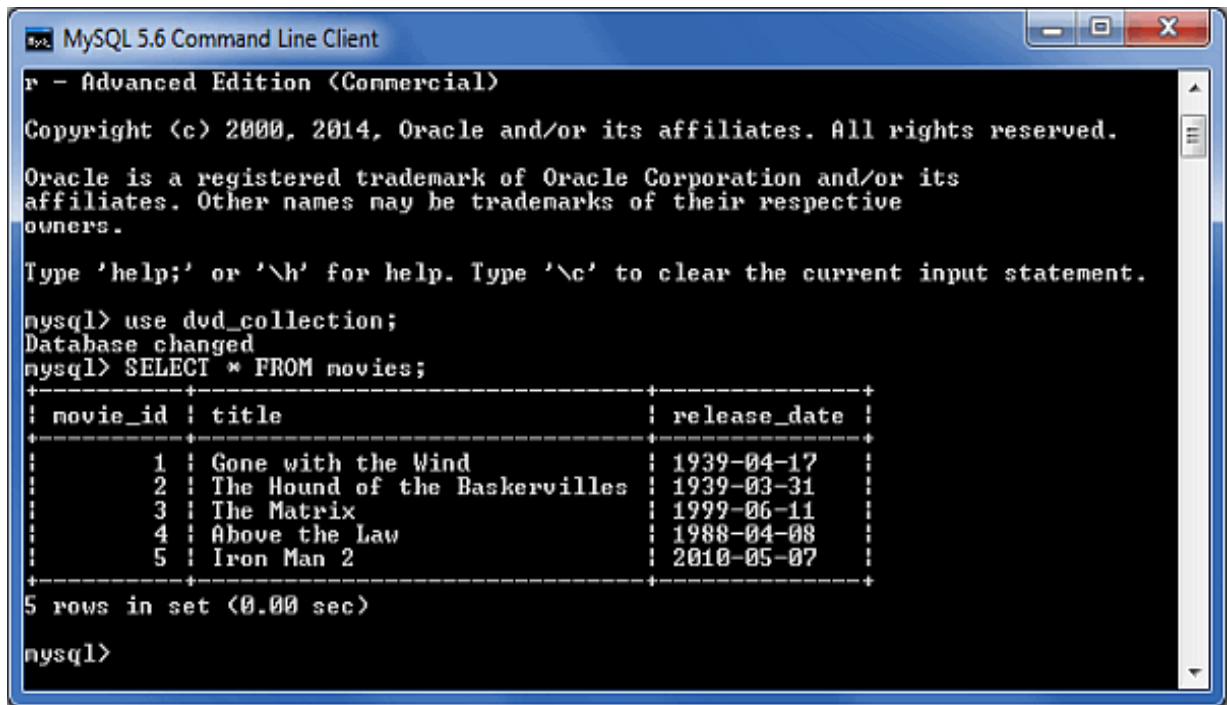
3. FRAMEWORK FOR ANALYSIS

This project can be broken into two major parts, selecting a database and performing analysis on the data stored in this database. When conducting research into what database to choose, the first question faced was: “*MySQL or NoSQL?*”.

3.1. MySQL

MySQL provides a rigid, table based database solution to users. MySQL, a relational database management system (RDBMS), also provides columns and indexes which can be used to perform queries and match specific data stored. MySQL is a popular open source database because it can be supported in most operating system and integrated with most programming languages, i.e. Python, Java, C, C++. Not only this but MySQL can support large volumes of data with up to 50 million rows of data allowed to be stored in the database [9]. MySQL is very useful for situations where you will be dealing with the same type of data, i.e. where attributes of the data do not change.

Figure 4 shows an example of what a MySQL table can look like. MySQL clearly has its advantages and should be considered carefully for any project. However relational databases are not suited to string databases [11].



```

MySQL 5.6 Command Line Client
r - Advanced Edition (Commercial)
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use dvd_collection;
Database changed
mysql> SELECT * FROM movies;
+-----+-----+-----+
| movie_id | title                                | release_date |
+-----+-----+-----+
| 1 | Gone with the Wind                  | 1939-04-17   |
| 2 | The Hound of the Baskervilles      | 1939-03-31   |
| 3 | The Matrix                         | 1999-06-11   |
| 4 | Above the Law                      | 1988-04-08   |
| 5 | Iron Man 2                         | 2010-05-07   |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>

```

Figure 4. A sample MySQL table [10]

3.2. NoSQL

No-SQL stands for “Not Only SQL”. One of the main differences between the two forms of databases is that No-SQL allows for each entry to have completely different attributes to the rest of the data. This is extremely useful for datasets when the exact attributes of the data are unknown in advance or when it is known that the attributes will change. No-SQL databases allow for projects to scale with ease and this is one of the most appealing attributes of choosing a No-SQL database. There are different forms of NoSQL databases. There are two main forms of NoSQL databases, document orientated databases and graph based databases.

3.2.1. Document Orientated Databases

While MySQL implements tables to store data, No-SQL document databases stores its data in a form of *JSON* text. Each entry is treated as one document. One of the most well-known examples of a Document Orientated Database is MongoDB. An example of a MongoDB document can be seen in Figure 5.

```

> db.Test_live.find().limit(1).pretty()
{
  "_id" : ObjectId("58d58a2ec0bb6b2a9c283e07"),
  "CDRID" : "1kaE0UoCi",
  "USSD" : "130#",
  "MSISDN" : "1da227a90f4d955fd8ead4b80afe7df9",
  "Time" : "22:39:50",
  "Date" : "2016-08-31",
  "Trans" : "1"
}

```

Figure 5 : An example of a MongoDB Document

Documents are treated as a whole and are not split based on the key value pairs that the document contains [12]. There are many benefits for using document based databases. Some of these include the lack of complex joins, powerful querying ability and an easy to follow structure for each document. The ability to use MapReduce [13] can also be found in MongoDB. Document based databases are easy to use and quick to get up and running.

3.2.2. Graph Based Databases

Graph based databases are different in the sense that they employ nodes, edges and properties to save and visualise data [12]. Graph databases use different relationships between nodes to find and search for required nodes. Neo4j would be one popular open-source graph database. In Neo4j all relationships are implicitly directional, this can be avoided by ignoring the direction query from the match [14]. Relationships in a graph database always have a beginning and end node. Therefore, when deleting a node a relationship could be accidentally deleted, as there can never be a missing link in a relationship.

Graph databases have their advantages and are used by some massive technology companies such as Facebook, PayPal, Google and LinkedIn [15]. Graph databases are especially useful if it is known that the data being studied is highly connected, e.g. a dataset of actors and the movies they acted in. It allows researchers to easily visualise data and see relationships between different nodes. Figure 6 shows an example of a graph database where the dataset contains actors and the movies they starred in.

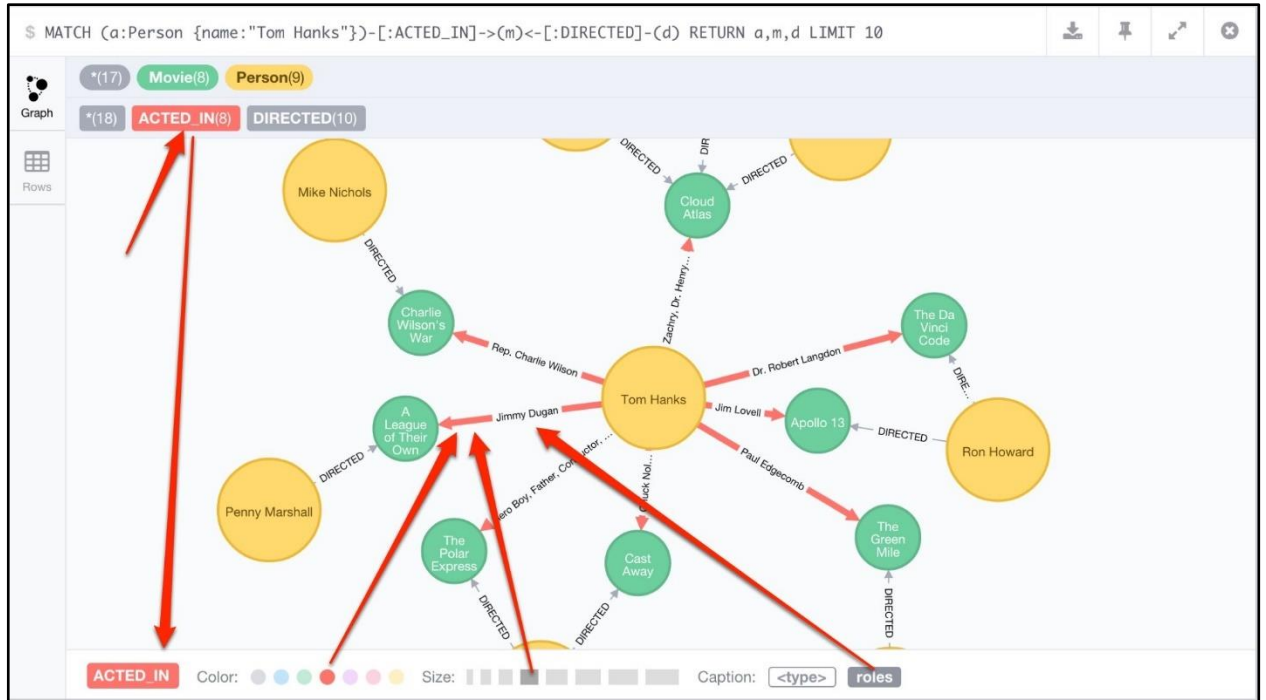


Figure 6 : Example of Graph Database, Neo4j [16]

3.3. SELECTION OF DATABASE

When researching databases for this project, one of the most important factors to consider is the type of data that will be used. In this case the dataset contains only USSD transactions from Chad.

While the dataset only contains USSD requests and responses, there exists a wide variety of different USSD services. Since the aim of this project is to make predictions based on the dataset, these codes must be deciphered to determine their functionality. Therefore, when deciding whether to use an SQL or No-SQL database, No-SQL was chosen as document databases offer more flexibility than the table orientated SQL databases [17] [18]. When selecting the database, the functionality of the USSD codes was not known. Therefore, it was not possible to determine whether the data would be highly connected. This was one of the reasons a Graph orientated database was not selected for this project.

There are many different document databases out there such as MongoDB, Apache Presto, Orient DB and Couch DB to name a few. When comparing them initially, it was concluded that Mongo and Orient should be researched further [19]. It was found that Mongo and Orient

seemed to be highly rated against other document databases with both having similar performance. OrientDb is not only a document database but a graph orientated database as well. This was one of the reasons it was shortlisted as a database for potential use in this project. Both have the ability to support the main programming languages which is a key issue to any project team when selecting a database. After more intensive research, a few issues of concern arose with regards to OrientDb [20]. These issues included instability with releases and the ability of null password to be able to unlock any OrientDb. The reliability of MongoDB and the fact that it is such a widely respected database lead to the decision of its use as the designated database for this project.

One of the main advantages of using a widely popular database such as MongoDB is the amount of available resources online. This makes it much easier to debug and solve the problems which were encountered throughout the project. MongoDB has a lot of online documentation and a significant amount of third party libraries. One example is PyMongo which allows for MongoDB commands to be easily integrated into Python scripts.

3.4. DATA PRE-PROCESSING

Once the process of selecting a database was completed, the next phase was to parse the raw text files into the database. This required separating them using the commas found in each transaction of the text files and then associating the correct attributes to each portion of a transaction.

3.4.1. Parsing of Raw Data into MongoDB

Python is a very useful programming language for file parsing. It contains a lot of in-built methods which help to deal with the alteration of strings and text. These methods include *.split()*, which divides a string into different substrings based on a character passed as a parameter to the function. Another useful function is *.strip()* which removes a specified substring at the start and/or end of a string. Using a combination of all the in-built methods, this creates the ability to target exact substrings. This became a necessity when trying to remove voucher codes, which are attached as part of the USSD code, in order to access the root USSD codes used in transactions.

Using Python syntax, as seen below, Figure 7 demonstrates how every file in a specified directory and its subdirectories can be accessed in just three lines.

```
1. for root, dirs, files in os.walk(root):
2. for FILENAME in files:
3.     #walk the directory and subdirectory and get every file
```

Figure 7: Example of looping through user specified directory

To parse all the data into the Mongo database, a Python script was used. This script had to deal with transactions of varying length and deal with them appropriately by assigning the correct headers depending on their location in the transaction string. The USSD responses and requests have some different attributes, for example the USSD response has a header called Error while USSD request messages do not. However, both contain an attribute called “Transaction” which is a binary number and defines whether the message is a request or a response. Each message is created as a single document in the database and their headers and the value associated with each header is defined by the binary value in the Transaction attribute. This attribute can tell whether the message is a USSD request or a USSD response. This was completed because, as previously mentioned, requests and responses have a different number of attributes.

Some details had to be encrypted to ensure subscribers privacy throughout this research, such as the MSISDN. To complete this, the Python library hashlib [21] was used. The implementation of this is found below in Figure 8.

```
1. if USSDRequestHeaders[count] == "MSISDN":
2.     #for privacy and ethical reasons the numbers in the db must be hashed
3.     i=hashlib.md5(i.encode()).hexdigest()
```

Figure 8: Hashing of MSISDNs for ethical reasons

It was discovered that when the data was first parsed, not all transactions had been completed successfully. By using the Status attribute any response which did not have a value of “0”, transactions that were in error could be detected and discarded. The CDR ID of the transaction in error could then be saved so that the request that initiated this transaction could also be removed from the database. This proved to be successful to a certain extent. Upon further research, it was found that some responses, while having a Status value of “0”, would contain an error message in the USSD code section of the message. To overcome this issue,

a pool of error keywords was built up. Further checks were built into the script which was used to parse the data into the database. While this proved to be successful by not allowing transactions in error to enter the MongoDB collection, it slowed the performance of the script due to the fact that extra checks had to be performed on each response. The end product resulted in a collection of USSD requests and responses which were without error.

To speed up the process instead of inserting each message one by one they are inserted using a bulk insert, which in this case inserted a dictionary of all the messages. Each transaction, whether it be a request or a response is saved into a list of dictionaries. Seen as there are hundreds of millions of individual messages, it is important to try and optimize every script at any opportunity. Optimisation was also a necessity as the average length of some scripts that were run over the course of this project were up to a few days.

3.4.2. Isolating Unique USSD codes

The idea behind isolating the unique USSD codes is that there would be a list of USSD codes which could then be used to match responses to each of these codes. The idea behind this was to strip the USSD codes to just the bare numbers and save that number, e.g. *502*78945# would be saved as 502 in the database as 502 is the actual USSD code while the rest is more than likely just a voucher code. Using Python methods, that have been discussed previously, extra information such as voucher codes could be removed. This would result in only the root USSD code being left and it could then be saved in a MongoDB collection. This proved very useful, for future scripts as this collection can be accessed and all the unique USSD codes could then be placed in an array in a matter of moments. It was also extremely helpful when retrieving the functionality of the USSD codes which will be reviewed in further detail later. Figure 9 demonstrates an example of some of the USSD codes that were saved.

```

{ "_id" : ObjectId<"58036bf4d50e6bb17f999599">, "USSD" : "148" }
{ "_id" : ObjectId<"58036bf4d50e6bb17f99959a">, "USSD" : "158" }
{ "_id" : ObjectId<"58036bf4d50e6bb17f99959b">, "USSD" : "541" }
{ "_id" : ObjectId<"58036bf4d50e6bb17f99959c">, "USSD" : "924" }
{ "_id" : ObjectId<"58036bf5d50e6bb17f99959d">, "USSD" : "233" }
{ "_id" : ObjectId<"58036bf5d50e6bb17f99959e">, "USSD" : "108" }
{ "_id" : ObjectId<"58036bf7d50e6bb17f99959f">, "USSD" : "580" }
{ "_id" : ObjectId<"58036bf7d50e6bb17f9995a0">, "USSD" : "595" }
{ "_id" : ObjectId<"58036bf7d50e6bb17f9995a1">, "USSD" : "715" }
{ "_id" : ObjectId<"58036bf8d50e6bb17f9995a2">, "USSD" : "729" }
{ "_id" : ObjectId<"58036bf9d50e6bb17f9995a3">, "USSD" : "119" }
{ "_id" : ObjectId<"58036bfad50e6bb17f9995a4">, "USSD" : "775" }
{ "_id" : ObjectId<"58036bfad50e6bb17f9995a5">, "USSD" : "153" }
{ "_id" : ObjectId<"58036bfbd50e6bb17f9995a6">, "USSD" : "721" }
{ "_id" : ObjectId<"58036bfbd50e6bb17f9995a7">, "USSD" : "587" }
{ "_id" : ObjectId<"58036bfcd50e6bb17f9995a8">, "USSD" : "525" }
{ "_id" : ObjectId<"58036bfcd50e6bb17f9995a9">, "USSD" : "737" }
{ "_id" : ObjectId<"58036bfed50e6bb17f9995aa">, "USSD" : "717" }
{ "_id" : ObjectId<"58036bfed50e6bb17f9995ab">, "USSD" : "333" }
{ "_id" : ObjectId<"58036c01d50e6bb17f9995ac">, "USSD" : "725" }

```

Figure 9 : Screenshot of the unique USSD codes without any vouchers

3.4.3. Correlating USSD codes to responses

To perform analysis on any set of data, it is required that the analysis be performed by someone who fully understands the data contents and meaning. This meant that work had to be completed to decipher the functionality of each USSD code. Using Correlation IDs, USSD requests and responses can be linked together. The responses can then be written to a file so they can be viewed and the functionality of the USSD codes can be determined. Figure 10 demonstrates how a list of unique USSD codes can be easily collected as mentioned in Chapter 3.4.2.

```

1. def get_ussd_codes():
2.     uscodearr = []
3.     cursor = ussd_codes.find({}, {'_id':0})
4.     #get all USSD codes stored in the collection
5.     for j in cursor:
6.         uscodearr.append(j)
7.     #get your ussd codes and append to an array
8.     return uscodearr

```

Figure 10 : Function which collects all USSD codes

Figure 11 uses regex expressions to match correlation IDs and find the response to a transaction. Even though only one USSD response will be returned it is still required to loop over a MongoDB cursor regardless of its length. Figure 11 allows for one single USSD request and response to be linked together and written to a file. This logic was employed

when attempting to discover the functionality of each USSD code that would have been collected from Figure 10.

```

1. cdr_id_cursor = everything.find({"$and": [{"CDRID":{"$regex":cdr_to_search_for}},
    {"Trans":"2"}]},{'_id':0,'USSD':1}).limit(1)
2. for cdr_index in cdr_id_cursor:
3.     code = g["USSD"]
4.     print g["Time"], " ", g["Date"], " ", i["USSD"]
5.     #The print statement is used to help debug and give progress feedback to the
    user
6.     f.write(code+'\t'+cdr_index["USSD"]+'\n')
7.     #write data to file

```

Figure 11 : Write Response that doesn't contain an error and its code to a file

When the code in Figure 11 was employed on a larger scale, it allowed for the requests and responses for every USSD code to be written to a file. This would result in a file with entries like the ones found in Figure 12. Figure 12 illustrates how the codes and responses can be joined together and it then allows for the interpreting of what service each code provides.

```

*125*1# Your balance will be sent to you via sms.      Your balance will be sent to you via sms.

*701*0262630666#      233262630666 has been notified of your call me back request.      233262630666 has been notified of your call me back request.
*701*0269015990#      233269015990 has been notified of your call me back request.      233269015990 has been notified of your call me back request.
*701*0262454712#      233262454712 has been notified of your call me back request.      233262454712 has been notified of your call me back request.
*701*0560983255#      233560983255 has been notified of your call me back request.      233560983255 has been notified of your call me back request.
*701*0262454712#      233262454712 has been notified of your call me back request.      233262454712 has been notified of your call me back request.
*701*0560983255#      233560983255 has been notified of your call me back request.      233560983255 has been notified of your call me back request.
*701*0560518505#      233560518505 has been notified of your call me back request.      233560518505 has been notified of your call me back request.
*701*0560983255#      233560983255 has been notified of your call me back request.      233560983255 has been notified of your call me back request.
*701*266656126# 233266656126 has been notified of your call me back request.      233266656126 has been notified of your call me back request.
*701*0266664214#      233266664214 has been notified of your call me back request.      233266664214 has been notified of your call me back request.

*130# Dear Subscriber your request is accepted.      Dear Subscriber your request is accepted.
*130# Dear Subscriber your request is accepted.      Dear Subscriber your request is accepted.

```

Figure 12 : Screenshot of file with all code functionality

With 182 unique USSD codes, there were a wide range of services on offer. Through USSD codes, subscribers of the network can access their Bank account, purchase insurance as well as many other services. These USSD codes permit users who may not own smartphones the ability to access services that most people in Europe would use mobile applications for instead.

3.4.4. DISCOVERING USSD CODE USAGE

Once it was discovered the service each USSD code performed, the next step in the project was to discover which codes were the most popular. This information was needed to indicate which services would provide sufficient data on which analysis could be performed. MongoDB has an in-built function which allows you to count the number of matches to a query, `.count()`. By combining regex expressions and this method, it became possible to get the percentage value for each code. Figure 13 shows the percentage that each code represents.

1. *125# - Your balance will be sent to you via sms. - 33.55%
2. *130# - Dear Subscriber your request is accepted. - 31.49%
3. *126# - You have bundled Successfully! You now have 87 MB Valid till 05-03-2016 23:51. Thank you! Click on the link below to play some action games. - 7.346977%
4. *202# - You have 140.92 mins for all Airtel calls 197 SMS valid until 05-03-2016 - 6.32%
5. *477# - You have successfully subscribed to Airtel's Sika Kokoo. Enjoy free Airtel calls free Airtel Money transfer and free WhatsApp - 5%
6. *500# - Error Code - 3.02869%
7. *701# - 233262630666 has been notified of your call me back request. - 2.76%
8. *703# - Dear Customer your number is 233560909737. - 2.55%
9. *554# - Dear Subscriber your request is accepted. - 2.22%

Figure 13 : Percentage for the most popular codes

It is interesting to note that the two most popular codes make up almost two-thirds of all USSD transactions over the six months. Both of these codes deal with a subscriber's mobile credit. '*130#' is a credit top up which allows subscribers to increase their credit balance. '*125#' sends the users a text with their credit balance. Due to the fact that the balance is sent via SMS, access to a subscriber's balance is therefore not possible and does restrict some forms of analysis which will be discussed later on.

3.5. REAL TIME DATABASE

The data that was provided had been recorded on an hourly basis with approximately fifty thousand new transactions recorded each hour. It was decided that a real time monitoring of the data should be added so analysis can be performed on all data including the most recent USSD requests and responses.

The aim of this task is to parse each new file, when it is created, and add the new USSD transactions to the MongoDB collection. Once this phase has been completed, the next aim

is to isolate all the ‘Call Me Back’ requests and put them in a separate collection. This is done so a csv file of all ‘Call Me Back’ requests can be generated for analysis later.

Watchdog is an event-drive Python library which monitors directories and the files within them [22]. This library was implemented in a Python script to monitor a user specified directory for any new files created. Once a new file is created, another script which will parse the data into MongoDB is called. Figure 14 shows the ability for other tasks to be called when a new file is created.

```

1. def create_process(self, event):
2.     folder='\\'.join(event.src_path.split('\\')[0:-1])
3.     file = '\\'.join(event.src_path.split('\\')[-1:])
4.     print("File "+folder+"\\ "+file)
5.     print("Test File "+event.src_path)
6.     fill_database_no_errors_user_input.fill_db(folder+"\\ "+file)
7.     isolate_one_code.fill_db(folder+"\\ "+file)
8.     os.remove("C:/Users/Conor/Documents/FYP/FYP_PredictingHumanBehaviour_Usin
g_CallDetailRecords-master/gephi_data.csv")
9.     os.system('mongoexport -d FYP_Airtel_Storage -c call_gephi_demo --type=csv
--fields "Source","Target","Time","Date" -o gephi_data.csv')
10.
11.
12. def on_modified(self, event):
13.     print("File modified")
14.     #self.create_process(event)
15. def on_created(self, event):
16.     self.create_process(event)
17. def on_deleted(self, event):
18.     print("File deleted")
19. def on_moved(self, event):
20.     print("File moved")

```

Figure 14 : Code used to monitor directories

When other events are triggered a new task could be added. This is definitely work that could be completed in the future, e.g. removal of data from database if a file is removed. The above figure shows the functionality of the Python script at present. The only event that triggers other scripts is when a new file is created. When this event occurs two other functions in other Python scripts are called. To fill the database with all data from the new file, *fill_db*

from *fill_database_no_errors_user_input.py* is called. This function takes in the new file as a parameter and reads its contents line by line. *fill_db* deals with the issues of different headers and removing transactions in error. Once all the new transactions have been properly saved in MongoDB, the next function (*fill_db* from *isolate_one_code.py*), manages to isolate all ‘Call Me Back’ requests which can be found in the new file.

Using the Python OS library, the old *csv* file which contained all ‘Call Me Back’ requests, was deleted and replaced with an updated version. Line 9 of Figure 14 demonstrates the use of *mongoexport* which outputs the data to a *csv* file. This now allows social analysis to be conducted on the most recent data using software tools such as Gephi, a graphing software tool, and MATLAB to name a few.

3.6. TECHNIQUES EMPLOYED TO HANDLE BIG DATA

This project contains 10,026 unique files with an approximate average of 50,000 lines per file and a total of 393,274,512 different transactions each to be checked and parsed. One of the major problems associated with this project was the time taken to run some Python scripts. To overcome this, scripts needed to be as efficient as possible by making use of some classic techniques known to overcome this challenge.

Multithreading allows for multiple jobs to occur at once in a computer program. Each thread has its own program counter, variables and stack so as to know which commands have been previously executed [23]. Concurrent processes speed up the operation. When possible, multithreading was incorporated into scripts to reduce the time taken to execute a script. Multithreading did lead to an increase of computational usage to reduce the execution time. This trade off was a more efficient use of resources as computation usage would always be quite low when Multithreading was not employed. This means that there were resources available to use. This is one of the reasons that multithreading was seen as a solution to the problem of long execution times.

Figure 15 shows a Python function which was created to allow multithreading to occur. This could be reused just by changing the target, i.e. which function you wished to multithread, and the arguments for that target. In this case, the amount of times a user topped up was being queried resulting in a single thread being assigned to each user. Each thread was saved

to a list that was iterated over to start each thread. A limit was placed on the number of threads that could be active at once to ensure the program did not crash, the `.join()` method waits until all active threads have completed before moving on. This function allowed multithreading to be used on any script easily and ensured run time was reduced.

```

1. def start_process():
2.     count = 0
3.     index=0
4.     threads=[]
5.     num_limit = 20
6.     users_arr = get_users()
7.     print 'Starting.....'
8.     for i in users_arr:
9.         t= threading.Thread(target = get_TopUp_Num, args=(i,))
10.        threads.append(t)
11.    print 'Finished Queuing Threads'
12.    for thre in threads:
13.        index = index+1
14.        thre.start()
15.        if(index%num_limit==0):
16.            for x in threads[count:index]:
17.                x.join()
18.                #print threading.activeCount()
19.            print 'Joining Threads',index
20.            count = index

```

Figure 15: Function to allow Multithreading

Exception handling was also used to reduce run time. To fully understand the error, the concept of the large component must be introduced. The large component of a network is the biggest cluster of nodes which are connected together. In this particular example where the number of credit top ups for users in the large component were being counted, this was employed. The amount of credit top ups were being saved in a Python dictionary with each user's MSISDN as a key. Originally a quick *if* statement was used to check if the user was part of the large component of the network. Considering there were over a hundred thousand members of the large component this became quite time consuming. The optimal solution was to initialise the dictionary with the subscribers, who were part of the large component, and to handle `KeyError` exceptions. These errors would only occur when a user was not part

of the analysis and therefore not saved as a key. This reduced the run time of the programme from days to a couple of hours.

4. ANALYSIS

This chapter will investigate the data that has been pre-processed in Chapter 3. It will attempt to analyse the relationships between subscribers and see how these relationships can affect their interaction with other USSD services.

4.1. SOCIAL ANALYSIS

The following subchapter will investigate social ties between subscribers and attempt to perform analysis on these relationships using software tools such as Matplotlib, MatLab, Excel and Gephi. In this dataset, which contains only USSD requests and responses, there is usually only one subscriber involved in each transaction, e.g. Credit Top Ups, Bank Account Balance Check, etc. To perform analysis on social ties requires information on at least two subscribers. For this reason, the USSD code that the majority of analysis was performed on was the ‘Call Me Back’ service. The ‘Call Me Back’ request allows for the investigation of ties between subscribers instead of the usual interaction between a service and a user. In each request, there are two MSISDNs contained in the transaction. One is found under the header named ‘MSISDN’ while the other is found appended onto the USSD code found under the ‘USSD’ header. The recipient of the code, found in the ‘USSD’ header, should also have their number hashed when performing analysis. This hashing operation is completed before the data is imported into Gephi, a software tool discussed in the next section. Figure 16 shows an example of a MongoDB document containing a ‘Call Me Back’ request.

```
> db.No_Errors_Data.find(<{"USSD":{$regex:/^*701/}}>).pretty().limit(1)
{
  "_id" : ObjectId("58c28fcec0bb6b0f10a103ad"),
  "CDRID" : "1nzZIje0-",
  "USSD" : "*701*0262630666#",
  "MSISDN" : "7bc65f6f4342d49242514a50ce53d71d",
  "Time" : "23:51:00",
  "Date" : "2016-02-29",
  "Trans" : "1"
}
```

Figure 16 : A sample ‘Call Me Back’ request

4.1.1. Network Visualisation and Analysis

The open-source software tool Gephi was used to perform network visualisation and statistical analysis on the network. Gephi has a wide suite of tools that allows for data to be

recorded on a range of areas from topology to degree range. Through Gephi, networks can be created by hand or by utilising an import function. This import function takes data straight from a graph database or from a *csv* file. Each user in the network is treated as a node while a ‘Call Me Back’ transaction between two users would be treated as an edge between two nodes. This is the process that allows the network to be visualised.

Gephi can also export graphs as a *csv* file. This proved to be very useful when calculating statistics such as weighted degree for each subscriber given the result could then be exported to a *csv* file. After, this new data could be imported back into MongoDB.

As mentioned previously, the large component of a network is the biggest cluster of nodes which are connected together. Gephi was used to isolate the large component of a network. Filtering out the rest of the network helps remove noise or smaller clusters which would not have enough data on which to perform significant analysis.

4.1.1.1. Network Decay

Network decay experiments are often performed to see how systems react to different attributes and how the network will decrease when various thresholds are increased [24]. Investigations into the proportion of the network remaining when the threshold for edge weight is increased were completed. Figure 17 shows the results of this experiment. The edge weight in the network represents the number of times one user has initiated a ‘Call Me Back’ request to the same user. This was performed to see how many ‘strong’ relationships exist in the network. The larger the edge weight, the stronger the social tie between the two users. By increasing the minimum edge weight for an edge to be present, the ‘weaker’ relationships will be filtered out. When this threshold increases, the number of edges that remain in the network should decrease, except in the extremely unlikely situation where each relationship has the same edge weight.

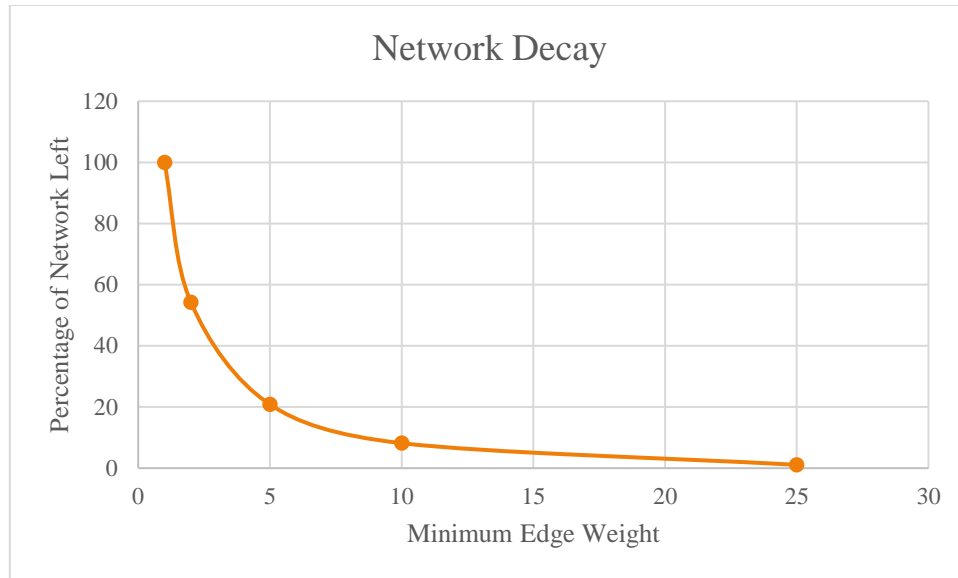


Figure 17 : The Network Decay as threshold on edge weight increased

Figure 17 shows that an exponential graph is describing the system. It can be inferred that while there are strong social ties between users, they only represent a minority of the overall system. This means that many subscribers in this telecoms area only lightly interact with others through this medium. The data from Figure 17 is spread over six months, therefore under 20% of the relationships formed through this USSD code have an average of one ‘Call Me Back’ request per month. The main form of communication between subscribers can be assumed to be the usual telecommunications services, i.e. SMS or voice calls instead of USSD. USSD would not be seen as a ‘normal’ way of communication so it is interesting to note its use as an alternative to the mainstream forms of communication within a telecommunications network.

4.1.1.2. Investigation Through Community Separation

Gephi is a very powerful software application which allows for the ability to divide a network into different communities based on who they interact with inside the area of that telecommunications network. Figure 18 shows different communities. Each community is represented by a different colour. This is done through the use of a modularity tool.

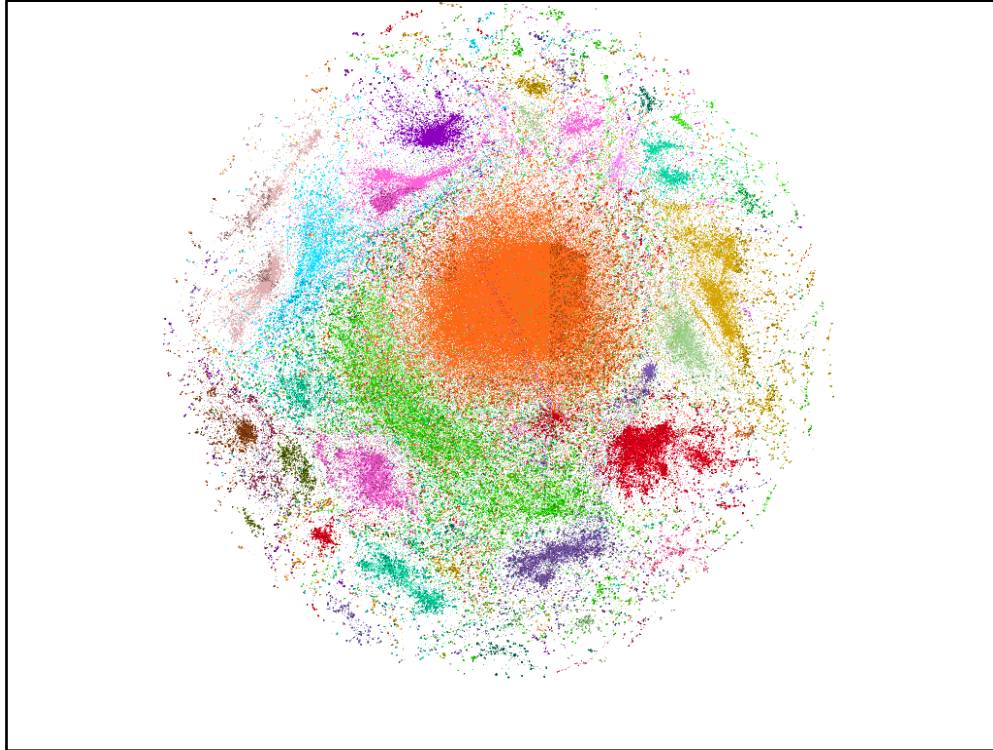


Figure 18 : Communities in the network

The modularity tool in Gephi can be passed a float parameter that can either alter the number of different groups in a network. When the default parameter of one was passed, there were hundreds of diverse communities. It was decided, to better highlight the difference in community activity, the amount of communities would be reduced to a more manageable number. Figure 18 displays a network with twenty-one unique groups of subscribers.

When Machine Learning packages were implemented on the dataset, the community in which a user was situated was passed as a parameter to develop some of these predictive models. These models will be discussed in further detail in Chapter 4.2.4.

Dividing a network based on different communities allows for analysis to be performed on a per community basis. Trends can then be studied and the different characteristics of each community can be examined. Figures 19 and 20 demonstrate the fluctuation in credit top ups and ‘Call Me Back’ requests in the various communities. The average credit top up per community is plotted in Figure 20 to help demonstrate which communities are more ‘active’ than others. The same is done in Figure 19 to show the average ‘Call Me Back’ Requests per community.

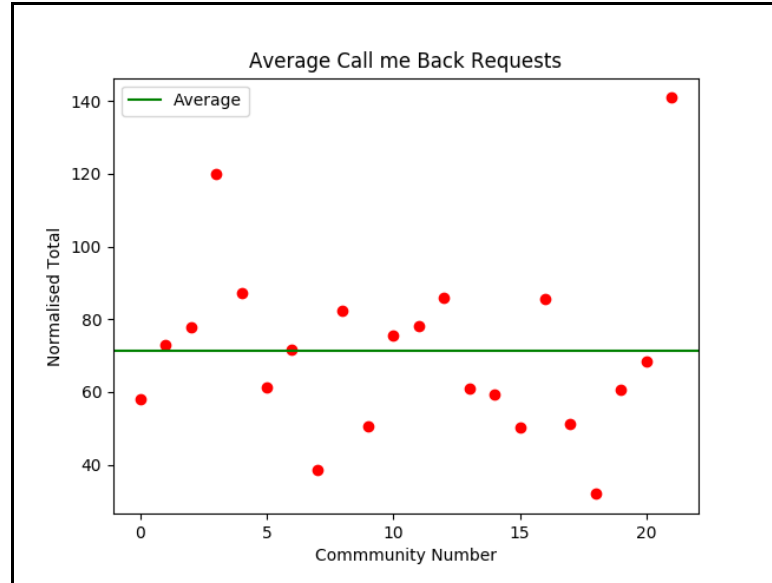


Figure 19 : Average Amount of 'Call Me Back' Requests based on Communities

Figure 20, demonstrates the average amount of credit top ups per community. It can be seen from the two graphs that ~50% of the communities, if they are above/below the average for credit top ups they will be the same for 'Call Me Back' Requests. This would imply that there is a relationship between the amount of times a community tops up and how often they initiate the 'Call Me Back' Request. This will be confirmed when examining the correlation between variables in the development of predictive models in Chapter 4.2.4.

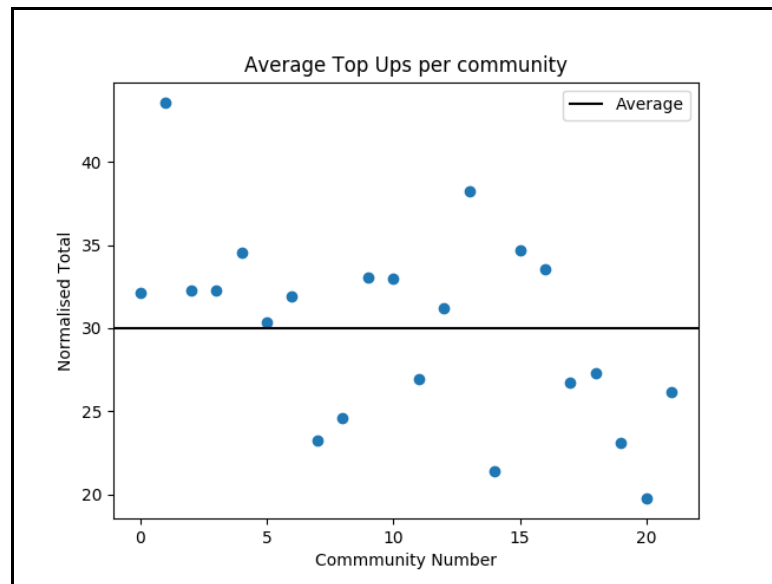


Figure 20 : Average Amount of Credit Top Ups based on Communities

4.1.1.3. Degree Distribution

When calculated, the mobile network has an average degree of 1.332. On average each subscriber who uses the ‘Call Me Back’ service contacts just over one other unique user in the area. This is a vast difference to the weighted degree of each user. The weighted degree of each user was calculated at a value of 14.879. This is substantially higher than the non-weighted degree. This means that on average a subscriber who uses the ‘Call Me Back’ request, will contact approximately one other person with it. However, between the two subscribers in this averaged relationship they will invoke the service approximately 15 times. It must be noted that, as mentioned in Chapter 4.1.1.1, this data is spread over six months which only gives an average of this code being invoked once every eight days for every ‘relationship’. Although this can’t be considered a daily part of a subscriber’s life, it offers an interesting insight into the social relationships of users in this mobile network and how they communicate to each other.

4.1.2. Fitting of Power Law to Data for Analysis

The power law states that a power law relationship exists when a relationship exists between two variables and a change in one component forces a variation in the other by a factor of a power [25]. The most common example of this is when the length of a square is doubled, the area of the square is increased by a factor of four. There are many examples of distributions which follow the power law, the distribution of website visits is one such example. This means that a small number of websites receive the majority of website views while the majority of websites receive only a handful of views [26].

Power laws, when plotted would then look similar to an exponential graph. Plotting a power law relationship like this can make it difficult to visualise how well a fit the Power Law is. This problem is solved by transforming the power law from an exponential graph to a straight line [25]. This is done by plotting the log of both axes as seen in Figure 21.

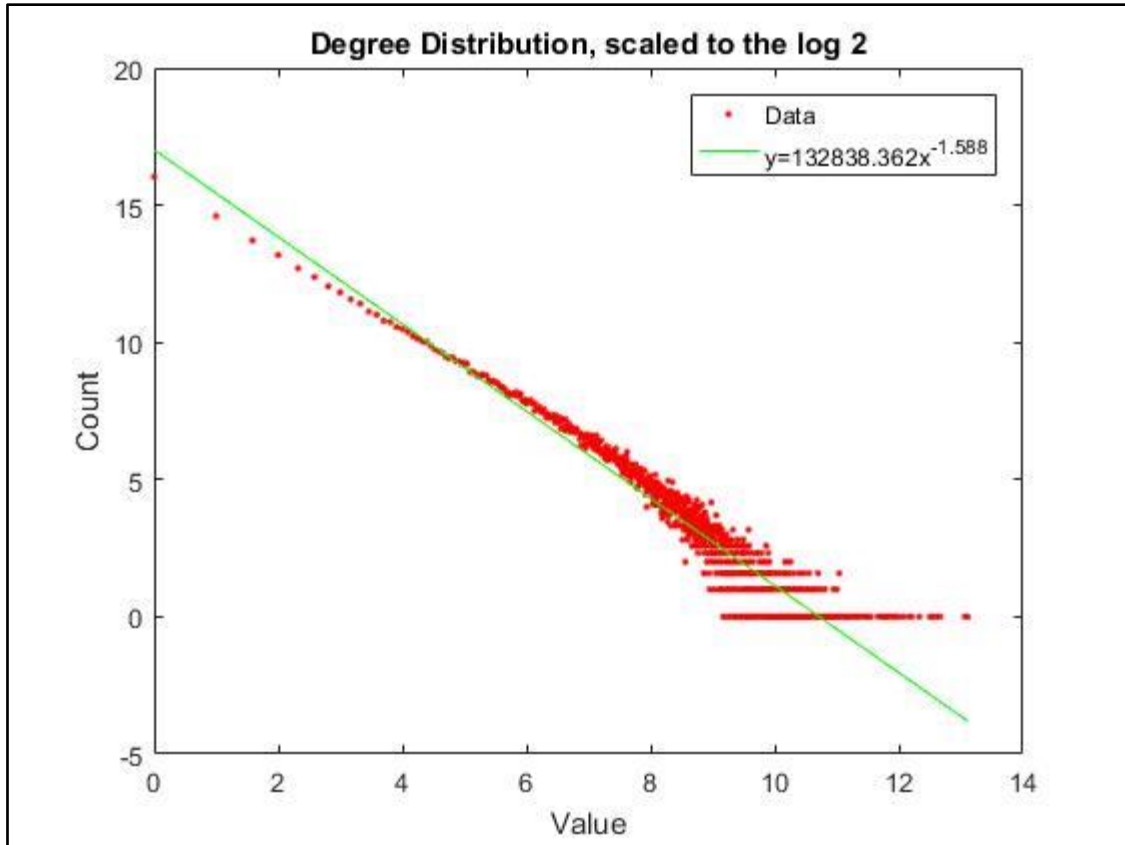


Figure 21: Power Law Distribution of 'Call Me Back' Requests

Figure 21 graphs the number of users who initiate x amount of 'Call Me Back' requests. To demonstrate the power law relationship as a straight line, both the x and y axis were graphed on a scaled \log_2 basis.

A scale free network is one in which the degree distribution follows the characteristics of the power law [27]. Using the evidence from Figure 21, it can be said that the network within this project can be classified as scale free. It can be seen that the majority of subscribers have a small degree. At the same time there are a small minority which have an extremely large degree. These results are similar to the wealth distribution which have been seen in the Pareto Principle [28]. The Pareto Principle, named after an Italian economist, noticed that 80% of land was owned by only 20% of the population. A similar observation can be made in this dataset. The majority of 'Call Me Back' Requests are initiated by a small number of users in the network.

$$N = \frac{A}{x^\alpha}$$

The above represents the equation for the Pareto distribution, where N represents the number of people with wealth larger than a certain income limit x , and A and α are constants [29]. This equation can be applied in this context to calculate the amount of subscribers who use the ‘Call Me Back’ Requests above a certain number of times. The constants needed to complete this calculation can be taken from the legend in Figure 21.

The tail of the data in Figure 21 skews the slope of the data. To get a more accurate slope and better estimate of constant values for the equation shown above, the Power Law could be split into two. This would remove reduce the skewedness that the tail end of Figure 21 introduces to the results.

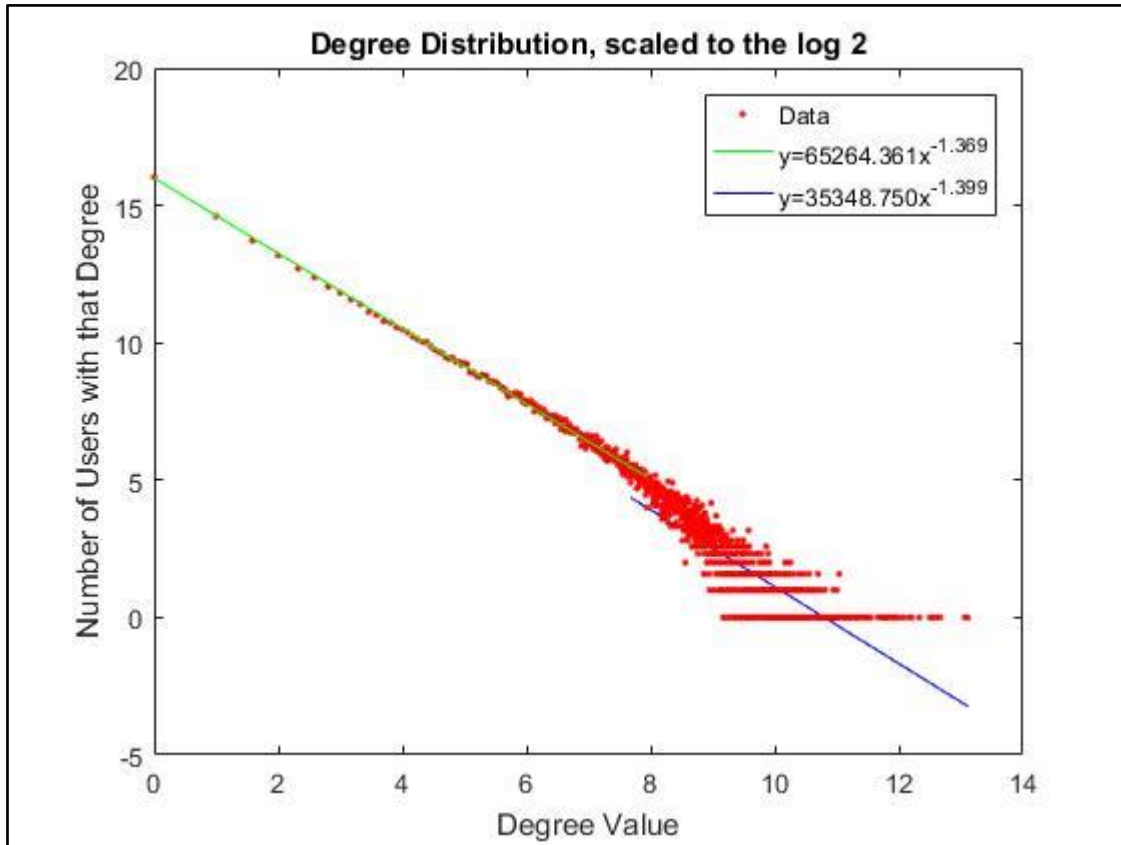


Figure 22: Split Power Law

Figure 22 shows the degree distribution of subscribers divided into two. The bottom half, i.e. the blue line, does not represent a very good fit and could be discarded. The top half of data ranging from 2^4 all the way to the max value ($\sim 2^{16}$) is represented by the green line. The green line almost perfectly fits the upper range of data. These new constants which are in the

legend in Figure 22 could then be used in the equation for the Pareto distribution to calculate the number of users above a set limit for degree distribution.

In further examinations, the credit top up distribution is explored to see if it fits a power law distribution also. Figure 23 shows a power law graphed in an attempt to fit the credit distribution. While the power law fits the tail end relatively well, the same can't be said about the head. The idea of a curved power law has been explored in previous research, one example is *Seshadri et al.* [30]. It is clear that the credit distribution is not a good a fit for the power law as the top up distribution. It could be said that the distribution found in Figure 23 is an example of what would fit the PowerTruck model developed by *Seshadri et al.* [30]. This Double Pareto Log Normal distribution matches the distribution found in the credit top ups of this dataset. This is an area for future work and exploration that has the potential to build on the work completed by *Seshadri et al.*.

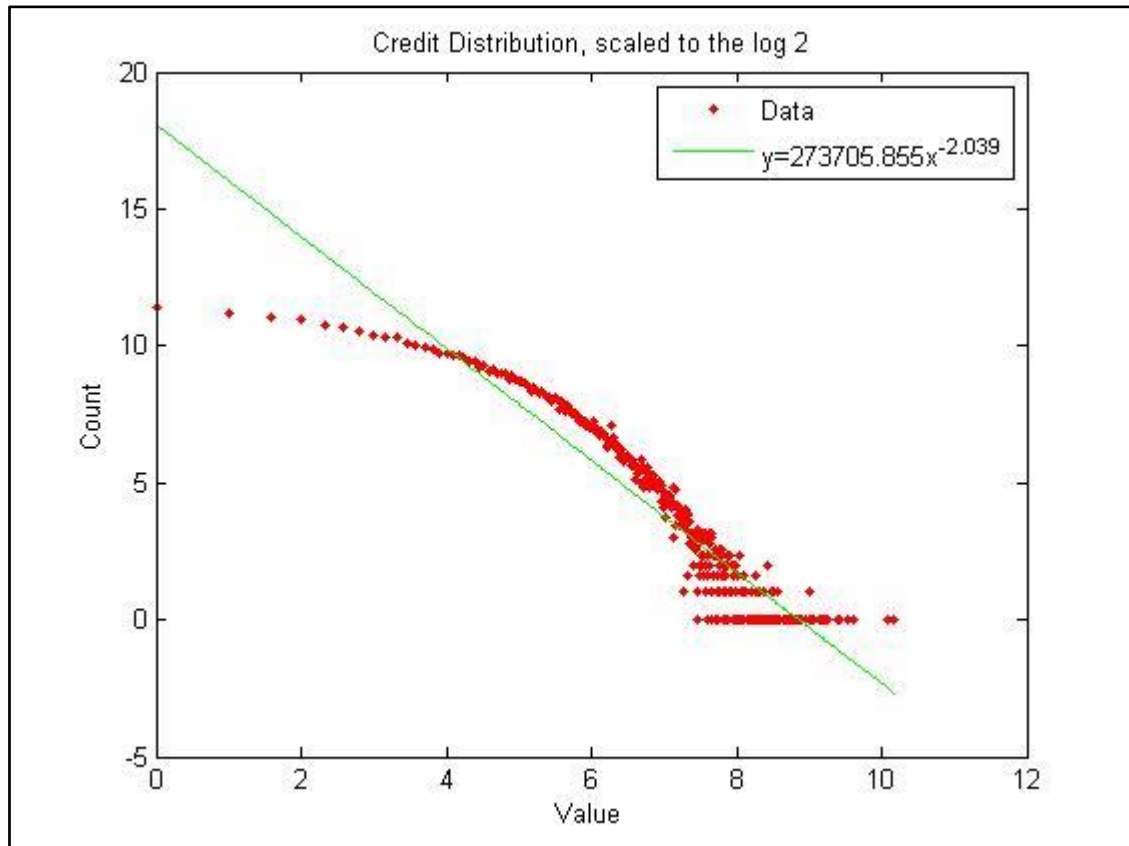


Figure 23 : Power Law for Credit Top Ups

4.2. ANALYSIS OF WEALTH AND ITS DISTRIBUTION

This subchapter aims to analyse the wealth of users by looking at the number of times a subscriber tops up their mobile credit balance and how their social interactions with others in the network can impact on this. This project also assumes one set rate for all credit transactions as no exact figure could be found or was given.

4.2.1. Principal Component Analysis

Principal Component Analysis is the reduction of dimensionality in a dataset [31]. This is done to visualise a dataset on a two-dimensional plane. PCA tries to combine different attributes to represent as much of the data as is possible. Weka [32], an open-source Machine Learning software tool, was employed to perform PCA on a *csv* file version of the dataset. This file was created using the '*mongoexport*' function.

Figure 24 below shows the number of times a user tops up their credit compared to a mix of the number of times a 'Call Me Back' request was initiated/received. The x axis is the number of times a subscriber's credit balance was topped up, while the y axis is represented by the equation below:

$$(0.707 * out\ degree) - (0.707 * in\ degree)$$

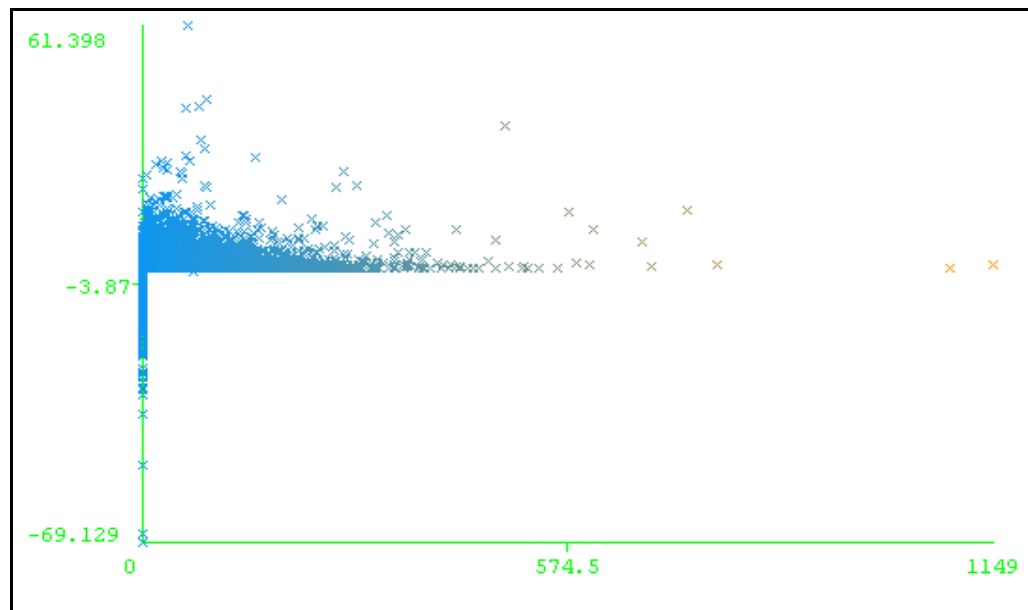


Figure 24 : PCA Graph

Figure 24 shows there is a stronger correlation between number of credit top ups and number of ‘Call Me Back’ requests initiated rather than the number of received. The right angle seen in Figure 24 suggests that the higher the number of ‘Call Me Back’ requests received, compared to the number initiated, the less likely a user is to top up their credit balance. The figure also shows that the users who have a higher number of credit top ups tend to have a similar amount of ‘Call Me Back’ requests initiated compared to the requests they have received.

4.2.2. Monthly Distribution

Some of the USSD codes in the original dataset represented only a fraction of the original dataset, many USSD codes founds were well below 1%. In these cases, there wasn’t enough relevant data to perform any meaningful analysis. Attention was then diverted to codes that formed a larger proportion of the data. Some of these included Credit Top Ups, ‘Call Me Back’ Requests and Data Bundles. The codes with a larger percentage can be found in Figure 12.

Figures 25 and 26 below plot the monthly usage of credit top-ups and ‘Call Me Back’ Requests, ranging from Month 1 (March) to Month 6 (August). Please note that there is only data from one MSC for the month of August compared to the rest where there are two months. To even this out the number for August was multiplied by two. Furthermore, the month of February was ignored because the dataset contained less than a week of data from this starting month.

The monthly credit graph in Figure 25 shows an interesting dip in the month of May. Without this trough, the graph would be close to a straight line. This makes the month of May an interesting case to study further. After more research was conducted into the activities that occurred in Chad during May 2016, it was discovered that the Presidential elections were held in that month. The same leader has ruled Chad for the past twenty-seven years. Protests have occurred during the year of elections to demand a new leader. On the day of elections all mobile internet, fixed internet connections and SMS messaging services were severed [33]. This would explain a dip in the usage of USSD codes on what would have been a very heavy day for the telecommunications network.

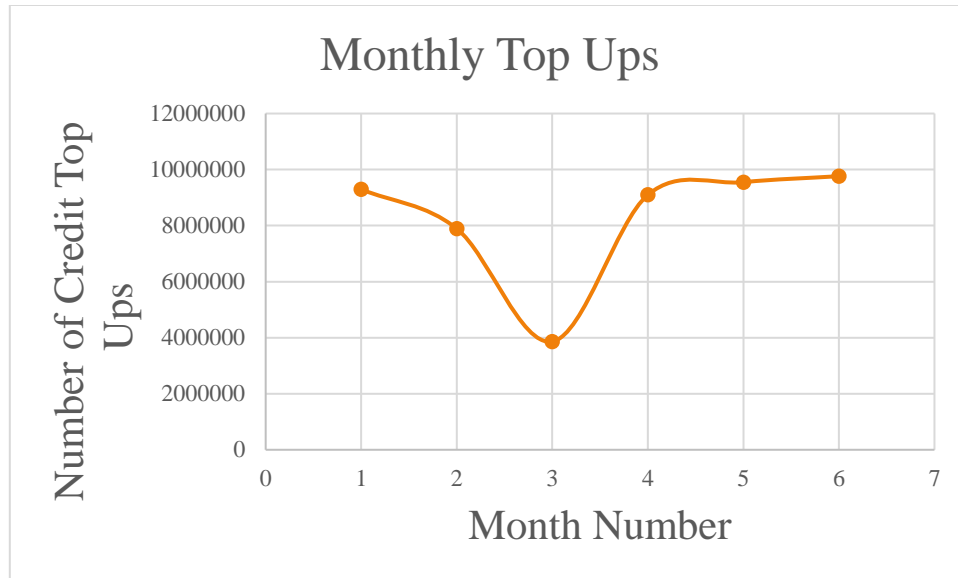


Figure 25 : Monthly Credit Tops

The monthly ‘Call Me Back’ requests have a similar trough in the data, found in Figure 26, except it is for exactly one month later. It could be explained that since credit top ups were low in the previous month and the fact they returned to normal the following month, there was no need for the ‘Call Me Back’ request to be used since subscribers now had credit.

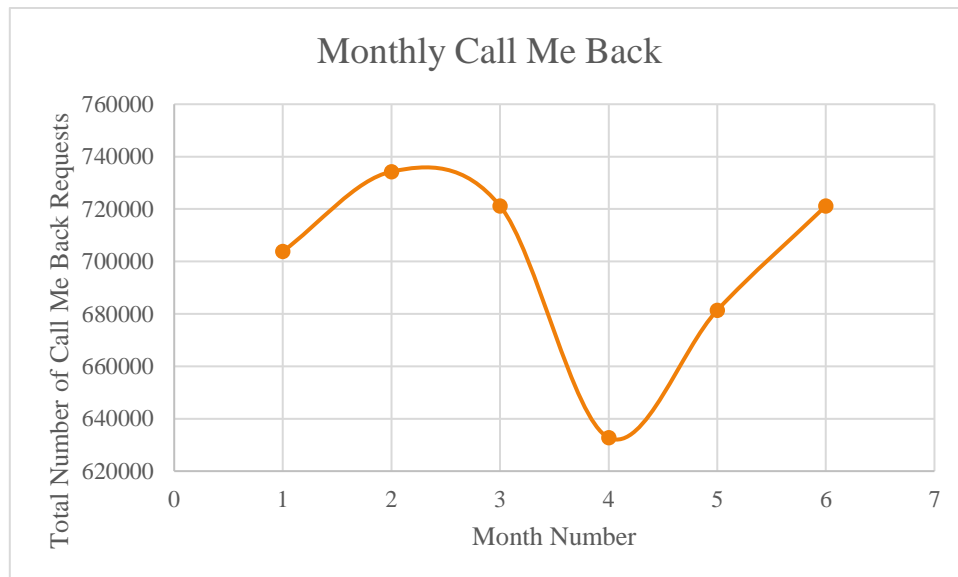


Figure 26 : Monthly ‘Call Me Back’ Requests

4.2.3. Categorisation of Users into Wealth Brackets

This section of the project hopes to define brackets of wealth and based on a user's interaction with the credit top up code place subscribers into these income brackets.

Figure 27 demonstrates the distribution of the USSD credit top up service. This exponential graph shows the number of users who topped up by 'x' amount of times. Many users top up by a small number over the course of six months. As the number of credit top ups increase, the number of users who top up by that amount decreases rapidly. Due to this fact, the distribution of credit top ups can't be described as normally distributed. This resulted in the assumption that 99.7% of the data is within three standard deviations from the mean not being applicable. A new solution to divide users into different wealth brackets then had to be created. It was decided to use percentile scores.

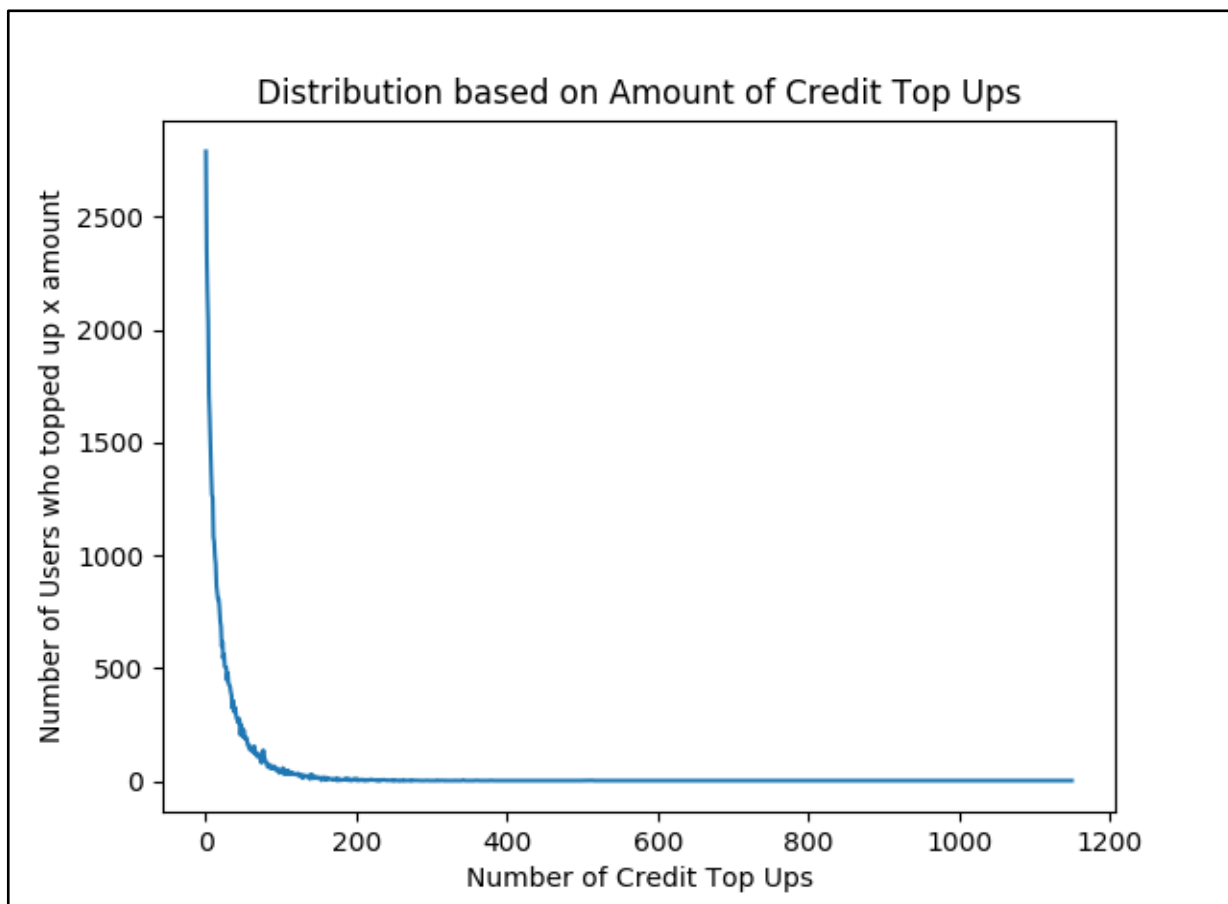


Figure 27 : Distribution of Credit Top Ups

By importing Python's statistics library, a percentile score could be calculated for each user in one line. Then based off this score, they can be assigned to a wealth bracket. Rules were assigned to divide the different classes of wealth. The classes were divided at each interval of twenty percent with the classification going from 'Poor', 'Below Average', 'Average', 'Above Average' to 'Rich'. The wealth distribution seen in Figure 28 mirrors the exponential graph seen in Figure 27.

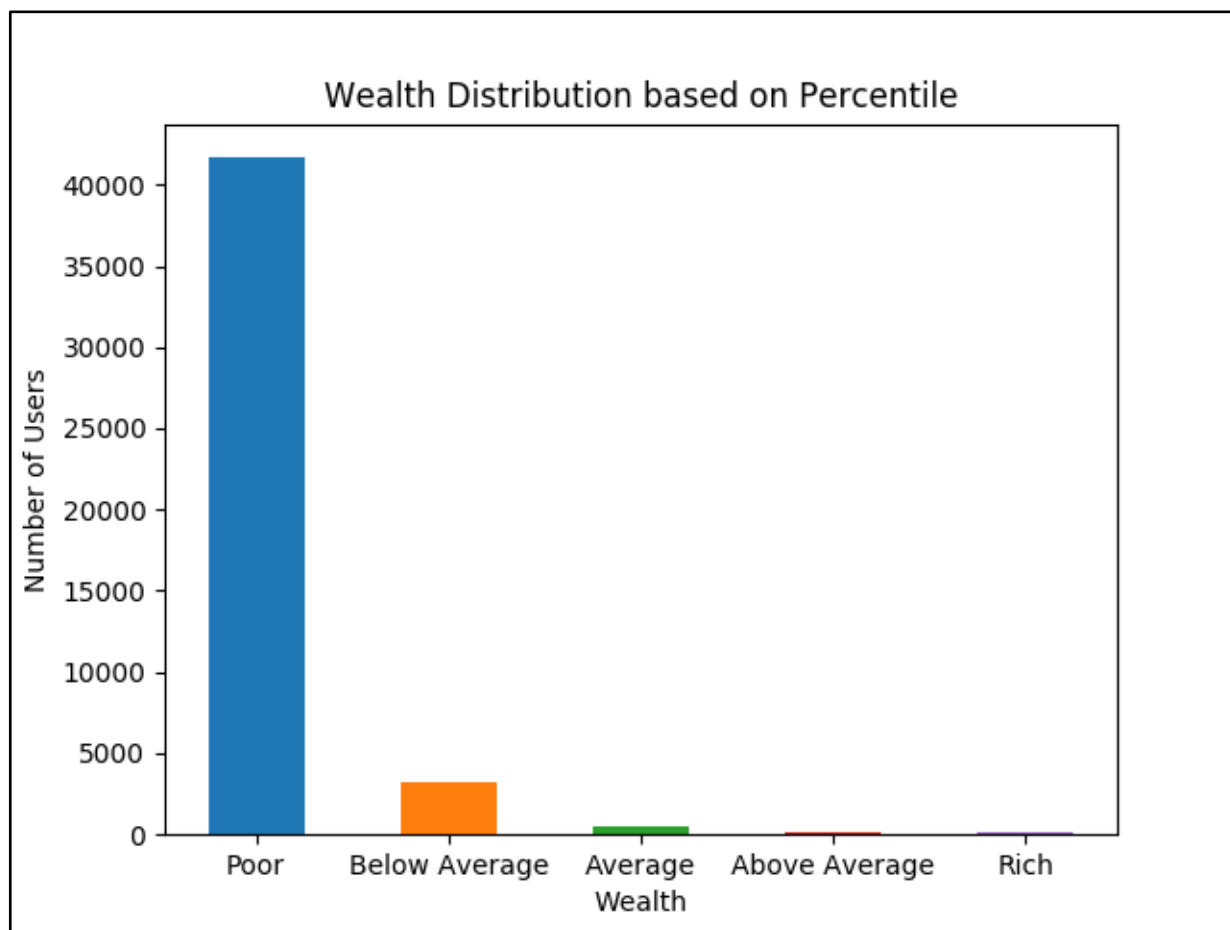


Figure 28 : Wealth Distribution

The majority of users are classified as being 'poor'. The next largest wealth bracket is 'below average'. This sorting of people demonstrates the number of subscribers that are classified as 'poor', more than 90%. The most recent data on poverty in Chad sets the poverty rate at 46.70% [34] and an average GDP per capita at \$1030 [35]. Chad is clearly not a country with a resounding amount of wealth and therefore Figure 28 is understandable. Even assuming a \$1 top up charge, it can be assumed that most subscribers would not be able to

divert ~33% of their daily expenses away from basic necessities on the scale that people in the ‘Rich’ wealth bracket are able to.

Therefore, it can be concluded that credit top ups can be effectively used to divide a network of users into brackets of different wealth as a means of visualising the spectrum of prosperity.

4.2.4. Predicting Subscribers wealth

Scikit-Learn [36] is a Python machine learning package with a wealth of different algorithms that can be used when imported. It was hoped that this section of the project would, when completed, predict the number of times a user would top up based on their social ties with other subscribers. Predicting a numeric value would require a regression algorithm to be used because the number of times of a subscriber tops up their credit balance is a numeric value. Although this problem could be converted to a classification problem by applying a decision function when calculating percentile scores to place the user in a wealth bracket.

The first algorithm to be tested was Linear Regression [37]. Linear regression, in this case, attempts to model the data using a multivariate linear equation

$$y = a_0 + a_1x_1 + \dots + a_nx_n + b$$

Where x is each attribute value and a is the weight associated with each attribute. The weights are calculated using the least squares fit method.

The mean square error is calculated for Linear Regression and it returned an error of $8.83e^{-24}$ on one of the occasions it was run. Ideally a mean squared error should be zero and the error is usually dependent on the values being used, e.g. calculating the best path to Australia dependent on wind patterns will be more acceptable to return a higher error than predicting the width of hair based on age. So even though the majority of users only use the credit top up code a small number of times, it can be said that the mean square error is so small that Linear Regression performed well on the dataset.

Scikit-Learn also allows for the correlation between variables to be calculated. This was calculated for the amount of credit top ups. The results in Figure 29 show that the number of times that a ‘Call Me Back’ request is initiated is the biggest factor in calculating credit top ups, outside of the actual number of credit top ups themselves.

```
C:\Users\Conor\Documents\GitHub\FYP>python SVM.py
mod_class      -0.017003
in_degree      -0.060046
out_degree      0.302346
total_degree    0.189715
CredCount       1.000000
Name: CredCount, dtype: float64
The accuracy is 94.619483844%
```

Figure 29: Correlation and SVM accuracy

Linear Support Vector Machines was also used in this project. This is similar to Linear Regression but it tries to find the optimal hyperplane which reduces the cost and tries to increase the accuracy of the algorithm. The results when SVMs were used was very encouraging with an accuracy of 94.6% accuracy recorded. In SciKit-Learn, arguments can be provided to Linear SVM to try to improve the accuracy. When these parameters were altered, the accuracy was increased to >99%. However, it can be said that the algorithm is now over fitting the dataset and developers must be wary to this. This problem can be avoided through the use of cross validation. With such an accurate algorithm, the wealth of a subscriber can therefore be classified using their social ties.

5. CONCLUSIONS AND FUTURE WORK

Even though this project met the requirements set out in the project specifications, there is always further work which can be completed.

Further exploration into the timing of USSD code usage would be an area of interest. By studying the timing of when a user invokes a code, it would be interesting to look at the possibility of trying to predict when a user will interact with a mobile network and see if patterns in a subscriber's life can be discovered or predicted.

Future work includes a balance check of users and the possibility of creating a credit rating system based on a user's balance. There are services in the dataset that contain balance checks which return the data in the response section of a USSD transaction instead of via SMS. This would allow for balances to be queried and this could be correlated with the timing of top ups in an attempt to see what value subscribers let their balance drop to before they top up again.

Another possible aspect of future work would be to perform cross-validation on the machine learning algorithms that were tested and to further test new algorithms such as a neural network. This could be done to find the optimal algorithm for this data and ensure the algorithms are not over fitting the dataset.

In conclusion, the aim of this project was to develop a framework to house and analyse unique Call Detail Records. This required the examination of the raw data, the exploration of frameworks to store this data and investigations on different forms of analysis on the dataset.

The project that has been outlined in this report focuses on understanding the different attributes of the raw data, how they can be used and why some of them can be discarded. It studies different forms of databases and researches the advantages and disadvantages of different frameworks. This project looks at how the data can be transferred from text files to the selected database, the techniques used to avoid USSD transactions in error and methods to reduce execution time and deal with big data. The addition of a real-time aspect to the framework was included as another feature to this project to provide the most up to date data for any future researcher implementing the framework. Visualisation and analysis of the

network was completed and the statistics retrieved from this were later used in predictive models. This visualisation allowed the social relationships between mobile subscribers to be examined. The examination of social ties lead to research being conducted into areas such as the Power Law and how this could fit to 'Call Me Back' and credit distributions. The trends and patterns in the usage of some USSD codes were discussed and interesting anomalies were uncovered, such as the dip in credit top ups during the month of Presidential elections. Using other open source tools, methods such as PCA and machine learning algorithms were employed to complete further research on the CDR dataset. The utilisation of Scikit-Learn, a Python Machine Learning package, allowed for the number of Credit Top Ups of a subscriber to be predicted using a user's social interactions within the network. The high accuracy displayed through the implementation of Linear Regression and Linear SVM were very promising. Finally, users were placed into different brackets of wealth based on a percentile score they received. This was calculated from the number of times they topped up their credit. By combining the predictive models developed and the division of a network into different wealth brackets, it can be said that a user's income bracket can be predicted using their social interactions within a telecommunications network.

REFERENCES

- [1] 48, (2017). *Example USSD Codes*. [image] Available at:
<http://community.48months.ie/t5/Most-Popular-Questions/Handy-short-codes/td-p/986>
 [Accessed 6 Apr. 2017].
- [2] Mocean, (n.d.). *USSD Gateway*. [image] Available at:
<http://www.mocean.com.my/unstructured-supplementary-service-data-solution.php>
 [Accessed 6 Apr. 2017].
- [3] Tangotelecom.com. (n.d.). *Tango Telecom / Data Monetisation Solutions / Tango DRE / Making Data Pay*. [online] Available at: <http://www.tangotelecom.com/home.html>
 [Accessed 7 Apr. 2017].
- [4] *Book Of Abstracts Oral*. [online] Available at:
http://www.netmob.org/www15/assets/img/netmob15_book_of_abstracts_oral.pdf
 [Accessed 25 Mar. 2017].
- [5] Fast Company. (2017). *This New Kind Of Credit Score Is All Based On How You Use Your Cell Phone / Fast Company*. [online] Available at:
https://www.fastcompany.com/3058725/this-new-kind-of-credit-score-is-all-based-on-how-you-use-your-cellphone?show_rev_content [Accessed 25 Mar. 2017].
- [6] Blondel, V., Decuyper, A. and Krings, G. (2015). A survey of results on mobile phone datasets analysis. *EPJ Data Science*, 4(1).
- [7] GSMA, (2017). *The Mobile Economy 2017*. [online] GSMA, p.6. Available at:
<http://www.gsma.com/mobileeconomy/#> [Accessed 25 Mar. 2017].
- [8] Du Toit, J. (2017). *Will There Still be a Need for CDR Analysis in the Future?*. [online] Iris Network Systems. Available at: <https://www.irisns.com/will-still-need-cdr-analysis-future/> [Accessed 25 Mar. 2017].
- [9] www.tutorialspoint.com. (2017). *MySQL Introduction*. [online] Available at:
<https://www.tutorialspoint.com/mysql/mysql-introduction.htm> [Accessed 25 Mar. 2017].

- [10] MySQL Table Example. (n.d.). [image] Available at: <https://dev.mysql.com/doc/workbench/en/wb-getting-started-tutorial-adding-data.html> [Accessed 5 Apr. 2017].
- [11] Hand, D., Mannila, H. and Smyth, P. (2012). *Principles of data mining*. 1st ed. New Delhi: PHI Learning Private Limited, pp.400, 420.
- [12] www.tutorialspoint.com. (2017). *What is NoSQL and is it the next big trend in databases?*. [online] Available at: <http://www.tutorialspoint.com/articles/what-is-nosql-and-is-it-the-next-big-trend-in-databases> [Accessed 26 Mar. 2017].
- [13] Docs.mongodb.com. (n.d.). *Map-Reduce — MongoDB Manual 3.4*. [online] Available at: <https://docs.mongodb.com/manual/core/map-reduce/> [Accessed 8 Apr. 2017].
- [14] Neo4J, U. (2017). *Undirected Relationship in Neo4J*. [online] Stackoverflow.com. Available at: <http://stackoverflow.com/questions/33368073/undirected-relationship-in-neo4j> [Accessed 26 Mar. 2017].
- [15] Neo4j Graph Database. (n.d.). *Why Graph Databases? - Neo4j Graph Database*. [online] Available at: <https://neo4j.com/why-graph-databases/> [Accessed 8 Apr. 2017].
- [16] Neo4j, (2017). *Graph Database Example*. [image] Available at: <https://neo4j.com/developer/guide-neo4j-browser/> [Accessed 26 Mar. 2017].
- [17] Ohlhorst, b. (2013). *NoSQL vs. SQL: What's best for the different data types of BI?*. [online] Available at: <http://searchoracle.techtarget.com/news/2240177654/NoSQL-vs-SQL-Whats-best-for-the-different-data-types-of-BI> [Accessed 30 Sep. 2016].
- [18] Ngumii, P. (2015) *SQL vs NoSQL: The differences*. Available at: <https://www.sitepoint.com/sql-vs-nosql-differences/> (Accessed: 21 December 2016).
- [19] *ArangoDB vs. MongoDB vs. OrientDB comparison* (2016) Available at: <http://db-engines.com/en/system/ArangoDB%3BMongoDB%3BOrientDB> (Accessed: 23 December 2016).

- [20] Leaks, O. and profile, V. my complete (2015) *Why you should avoid OrientDB*. Available at: <http://orientdb leaks.blogspot.com/2015/06/the-orientdb-issues-that-made-us-give-up.html> (Accessed: 1 October 2016).
- [21] Docs.python.org. (n.d.). *14.1. hashlib — Secure hashes and message digests — Python 2.7.13 documentation*. [online] Available at: <https://docs.python.org/2/library/hashlib.html> [Accessed 8 Apr. 2017].
- [22] Pypi.python.org. (2017). *watchdog 0.8.3 : Python Package Index*. [online] Available at: <https://pypi.python.org/pypi/watchdog> [Accessed 30 Mar. 2017].
- [23] www.tutorialspoint.com. (2017). *Operating System - Multi-Threading*. [online] Available at: https://www.tutorialspoint.com/operating_system/os_multi_threading.htm [Accessed 1 Apr. 2017].
- [24] Lu, Z. and Li, X. (2016). Attack Vulnerability of Network Controllability. *PLOS ONE*, 11(9).
- [25] Necsi.edu. (2017). *Concepts: Power Law / NECSI*. [online] Available at: <http://www.necsi.edu/guide/concepts/powerlaw.html> [Accessed 3 Apr. 2017].
- [26] Adamic, L. and Huberman, B. (2017). *Power-Law Distribution of the World Wide Web*. [online] Science.sciencemag.org. Available at: <http://science.sciencemag.org/content/287/5461/2115> [Accessed 3 Apr. 2017].
- [27] Barabási, A. and Bonabeau, E. (2003). Scale-Free Networks. *Scientific American*, [online] 288(5), pp.60-69. Available at: [http://www3.nd.edu/~networks/Publication%20Categories/01%20Review%20Articles/ScaleFree_Scientific%20Ameri%20288,%2060-69%20\(2003\).pdf](http://www3.nd.edu/~networks/Publication%20Categories/01%20Review%20Articles/ScaleFree_Scientific%20Ameri%20288,%2060-69%20(2003).pdf).
- [28] R. Kiremire, A. (2011). 1st ed. [ebook] pp.1-2. Available at: http://www2.latech.edu/~box/ase/papers2011/Ankunda_term paper.PDF [Accessed 3 Apr. 2017].
- [29] Rosie Dunford, Quanrong Su, Ekraj Tamang and Abigail Wintour, The Pareto Principle, The Plymouth Student Scientist 7, (1), 140-148 (2014).

- [30] Seshadri, M., Bolot, J., Machiraju, S., Faloutsos, C., Sridharan, A. and Leskovec, J. (n.d.). *Mobile Call Graphs: Beyond Power-Law and Lognormal Distributions*. [online] Available at: <http://www.cs.cmu.edu/~jure/pubs/dpln-kdd08.pdf> [Accessed 4 Apr. 2017].
- [31] Explained Visually. (n.d.). *Principal Component Analysis explained visually*. [online] Available at: <http://setosa.io/ev/principal-component-analysis/> [Accessed 8 Apr. 2017].
- [32] Weka PCA tutorial. (2011). [video] Available at: https://www.youtube.com/watch?v=PY20auk5F_E [Accessed 8 Apr. 2017].
- [33] The Citizen, (2016). Internet remains cut in Chad after tense elections. [online] Available at: <http://www.thecitizen.co.tz/News/Internet-remains-cut-in-Chad-after-tense-elections/-/1840340/3156880/-/1331m1f/-/index.html> [Accessed 1 Apr. 2017].
- [34] Cia.gov. (2017). *The World Factbook — Central Intelligence Agency*. [online] Available at: <https://www.cia.gov/library/publications/the-world-factbook/fields/2046.html> [Accessed 1 Apr. 2017].
- [35] Nationsencyclopedia.com. (2017). *Income - Chad - product, average, growth, annual, power*. [online] Available at: <http://www.nationsencyclopedia.com/Africa/Chad-INCOME.html> [Accessed 1 Apr. 2017].
- [36] 2016, (2010). *Installing scikit-learn — scikit-learn 0.18.1 documentation*. [online] Available at: <http://scikit-learn.org/stable/install.html> [Accessed 23 Dec. 2016].
- [37] Stat.yale.edu. (2017). *Linear Regression*. [online] Available at: <http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm> [Accessed 1 Apr. 2017].