NTNU
Norwegian University of
Science and Technology

Algorithms and Methods for Organised Cybercrime Analysis

Jan William Johnsen

01-06-2016

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Norwegian University of Science and Technology, 2016

Supervisor 1: Prof. Katrin Franke
Supervisor 2: Prof. Slobodan Petrović

Documented Changes to Master Thesis

Jan William Johnsen

22-08-2016

List of Changes

These changes was made after the final delivery and before publication.

- Changed the global hyperlink colour from blue to black.

Preface

This thesis concludes my Master of Science education in Information Security, digital forensics track, at The Norwegian University of Science and Technology (NTNU) in Gjøvik. It was performed throughout my 10th semester, spring 2016, at the faculty of Computer Science and Media Technology. My supervisor on the project has been Katrin Franke and Slobodan Petrović, both is professors at NTNU in Gjøvik.

The topic idea of this thesis was one of the suggested project proposals during a collected meeting with faculty staff in late 2015. It has later been defined through a project planning course, which had the delivery date in Dec 2015.

The author has a bachelor degree in Information Security from Høgskolen i Gjøvik, before they merged with NTNU at the beginning of 2016. We give comprehensive information for the background theory so the reader can follow the later chapters of this thesis. However, it would benefit the reader to be already familiar with or have an interest in graph theory.

I hope you will enjoy reading this thesis.

Gjøvik, 01-06-2016

Jan William Johnsen

Jan William Johnsen

Acknowledgment

I would like to thank my supervisor, Professor Katrin Franke, for her excellent guidance and assistance throughout the project. During the thesis, she has shown a keen interest in my work and always been able to provide insightful discussions, which has guided this thesis. Furthermore, I would also thank my co-supervisor, Professor Slobodan Petrović, who helped me to improve this thesis with his extensive knowledge in the field of graph theory.

I would also thank my classmates David and Martin for the company during the entire process. Furthermore, I will thank my classmate Espen for being a skilled table tennis opponent and for discussion, feedback, and company during the whole process. I would also like to thank my current classmate Lars Christian, and my former classmate Lars for valuable discussions, feedback and proof reading of my work.

Finally, I would like to thank my family for motivation and support throughout my studies.

J.W.J.

Abstract

Law enforcement agencies reports that Organised Criminal Groups are moving more of their activities from traditional crime into the cyber domain. Where they form so-called Darknets, whose purpose is to act as marketplaces for illegal material, products, and services. These activities form a part of the Crime-as-a-Service business model, which drives the digital underground economy by providing services that facilitate almost any type of cybercrime. The challenge for law enforcement is knowing which entity to target for effectively taking down these network structures. This thesis seeks to use graph-based algorithms and methods to analyse network structures to identify interesting and relevant individuals within such networks. More specifically, it proposes Social Network Analysis (SNA) methods for the process of identification of such individuals.

The thesis analyse each SNA method to identify those methods that identify the most central individuals within a network. Also, it will analyse how the usage of different graph construction techniques can be applied to the process of identification. The thesis contributions is to try to bridge the gap between law enforcement agencies and cybercrimials by proposing an improved way of prioritising individuals within these networks.

Abbreviations

k**-NN** k-Nearest Neighbour.

ANN Artificial Neural Network.

API Application Programming Interface.

CaaS Crime-as-a-Service.

CoC Chain of Custody.

CSV Comma-Separated Values.

DBMS Database Management System.

DDoS Distributed Denial of Service.

GEXF Graph Exchange XML Format.

ICT Information and Communication Technologies.

IDS Intrusion Detection System.

MDL Minimum Description Length.

ML Machine Learning.

N/A Not Available.

OCG Organised Criminal Group.

OOV Order of Volatility.

PCA Principal Component Analysis.

SNA Social Network Analysis.

SOM Self Organising Map.

SVM Support Vector Machine.

XML Extensible Markup Language.

Glossary

Analogy a comparison between things that have similar features.

Attribute to regard as resulting from a specified cause; consider as caused by something indicated.

Commerce an interchange of goods or commodities, especially on a large scale between different countries or between different parts of the same country.

Concert together or in some co-ordinated manner.

Corpus a collection of written or spoken material in machine-readable form, assembled for the purpose of lingustic research.

Digital evidence digital information and data that contain reliable information that supports or refutes a hypothesis about an incident, where the data is stored on, received or transmitted by an electronic device.

Digital forensic is a structured process of investigating by collecting, identifying and validating digital information for the purpose of reconstructing past events. The goal is to preserve any Digital evidence in its most original form.

Facilitate to make something possible or easier.

Induce to bring about, produce or cause.

Infer to derive by reasoning; conclude or judge from premises or evidence.

Information security safe-guarding data from unauthorized access or modification to ensure its availability, confidentiality, and integrity.

Pendent rely on or being dependent of some other thing.

Recapitulation to give a brief summary of something.

Reciprocate to do something for or to someone who has done something similar for or to you.

Contents

List of Figures

List of Tables

Listings

1 Introduction

1.1 Topics covered by the project

Anonymisation techniques enable users to communicate freely without the risk of being traced, which allow them to host and connect to secret services and forums in parts of the Internet, known as Darknets. These underground forums are being used by Organised Criminal Group (OCG) as marketplaces for illegal material, buying and selling of products and services, and share experience and expertise [1]. These activities, combined with the Crime-as-a-Service (CaaS) business model, drives the digital underground economy by providing services that facilitate almost any type of cyber crime [1].

An extensive network of persons that fulfill specific functions build the CaaS business model [2]. For example, a hacker discovers a vulnerability in a software program, which can be sold to another who writes an exploit that uses this vulnerability to take control of vulnerable machines. When the hacker has control of these devices, they can be sold to another hacker who might group them with other compromised machines to form a botnet of remotely controlled computers. The botnet becomes a platform from which cyber criminals can hire, to launch attacks against websites or networks.

Digital forensic is a process of investigating past events, by collecting, identifying and validating digital information. However, it is essential to gather intelligence about these networks to increase the success of a digital forensic investigation. What is both important and challenging is to filter out uninteresting information, leaving entities of interest for the investigation. With potentially many thousands of online criminals in one underground forum, efficient algorithms and techniques must be used to filter all of the criminal entities.

1.2 Keywords

Social Network Analysis, network and graph analysis, graph theory, data mining and digital forensic.

1.3 Problem description

Many OCGs are moving their activities from traditional crime into cyber crime [1, 3], where the challenge lies in understanding how they organise online. The population in these underground forums is separated into two distinct groups. The larger demographic consists of inexperienced persons with low or little technical skill that carry out minor offenses. Whereas the smaller demographic consists of highly skilled individuals, that not only conduct the most damaging cyber attacks but they also sell their services to the larger population [1].

The anonymity technology cyber criminals use to mask their activity online leaves law enforcement agencies insufficiently prepared for new threats from cyber crime. The conservative and reactive nature of policing create the risk of developing a blind spot for those crimes.

Law enforcement agencies are aware of the growing problem of online OCGs, and they recommend to target individuals or groups with high reputation on these underground forums. Alternatively, to target their key support services when it is not possible to target them due to lack of attribution or jurisdictional issues [1]. The challenge lies in knowing the relationships between cyber criminals, knowing how they organise and where to focus law enforcement investigation and intervention.

Cyber criminals create electronic trails as part of their online activities and interactions. These trails are contained within enormous data sets that are difficult to filter, analyse and interpret using traditional data processing methods. They are often not immediately visible but may be hidden in the data in the form of relationships and correlations [1].

1.4 Justification, motivation and benefits

Law enforcements recommends a focus on dismantling cybercriminal infrastructure [1, 3], to disrupt the critical services that support or enable cybercrime. The arrest and prosecution of targets within the small (and highly skilled) population brings a few advantages. This cyber criminal "elite" consists of fewer targets, but they contribute more to the CaaS with new ideas and innovation of techniques and services. By removing targets within this group, one will cause more disruption than by removing targets from the large population, as the numbers of highly skilled cybercriminals are limited. Also, their technical skills are hard to replace by a group with lower skills.

Although targeting the cyber criminal "elite" may be particularly challenging due to jurisdiction. Many secondary targets provide skilled and essential services whose removal would cause considerable disruption to the market [1]. When cybercrime tools and services become more available and easy to access, the less technical group will grow, which in turn leads to the development of more highly skilled individuals. Therefore, it is important to develop useful algorithms and methods for targeting online cyber criminals.

1.5 Research questions

This thesis is to a large extent motivated by the use of graph analysis for identification of influential individuals and the behaviours found within a group of people. In particular, this thesis seeks to answer the following questions:

1. *Which features can be used to identify important and influential individuals within a network?*
2. *How can graph construction techniques be applied in digital forensics for identifying targets?*

Our research questions are of an explorative nature and aim to gain an understanding of relations between individuals. The answer to the first research question will provide the knowledge that can help understand and explain one particular organisation or group. The answer to the second research question will provide the knowledge of how graph construction algorithms can be of help for law enforcement agencies in an investigation.

1.6 Contributions

A communicating network is a dynamic structure and has both large and small changes over time. This master thesis seeks to provide a better understanding of the abilities to find communication relationships that can identify high-value targets in a network. Particularly with a focus on graph-based methods, where this thesis should provide methods and techniques that law enforcements could utilise for investigating relationships between cyber criminals.

The organised cyber crime advocates a more creative approach to combatting traditional law enforcement investigations, prosecutions, and surveillance methods. It has to the author's knowledge not been discussed previously, and it is important to analyse this problem as these criminal organisations continue, even if leadership or membership changes over time.

The intended goal of this thesis is to evaluate how the use graph algorithms can solve the problem of finding high-value targets, to provide law enforcement agencies a more comprehensive list of targets.

1.7 Thesis outline

This section provides a brief summary listing of the contents presented in this thesis. The listing is based on the chapters, where each chapter and its content is described. First, the necessary background theory and related work are presented. Then the methodologies for this thesis is presented, followed by the experiment design, setup, and execution. Finally, a recapitulation describes the experiment results with a discussion and conclusion, as well as further work.

- Chapter 2 provides an overview of the field of digital forensic and Machine Learning. It also describes the most fundamental knowledge of graph theory for the section about Social Network Analysis.
- Chapter 3 presents the state of the art from published literature regarding organised crime. Further, we discuss the most common methods for analysing social networks.
- Chapter 4 provides an overview of the methodology used to guide this thesis. It includes a description of the dataset, feature quality evaluation, and a discussion of the different methods used to achieve our results.
- Chapter 5 presents the experiments performed, technical specification of the equipment, software requirements necessary to perform the experiments. Further, it discusses the results from feature evaluation and graph construction, as well as expected results.
- Chapter 6 provides a recapitulation of the thesis and the most significant findings, including theoretical implications, practical considerations, and proposals for future work.

2 Background

The previous chapter gave an introduction to the thesis and introduced the research questions that will guide this thesis. It also described the justification and motivation for thesis and its contributions. The following chapter will present relevant background material. It begins by describing different types of OCGs and strategies to combat the threat from these groups. Then it describes the field of research known as Machine Learning (ML), as some of the ideas within this area is used in this thesis. Then we give an introduction to graph theory. It serves as an introduction to the section about Social Network Analysis (SNA), as this field employ methods from graph theory to understand social networks and structures. At the end of this chapter, we provide a recapitulation to summarise the most relevant background theory provided in this chapter.

2.1 Organised crime

Before developing successful strategies and techniques to combat organised criminal activity, it is important to define organised crime and how it functions [4]. The term "group" separates individual crime from crimes committed by groups, and it is usually used to describe a group of people who cooperate to accomplish a goal. The definition of OCG from Article 2 of the *United Nations Convention against Transnational Organized Crime* [5] is adopted in this thesis:

> A structured group of three or more persons, existing for a period and acting in concert with the aim of committing one or more serious crimes or offences to obtain, directly or indirectly, a financial or other material benefits.

A "structured group" is referred to as a group that have existed for some period before or after the offence(s) [5]. It is not necessary to have formally defined roles for its members, continuity of its membership or a developed organisational structure. However, many OCGs incorporates many of the successful structures from legitimate business organisations [4]. The structures includes common rules, a hierarchy of authority, division of labour and responsibility, individual specialisation and specialised training. Their activities are often local to one country, but they can also be transnational organisations. Transnational crimes are crimes that have an effect across national borders, and that offend fundamental values of the international community [6]. The activities of these groups have grave implications on (inter)national security (e.g. political, economic and social areas). As their operations include harassment, fraud, unlawful propaganda, pornography and prostitution, theft, money laundering, espionage, drug and human trafficking, identity theft and financial scams [7, 8].

OCGs have many different structures and types, just like any legitimate business organisation. Therefore, our focus will be on describing the types that utilise Information and Communication Technologies (ICT) to commit offence(s) in the cyberspace domain. We differentiate between "cyber-enabled" OCGs that move some of their operations online and the "cyber-dependent" OCGs that operate exclusively online.

2.1.1 Organised crime groups typology

The modern ICT provides over tree billion people with access to the Internet, which had a global growth of 6.6% in 2014 [9]. This trend with global expansion will continue as telecommunication companies expand their broadband services, and more people acquire personal devices such as computers, smartphones and tablets.

Commercial businesses use ICT's bandwidth speed and availability to provide services for their customers all over the world, e.g. online banking and e-commerce. Even national and local government organisations recognise the reduced cost by providing electronic services to their citizens. Internet services such as banking, shopping and e-commerce are now linked to almost every part of our daily lives, and we are becoming increasingly globalised and interconnected.

This technology creates an asymmetry between criminals and law enforcement. Where one criminal has access to countless victims from anywhere in the world, and law enforcement struggle to determine the scope of the criminal activity. Criminals that are using this technology use it to conduct new crimes or to commit traditional crimes in new ways. Kim-Kwang R. Choo [8] identified three types of OCGs that exploit advances in ICT: *traditional organised crime groups*, *ideologically or politically motivated organised groups* and *organised cybercriminal groups*. The two first types are cyber-enabled, and the last is cyber-dependent.

Traditional organised crime groups are mainly involved in profit generating activities such as extortion of businesses, monopolising, bouncership in nightclubs, prostitution and human trafficking, narcotics and weapon trafficking, illegal bookmarking, unlicensed money lending and collection of protection money [8]. Money has always been the driving force behind traditional OCGs, but they have adopted ICT to facilitate or enhance their profit-generating criminal activities [8].

Ideologically and politically motivated organised crime groups are mainly driven by ideology or political views. These types of groups and organised crime have been considered separate becuase they did not share the same motivating factors for their activities [8] But in recent years, there has been a convergence between them. Warren [10] and Charlton [11] have noted that terrorist groups such as Al-Qaeda have recruited experts in OCGs for their money laundering expertise, to help them overcome their money supply issues.

Organised cybercrime groups are involved in profit generating activities such as hacking, Distributed Denial of Service (DDoS) attack, malicious software, piracy, online fraud and identity theft. Since these groups operate exclusively online, it is likely that they operate with different organisational structures. For exampple, they co-operating for a limited period to conduct a specifically defined task or set of tasks [8]. Cybercrime groups are therefore more loosely structured, flexible and transnational than the other types of groups. Organised cybercrime groups tend to have smaller membership sizes [8], as their physical strength is irrelevant in the cyber domain. They rely more on knowledge over technology to bypass electronic defences. Therefore, their strength is in software and not in the number of individuals. Where the most technologically sophisticated individuals sell their knowledge and expertise to others via the CaaS business model.

2.1.2 Organised crime analysis

There is not much theory for the analysis of digital OCGs. It is either because law enforcements agencies do not want to leak information about how they operate, it is challenging to analyse those types of crimes, or this field of analysis is still being developed. However, they often involve methods and processes that we find in a digital forensic investigations. These processes are employed after a (cyber) crime has been commited, and reported to the authorities. In 2001, Digital Forensics Research Workshop defined digital forensic as [12]:

> The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.

The investigation process is done by using forensically sound and rigorous methods for handling digital evidence, to preserve it in its most original form. The goal for the forensic examination is to find facts, and via these facts to recreate the truth of an event [13]. Figure 1 shows the process sequence for gathering digital evidence, which consists of six phases:

Identification includes actions such as system monitoring and audit analysis to verify that an incident has occured.

Preservation involves setting up necessary equipment to gather the evidence to ensure an acceptable Chain of Custody (CoC) and Order of Volatility (OOV). This step is crucial to avoid contamination of proof.

Collection retrieves relevant data via approved methods of recovering techniques.

Examination and analysis consist of recovery of hidden or encrypted data, data mining, timelining, evidence validation and other analytical methods.

Presentation consists of documentation, expert vitness in court and safe storage of digital evidence.

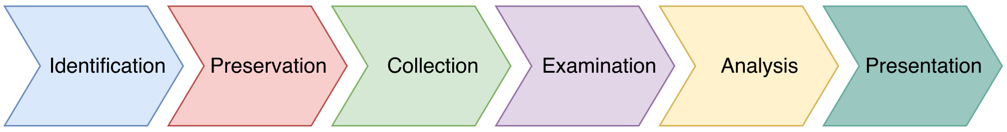

Figure 1: Forensic investigation process

In addition to the requirement that evidence must be legally obtained to be admissible in a court of law, there are also two essential forensic principles that investigators must follow: CoC and OOV. Investigators must be aware of the order of evidence collection given in the OOV principle. Digital data have different lifetimes before they either vanish, are subjected to change or modification. The OOV principle allows an investigator to think about the life of the data, to collect the most volatile data first. However, by collecting more volatile data such as memory, they will inadvertenly change less volatile data on the system, such as information in hard disks. OOV principle should be considered before starting an investigation; the same goes for CoC. It is “the chronological documentation of the movement, location and possession of evidence” [13]. This principle documents the integrity of evidence, to show that it has not been changed since acquisition. Therefore, it is important to document the evidence collection phase. Hash functions are often used to ensure the integrity of digital data.

2.2 Machine learning

Advances in ICT gives us the ability to store and process significant amounts of data, as well as to access it over a computer network [14]. Think of the example of a supermarket chain, with hundreds of stores all over a country, selling thousands of goods to millions of customers. The point of sale terminals generates gigabytes of data every day, by recording the details of each transaction between the customer and the store: date, goods bought, money spent and so forth. Investigation of this type results in the collection of vast amounts of data, where tiny pieces of evidence are hidden in chaotic environments [15]. ML is a field of study that can analyse this amount of data and turn it into information that we can use [14], for example, to make predictions.

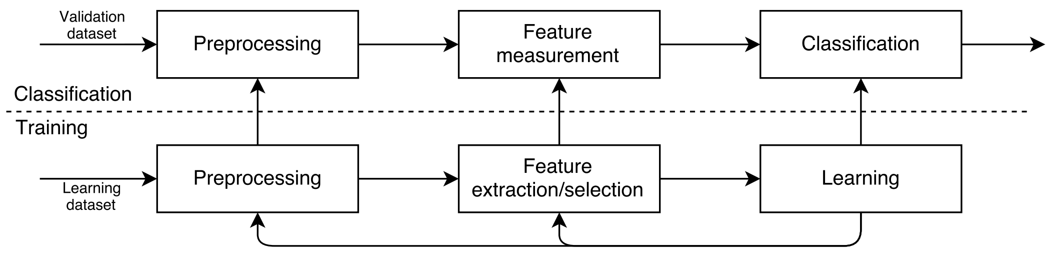

Figure 2: Machine learning processs [16]

2.2.1 Machine learning methods

We differentiate ML methods on how the obtained (induced) knowledge is used [17]: classification, regression, clustering, learning of associations, relations and differential equations. This thesis explains the first three ML methods, as indicated by the blue colour in Figure 3.

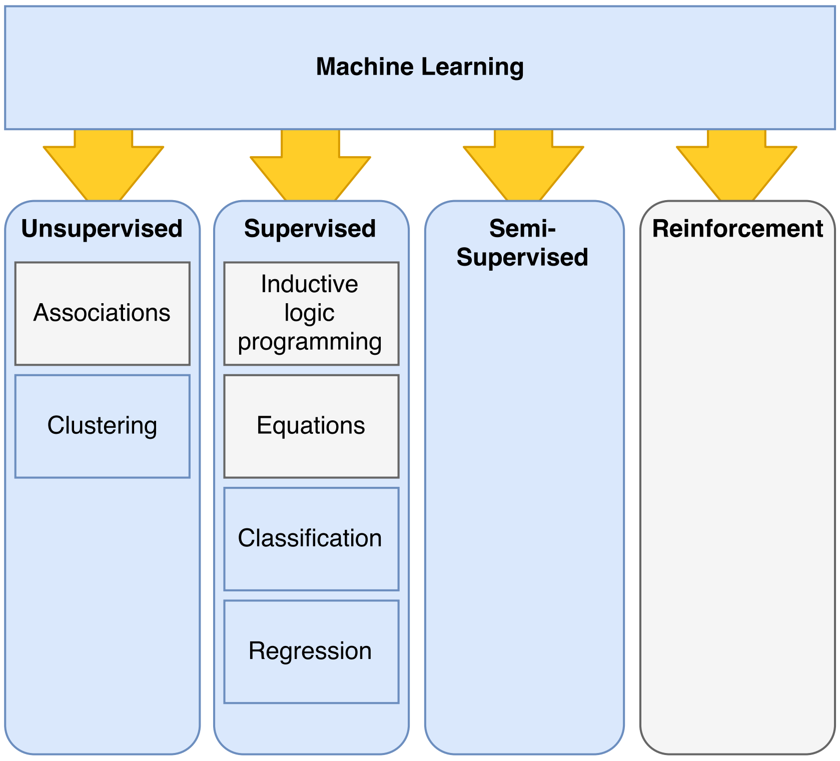

Figure 3: Machine learning taxonomy [17]

Supervised learning

Supervised learning is the task of inferring a function from a labelled data set [18]. The set L consists of n number of samples, and each sample is a pair composed of a feature and the desired output value (often called a *target variable*, *class* or *label*). The data set typically look like $L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x is a vector of length m and y is the desired label for the input sample (x, y). The data set L is typically divided into two complementary subsets.

The first subset L_t (called *training set*) is presented to the ML method to train the model, by pairing the input with the target variable. The second subset L_v (called *validation set* or *testing set*) is used to evaluate how well the model has been trained and to estimate model properties such as classification or regressional error. Methods for dividing the original data set into a training set and validation set is discussed in Section 2.2.4.

This section will cover the two supervised learning categories *classification* and *regression*. In classification, the class is categorical (described in Section 2.2.2), and the goal is to produce an inferred function from the learning set, that would determine the correct class for unseen samples in the validation set.

An example of a (linear two-class) classification problem can be seen in Figure 4. The classification algorithm learns to infer a function $f(x) \rightarrow y$ (where y is categorical) that can separate the square and circle classes from each other, using data samples in the training set. Where the red line illustrates the inferred function. This function is evaluated by testing it on the validation set, counting the number of correctly and incorrectly classified data samples to get the accuracy for the function. Some common classification algorithms are decision trees, naive-Bayes, Bayes network, decision rules, k-Nearest Neighbour (k-NN), linear discriminant functions, logistic regression, Support Vector Machine (SVM) and Artificial Neural Network (ANN) [17, 18].

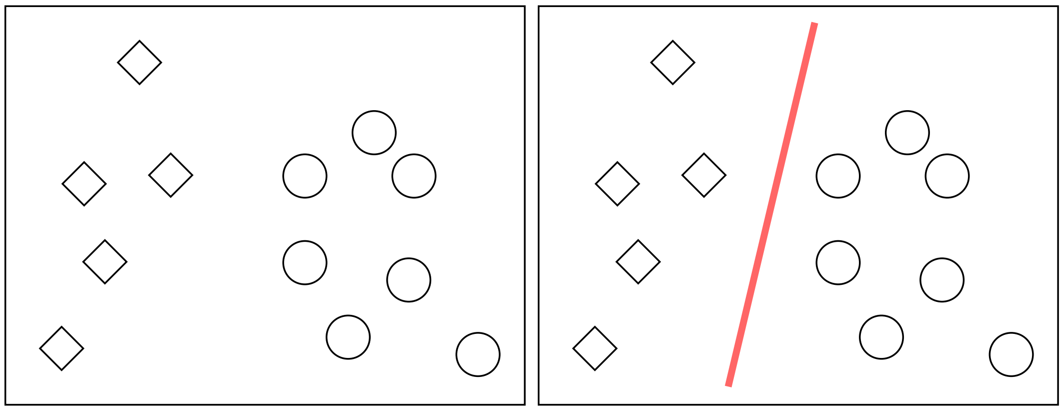
Figure 4: Supervised learning: Classification

For regression, the target variable is numerical/real-valued (described in Section 2.2.2), and it is not reasonable for the regression algorithm to predict precisely the target variable. Instead, the requirement is the prediction to be close to the correct ones. The key difference between classification and regression: in regression, the measure of error is based on the magnitude of the difference between the real-valued prediction and the correct one, and in classification it relies on the equality or inequality of these two labels [18].

An example of a regression problem can be seen in Figure 5. The regression algorithm learns to infer a function $f(x) \rightarrow y$ (where y is continuous) that minimise the error of the predicted value. The red line illustrates the inferred function with the least error. Some common regressional algorithms are regression trees, linear regression, locally weighted regression, SVM (for regression) and ANN [17, 18].

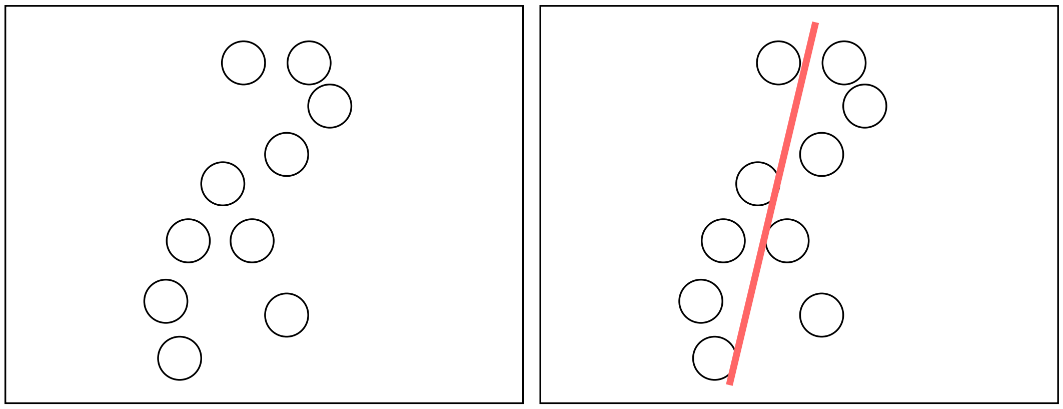

Figure 5: Supervised learning: Regression

Unsupervised learning

Unsupervised learning is the task of inferring a function to describe hidden structures from an unlabeled data set [17]. The set U consists of n number of data samples, and each sample contains only an input object. The data set typically looks like $U = \{x_1, x_2, \ldots, x_n\}$, where x is a vector of length m. The distinction from supervised learning is that data samples do not have a label (the desired target variable), so there is no error or reward to evaluate a potential solution generated by the unsupervised learning algorithm.

The goal will, therefore, be to organise the data samples into groups with similar characteristics. This process of organisation is called *clustering*, which is by far the most popular unsupervised learning method [17]. The task of clustering is to group a set of similar samples into groups (called *clusters*) where the groups are more analogous to each other than those in other groups.

Distance measures are often interchangeably described as *similarity measures* in the literature, although there are differences between distances and similarities, they are both referred to as distances in this thesis. A small distance is equivalent to high similarity, and a large distance is equivalent to low similarity. Unsupervised learning algorithms use distance measures such as *Euclidean distance* (2.1), *Manhattan distance* (2.2) and *Minkowski distance* (2.3) (for $c = 1$ and $c = 2$ the Minkowski metric becomes equivalent to Manhattan and Euclidean distance respectively). Where p and q is 1-dimensional feature vectors.

$$d(p, q) = \sum_{i=1}^{m} \sqrt{(p_i - q_i)^2} \tag{2.1}$$

$$d(p, q) = \sum_{i=1}^{m} |p_i - q_i| \tag{2.2}$$

$$d(p, q) = (\sum_{i=1}^{m} |p_i - q_i|^c)^{1/c} \tag{2.3}$$

There are two types of clustering: *hierarchical clustering* and *partitional clustering*. Partitional clustering partition the data samples into a specified number of clusters and then evaluate these clusters by either minimising or maximising some numerical criterion (such as distance measure) [17].

Figure 6 is a typical clustering problem where each data sample is associated with a vector (two dimensions in this example). Partitional clustering algorithms require the user to specify the number of clusters (two clusters in this example), and the algorithm will initialize the *centroids* (mean vector representing all samples in that cluster). These centroids will continue to move to the centre of the cluster as long as data samples change clusters, updating the centroid mean each time. The clusters have *converged* when none of the data samples has changed clusters, as illustrated by the blue and green clusters in the figure.

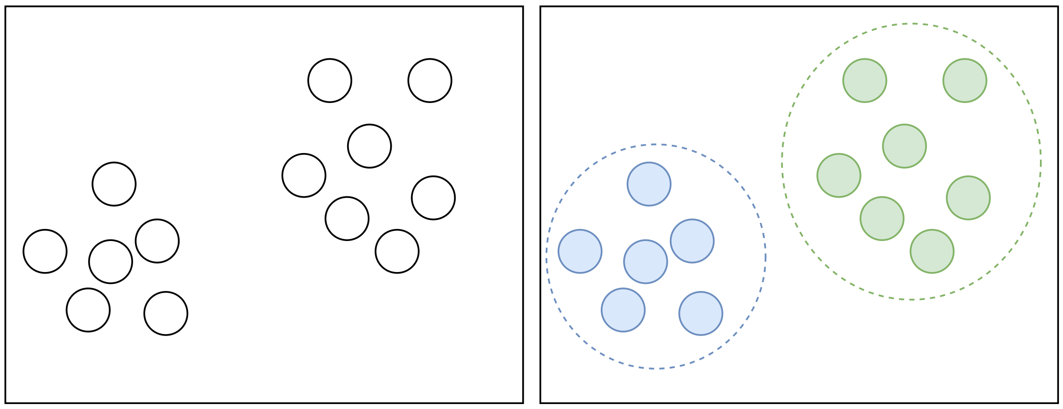

Figure 6: Unsupervised learning: Partitional clustering

Hierarchical clustering can be further divided into two categories: bottom-up (agglomerative) or top-down (divisive) [17]. Bottom-up hierarchical clustering begins with each example as separate clusters and merges them in successively larger clusters, whereas top-down hierarchical clustering begins with the whole set of examples and proceeds to divide it into successively smaller clusters [17]. Both categories for hierarchical clustering seeks to build a hierarchy of clusters, usually displayed as a dendrogram (see Figure 7).

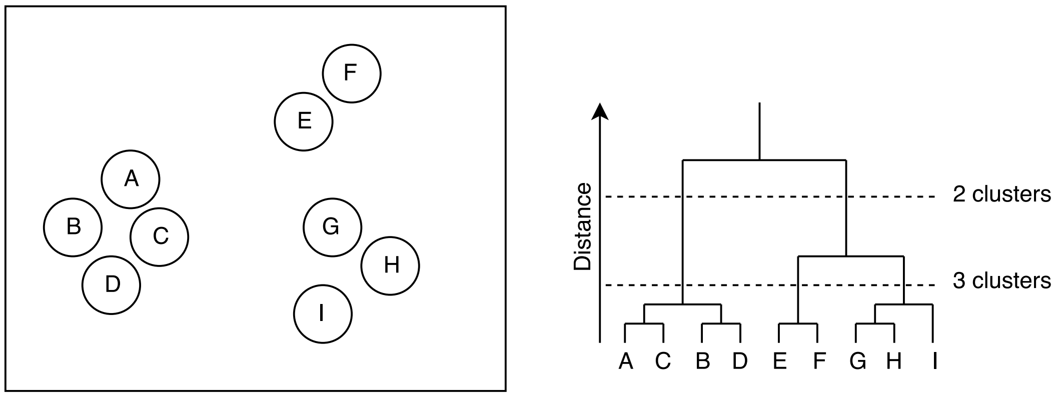

Figure 7: Unsupervised learning: Hierarchical clustering

A good clustering result will have both *high intra-cluster distances* and *low inter-cluster distances* [17]. The first is the sum of distances between objects in different clusters are maximised while the latter is the sum of distances between objects in the same cluster are minimised. Some common clustering algorithms are k-means, affinity propagation, mean shift, spectral clustering, hierarchical clustering and competitive learning (Self Organising Map (SOM)).

Semi-supervised learning

Semi-supervised learning is a type of supervised learning that in addition to labelled data also uses unlabelled data in the task of inferring a function. The motivation for using unlabelled data is that labelled data is expensive to generate. It is time-consuming and requires expert knowledge of the data, whereas unlabeled data is not difficult to obtain. ML researchers have found that using a small amount of labelled data together with a larger amount of unlabeled data can produce considerable improvement in learning accuracy [17, 18].

Semi-supervised learning is a type of supervised learning that makes use of one small labelled data set L together with one unlabeled data set U for training. Set $L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and $U = \{x_1, x_2, \ldots, x_n\}$, where x is a vector for length m and y is the desired label for the corresponding input object.

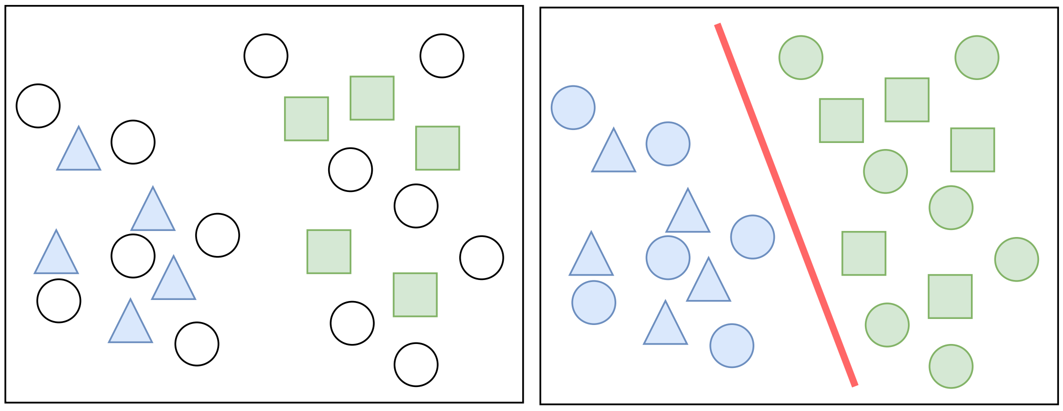

Figure 8: Semi-supervised learning

2.2.2 Features and feature quality

Features are often referred to as *attributes*, *variables* or *properties* in the literature [17, 18]. However, this thesis makes a distinction between attribute and feature. Features have so far been talked about being some abstract thing, but they are one of the most fundamental descriptions we have of objects in the real world that can be interpreted by a machine.

Objects in the real world can be described by several *attributes* such as colour, length, weight and so forth. A *feature* is a specification of an attribute and its value [19], i.e. "its weight is 25 kg" is a specification. A series of features can be conveniently described by a *feature vector*, which is an n-dimensional vector of features (usually numerical) that represents some object. *Dimension* usually refer to the number of features.

Features can take on several roles, such as class as described in Section 2.2.1 supervised learning. Supervised learning problems have a data set S that contains several samples $s \in S$, where $s = \{x, y\}$ (feature vector x and class y). The data samples usually only belong to one class, but there can exist several different classes (which will not be covered in this thesis). The feature vector is a n-dimensional vector of features $x = \{x_1, x_2, \ldots, x_n\}$. Each feature x usually take on a numerical value, either discrete or continuous.

Different learning algorithms require various types of features. Conversion between feature types is typically done during the preprocessing phase, and it is further discussed in Section 2.2.3. This is a list of four different feature types and examples of the values they can take:

- *Categorical* is a finite number of discrete values, and there are two kinds to denote the ordering [19]. The type *nominal* denotes that there is no ordering between the values, such as colour. The type *ordinal* denotes that there is ordering, such as low, medium or high.

- *Numerical* is a finite number of discrete or continuous values. Discrete values are counted, so they can only take on a certain number of whole values $(1, 2, 3, 4, 5, \ldots)$, whereas continuous values can take on an infinite number of possible values within a range (between the range 1.0 and 2.0 the values can be $1.1, 1.01, 1.001, 1.0001, \ldots$).
- *Boolean* is a binary value, and it relates to a data type that can have two possible values representing *true* or *false*, typically 1 or 0 respectfully.
- *String* is a finite number of zero or more alphabetic characters and it usually make a distinction between lower and upper-case characters, such as $[A, B, C, \ldots, Z]$ and $[a, b, c, \ldots, z]$. In addition to characters, some strings also allow numbers (alphanumerical).

Feature quality measures

Each feature need to be compared to each other because not all features provide equal quality to the classification or regression task [17, 18]. *Feature quality measures* will not be covered in depth in this thesis. It is mentioned because estimating the usefulness of a feature for predicting the label is one of the most important tasks in ML [17]. Some common measures are *information gain*, *Gain-ratio*, *distance measure*, Minimum Description Length (MDL) and *Gini-index*.

The previously mentioned measures only estimate the quality of one feature independently of the context of other features [17]. For problems with strong dependencies between the features, algorithms such as *ReliefF* and its regressional version *RReliefF* should be used. Since these algorithms also take into account the context of other features.

2.2.3 Data preprocessing methods

"Garbage in, garbage out" is a common concept in computer science. The concept refers to the logical processes that computers and algorithms operate by, where their quality of the output (result) is determined by the quality of the input [20]. ML algorithms learn from data, more precisely the features from each data sample. The algorithms can only be as good as the data it receives, and it is, therefore, crucial that they receive the right features. Each algorithm requires features that are meaningful, of a specific type and scale.

This section will go through some basic methods for conversion between discrete and continuous values, the challenge of missing or unknown features and how to reduce the amount of features.

Feature discretisation

Many ML algorithms are known to produce better models by discretizing continuous features [21, 22]. *Discretisation* of continuous features is the process of splitting continuous values into a finite set of intervals that are treated as discrete values. The goal of discretisation is to find a set of cut points to partition the range into a smaller number of intervals [22]. The challenge is that when continuous values are discretized, there is always some amount of *discretisation error*.

Discretisation can be both unsupervised or supervised, but typically continuous values can be divided into intervals of equal width (Figure 9) or equal frequency (Figure 10). For each discretisation of continuous value, some information is lost since values within the same range cannot be distinguished anymore. The goal for both unsupervised and supervised discretisation algorithms is to reduce the error between the original continuous value and the discretized value, so to avoid the loss of information in the discretisation process. The algorithms should find the *optimal number of intervals* and the *optimal boundaries for each interval* [17].

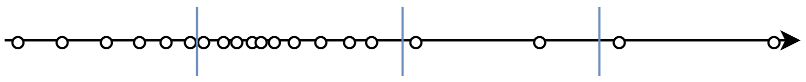

Figure 9: Equal width discretisation

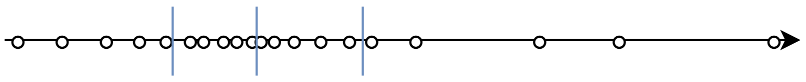

Figure 10: Equal frequency discretisation

Missing feature values

In real world data sets, it is usually the case that some of the feature values are missing. They can also be corrupt or incorrect, but this is much harder to discover in large data sets. One strategy is to discard the samples with missing values [14], but this is not a good strategy when the dataset is small. The non-missing values in the sample can also contain information necessary for the problem. Missing values can either be ignored or replaced by an estimation, called *imputation*.

Some of the most common schemes are to replace the values with zero or calculate its value based on a probability distribution [23]. The probability distribution for continuous values can be substituted by the mean (average) of the available feature values, whereas discrete values can be replaced by the most frequent value [14].

Feature selection and feature extraction

ML algorithms are known to perform worse when many features are not necessary for model prediction [24]. Some of the problems with high-dimensional datasets (i.e. datasets with more than 10 features) is that not all the measured features are important for understanding the underlying problem of interest [17]; it can be affected by the *curse of dimensionality* (described in Section 2.2.4); it increase the complexity and thereby the run-time of the learning algorithm; and it is harder to interpret, explain and visualise. Techniques to reduce the number of total features are usually performed to avoid these problems, and it can be divided into feature selection and feature extraction [17].

Feature selection is the process of selecting a subset of relevant features, and it should be used when the dataset contains many features that are either unneeded, redundant or irrelevant [25]. These features can be removed without loss of information. The process will either include or exclude features present in the dataset without changing them. Feature selection techniques can be divided into three categories [17]: filtering, wrapper and embedded. Where each category of techniques have both advantages and disadvantages.

Filtering is the simplest and quickest methods for feature subset selection. These methods estimate the feature quality via a function and the best n features out of total m features are selected, where $n < m$. The functions usually consider the features independently of each other (e.g. MDL), but features with strong dependencies can be evaluated by functions that take dependencies between the features into consideration (e.g. ReliefF). The number n can be chosen in advance, or determined dynamically by the number of features whose quality exceeds some threshold value [17].

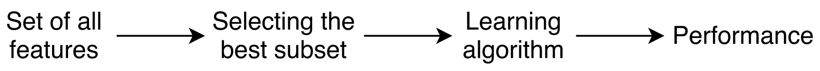

Figure 11: Filtering methods

Wrapper methods are slower than its counterparts, but it can provide the best feature subset. It uses algorithms that search the space of feature subsets and testing the performance of each subset using a learning algorithm [24]. The subset that gives the best performance is selected for final use. There is 2^m possible subset to search with a total of m features. An exhaustive search is often impractical, so most wrapper methods incorporate a heuristic process to narrow the search space [24]. The disadvantage with wrapper methods is that it tends to be computationally intensive.

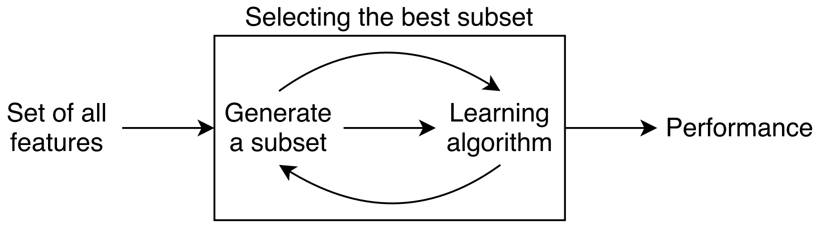

Figure 12: Wrapper methods

Embedded methods are ML algorithms that include an embedded feature selection method [24] in the learning algorithm and therefore becomes an inherent part of the learning process, such as decision trees. These algorithms learn by splitting the original dataset into subsets based on the feature quality values.

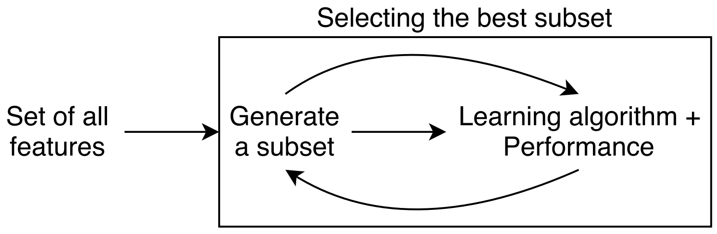

Figure 13: Embedded methods

The learning model can be enhanced by the feature selection and seemingly get better results if it is done as a preprocessing step [26]. When feature selection is performed before model selection and training, then the learned model will overfit (described in Section 2.2.4) and be biassed towards the dataset. Therefore, feature selection methods should be performed on the prepared fold (described in Section 2.2.4) right before the model is trained.

Feature extraction should be distinguished from feature selection as it does not select the best subset of features. Instead it uses a function to calculate new features based on the original features [17]. This function is a mapping from the original high-dimensional space to a new space with fewer dimensions. Principal Component Analysis (PCA) is a transformation of the original dataset by selecting the subspace that has the largest variance.

PCA generates a new coordinate system for the feature with the largest variance by any projection of the data set lies on the first axis (called the first principal component), the second largest variance on the second axis and so on [17]. PCA generates a representation of the original feature space that captures the content of the original data. It is sometimes the case that classification or regressional problems can be carried out in the reduced transformed space more accurately than in the original space [17].

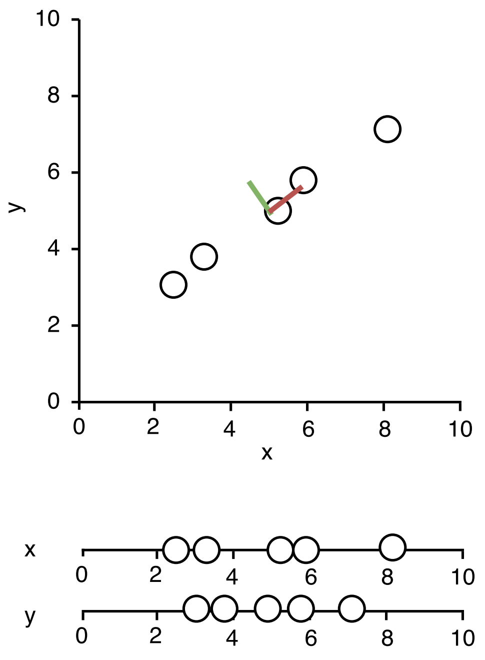

Figure 14: Original dataset before PCA [27]

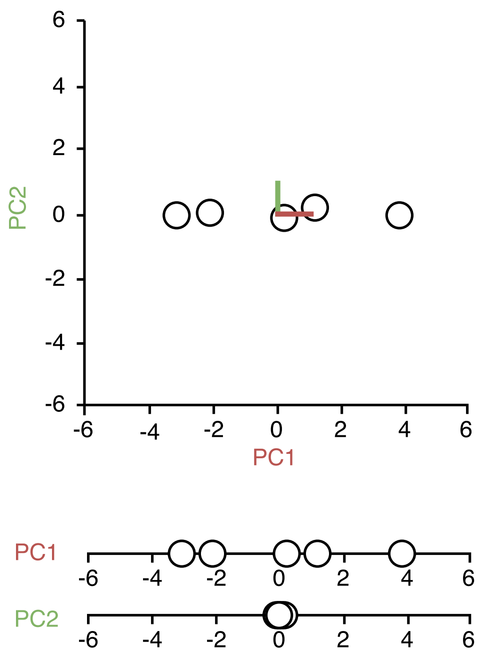

Figure 15: New dataset after PCA [27]

2.2.4 Applications and challenges

ML is widely applied in digital forensic to find digital evidence in electronic devices such as mobile phones, personal computers, hard disk, cameras, music players, game consoles and so forth. Text documents and e-mail messages represent a primary source of evidence during forensic analysis [28]. Domingos et al. [29] applied SVM to the task of authorship verification and attribution. They retrieved between 84% and 100% of e-mail messages sent from three different authors in an e-mail corpus with 156 messages.

File fragmentation occurs when file systems lay out the contents of files in a non-continuously way and is maybe the most investigated problem in digital forensics [28]. The problem is that it is hard to identify file fragments that belong together when the mapping provided by the file system is corrupt or otherwise missing. Li et. al [30] have shown a classification accuracy higher than 90% for common file types such as PDF, JPEG, and GIF.

ML represents a resource that can be exploited to facilitate the activity of the forensic analyst [28]. However, there exist several problems that must be considered when using ML algorithms. The problem of overfitting, the curse of dimensionality and no free lunch theorem will be discussed in this section.

Overfitting

The largest problem in ML is the problem of *overfitting* [31]. Overfitting occurs when the learned model begins to "memorise" the training data rather than learning to generalise from them. A learned model can perfectly classify the training data, but it typically fails to classify new or otherwise unseen samples. Methods such as cross-validation or pruning are used to avoid overfitting [31].

Cross-validation methods try to predict the performance of the learned model by using a validation set that is independent of the data used to train the model. These methods are divided into exhaustive or non-exhaustive cross-validation.

An example of exhaustive methods are *leave-p-out* and its particular case *leave-one-out*, where $p = 1$. Leave-p-put uses p samples as the validation set, and the rest of the samples will be utilised for training the model. This process is repeated for all the combinations it is possible to split the original dataset. This process produces $\binom{n}{p}$ possible combinations. However, it is impossible to train and validate all combinations as soon as the total number of data samples n becomes large [32]. Leave-one-out is illustrated in Figure 16.

An example of non-exhaustive methods are *k-fold* and its particular case *2-fold*, where $k = 2$. K-fold splits the original dataset randomly into k equal sized subsets called folds. $k - 1$ folds are used for training the model, and one fold is retained as the validation data. This is repeated until all the k folds have been used as the validation set. The results from all the folds can then be averaged to produce a single performance estimation [32]. k-fold with $k = 10$ is most frequently used in practice [17], but k is usually reduced when the learning takes much time. When $k = n$, where n is the number of samples, then k-fold becomes identical to leave-one-out exhaustive method. 4-fold cross-validation is illustrated in Figure 16.

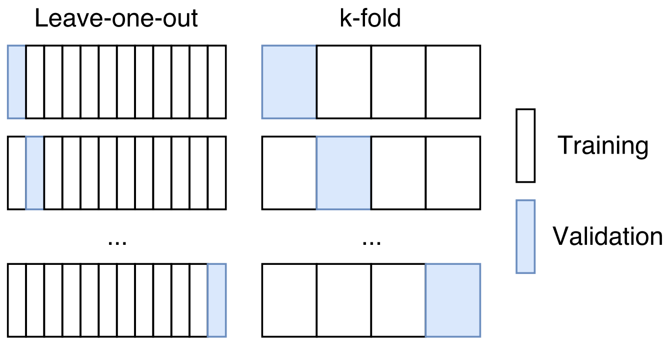

Figure 16: Cross validation methods

Curse of dimensionality

The second biggest problem in ML is the *curse of dimensionality* [31]. This expression was used by Bellman in 1961 to refer to the fact that many algorithms that work fine in low dimensions do not necessarily behave similarly in higher dimensions. The goal for ML algorithms is to generalise beyond the samples in the training set, but correctly generalising becomes exponentially harder as the dimensionality grows [31].

There is a maximum number of features where the performance of a classifier will degrade rather than improve. Figure 17 illustrates the curse of dimensionality. There is a limited number of features where the performance (accuracy) of a classifier improves, but the performance is reduced as the number of features is increasing. The curse is caused by the fact that a fixed number of training samples covers only a small fraction of the total possible input space [31]. There are many ways to mitigate the curse of the dimensionality, including increasing the number of training samples, feature selection and dimensionality reduction (feature extraction). However, there is no single solution to the many difficulties caused by the curse of dimensionality [33].

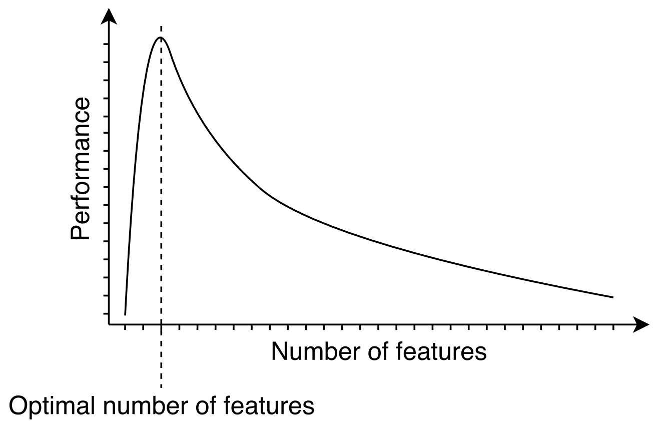

Figure 17: Curse of dimensionality

No free lunch

The *no free lunch* theorem states that there is no ML algorithm that outperforms all others in any given task [34]. ML algorithms generate a representation of reality by discarding unnecessary details and information. This representation (model) is generated based on some assumptions, which may hold in some situations but may not hold for other situations [35]. Each model must use some knowledge or assumptions beyond the data it is given to generalise beyond it [31].

2.3 Graph theory

Graph theory algorithms will be heavily applied in this thesis to analyse datasets of communicating OCGs. Therefore, Section 2.3 will cover the fundamentals of this mathematical field, and provide the knowledge of graph terminology, how graphs are represented and how they are interpreted. This section serves as an introduction to understand SNA, as it is the process of investigating social structures through the use of graph theory. SNA will be covered in Section 2.4, where it is discussed the various levels of analysis on an ego or complete network. This section also covers the graph construction algorithms that is used in this research to evaluate how they can be applied in digital forensic processes.

Graph theory is a mathematical field, and it is the study of graphs, which are mathematical structures used to model pairwise relationships between objects. More formally, a graph G consists of V, a non-empty set of *vertices* (also called *nodes* or *points*) and E, a set of *edges* (also called *arcs* or *lines*) [36]. Names and terminology vary between different disciplines, e.g. law enforcement agencies often call them entities and relationships [37] whereas social network analysis often call them actors and ties [38]. However, this section is going to follow the exact mathematical terminology.

Figure 18 illustrate a simple graph and the set of vertices and edges contained in it. *Set* is a collection of objects, and in Figure 18 the set of vertices contains $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and a set of edges $E = \{(v_1, v_2), (v_1, v_5), (v_1, v_6), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_3, v_5), (v_5, v_6)\}$.

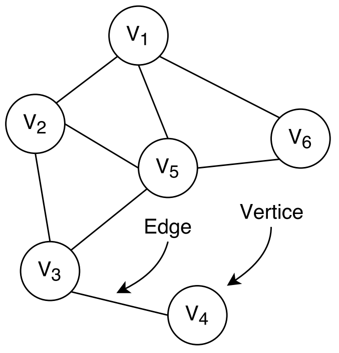

Figure 18: Graph

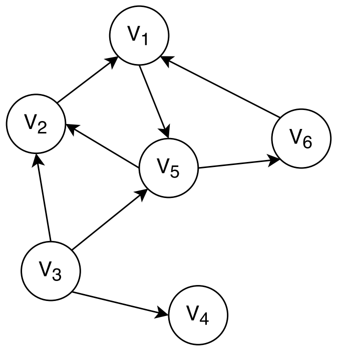

Figure 19: Digraph

Edges in a graph can have different properties depending on the context. Edges can be undirected or directed; the first is called a *graph* (or *undirected graph*), and the latter is known as a *digraph* (or *directed graph*). When the number of edges between vertices is important, one allows multiple edges between pairs of vertices; then it is called a *multigraph*. Graphs can also contain loops (a relationship to itself). However, undirected and directed edges are the two most common concepts when graphs are discussed.

Graph terminology			
Type	Edges	Multiple edges	Loops
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Table 1: Graph terminology [36]

2.3.1 Vertex degree

Vertices also have a wide variety of properties that will be covered in this subsection. Two vertices in a graph are called *adjacent* (or *neighbours*) if they are endpoints of an edge, *endpoints* is vertices at either end of an edge e (the edge is also said to *connect* two vertices). For example, v_3 and v_4 are adjacent in Figure 18.

Vertex degree keeps track of how many edges are incident with a vertex, by counting the number of edges that share one vertex. So the *degree* of a vertex in a graph is the number of edges incident with it [36] (a loop contributes twice to the degree), and is denoted $deg(v)$ for vertex v. E.g. the vertex degrees in the graph in Figure 18 is $deg(v_4) = 1$, $deg(v_6) = 2$, $deg(v_1) = deg(v_2) = deg(v_3) = 3$ and $deg(v_5) = 4$. A vertex with degree zero, $deg(v) = 0$, is called *isolated* because it is not connected to any other vertex in the graph. A vertex with degree one, as vertex v_4 in Figure 18, is called *pendant*, since it is pendant on the other connected vertex to stay connected to the graph.

Inspecting the degree of vertices are important in many fields that use graph theory to model and produce a solution to their problems. For example in sociology to model the central and most important actors in a group of people, and this is discussed in Section 2.4.

A digraph has different properties from what is previously discussed. Since the edges now have a direction. Consider the edge (u, v), u is said to be *adjacent to* v and v is said to be *adjacent from* u [36]. For example, the edge (v_3, v_4) in Figure 19 illustrate this, and v_3 is called the *initial vertex* of the edge, and v_4 is referred to as the *terminal vertex* (or *end vertex*) of the edge. Consequently, the initial and terminal vertex of a loop are the same.

Vertex degree in a digraph is also denoted according to the direction of the edge. The *in-degree* of a vertex, denoted by $deg^-(v)$, is the number of edges with v as their terminal vertex, and the *out-degree* of a vertex, denoted by $deg^+(v)$ is the number of edges with v as their initial vertex. A loop contributes one to both in-degree and out-degree. E.g. in-degree of vertex v_1 in the digraph in Figure 19 is $deg^-(v_1) = 2$ and the out-degree is $deg^+(v_1) = 2$. Since this vertex is the

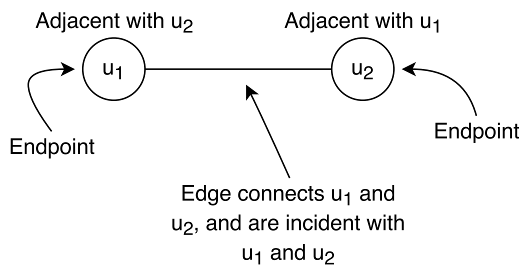

Figure 20: Graph vertex degree properties

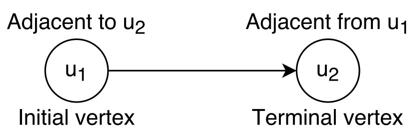

Figure 21: Digraph vertex degree properties

terminal node to two edges (from v_2 and v_6, and the initial node for one edge (to v_5). Each edge has an initial and terminal vertex, so the sum of in-degree and out-degree of the graph are the same. Let $G = (V, E)$ be a digraph, then $\sum_{v \in V} deg^-(v) = \sum_{v \in V} deg^+(v) = |E|$ [36].

In a digraph, a vertex with only out-going edges is referred to as a *source*, and one with only in-going edges is known as a *sink*. Vertex v_3 in Figure 19 illustrates a source, and vertex v_4 shows a sink.

2.3.2 Subgraph

A *subgraph* is a subset of the vertices in a network, and all the edges linking these vertices. In other words, it has a partial one-to-one correspondence of vertices and edges of the original graph and the subgraph. Any group of vertices can form a subgraph, as long as the edges are preserved. Figure 22 illustrate this property.

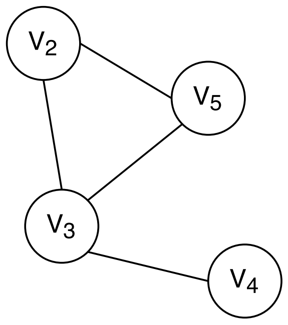

Figure 22: Subgraph of Figure 18

Component

A *component* consists of a subset of vertices in which all the vertices are connected to one another by at least one path [38]. For example, Figure 23 illustrate four components in an undirected subgraph, they are $\{(v_1, v_2, v_3), (v_4), (v_5, v_6, v_7, v_8), (v_9, v_{10})\}$. A vertex with no incident edges is itself a component (see vertex v_4 in Figure 23). A graph can be called *connected* when that graph has exactly one connected component, consisting of the whole (sub)graph.

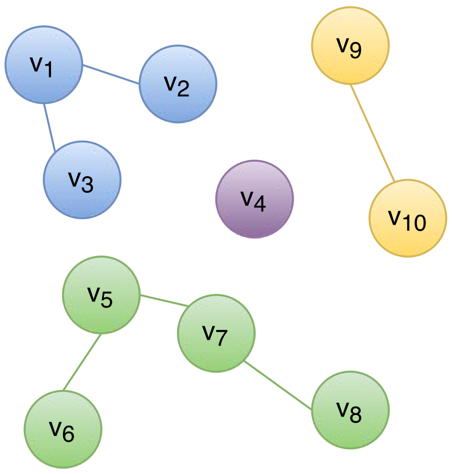

Figure 23: Connected components (four components)

In a directed subgraph, the components are said to be either *weak* or *strong*, depending on the edge direction. Figure 24 illustrate component properties. A strong component is a subgraph in which there is a path from every vertice to every point following all the edges in the direction they are pointing. A weak component is a subgraph which would be connected if we ignored the direction of the edges.

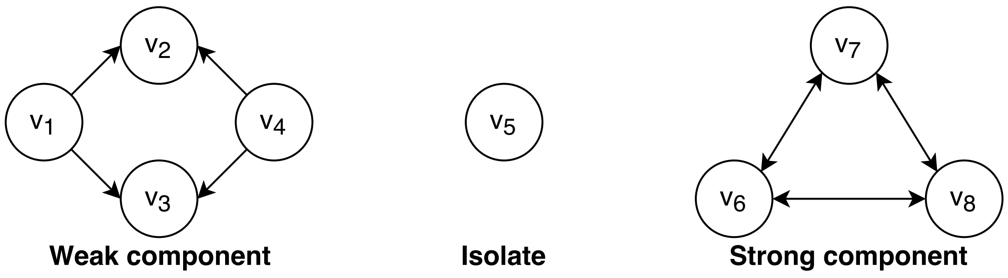

Figure 24: Types of directed components

Cliques

A *clique* is a complete subgraph where each pair of vertices are connected, and the clique of largest possible size is referred to as a *maximum clique* [39]. A maximum clique is the largest clique that cannot be extended by a new vertex. Clique can be extended to *k-clique*, where k is the size of the clique. E.g. the graph in Figure 18 have three 3-cliques: (v_1, v_2, v_5), (v_1, v_5, v_6) and (v_2, v_3, v_5).

2.3.3 Graph representation

We have only illustrated graphs visually with vertices and edges until now. However, this is only an effective illustrations for humans to understand graphs and computers will have difficulties to interpret this form of model. Therefore, there exist many different forms of modelling graphs that are easier for computers to process. Selecting the method for representing the graph will depend on the problem. There are two common representations to represent a graph; they will be discussed in the following subsections.

Adjacency list

The first method is to use an *adjacency list*, which specifies the vertices that are adjacent to each vertex of the graph [36]. An adjacency list can be used for both undirected and directed graphs. However, it cannot represent graphs with multiple edges. The adjacency list is also the best representation of large and *sparse graphs*, which means a graph with few edges between elements in the set of vertices. The adjacency list store a graph in a more compact form than an adjacency matrix, however, this effect decreases as the graph becomes denser (more connections between vertices). Also, for some algorithms, there is a significant advantage to having a list of the adjacent vertices [40].

However, the limitations with the adjacency list are that it takes more time to insert or delete an edge, to check if there is an edge between two vertices, and it is not an efficient implementation for graphs that changes dynamically [40].

Vertex	Adjacent vertices
v_1	v_2, v_5, v_6
v_2	v_1, v_3, v_5
v_3	v_2, v_4, v_5
v_4	v_3
v_5	v_1, v_2, v_3, v_6
v_6	v_1, v_5

Table 2: Undirected adjacency list for Figure 18

Initial vertex	Terminal vertices
v_1	v_5
v_2	v_1
v_3	v_2, v_4, v_5
v_4	
v_5	v_2, v_6
v_6	v_1

Table 3: Directed adjacency list for Figure 19

Adjacency matrix

The second method is to use a matrix representation, and there are two commonly used matrices to represent graphs. The first is an *adjacency matrix*, which is an $n \times n$ zero-one matrix with 1 as its (i, j)th cell when v_i and v_j are adjacent, otherwise 0. A *cell* is the intersection of a *row* with a *column*, and each cell represents the presence or absence of ties between row i and column j. The values of cells are referred to as (i, j). Note that the adjacency matrix can contain numerical values [36], e.g. for the number of multiple edges (and loops) or weighted edges.

A matrix is either symmetric or asymmetric for undirected or directed graph respectfully. Figure 25 is a symmetric matrix since the graph has undirected edges, so this means that an edge contributes once to each vertice.

The advantage with a matrix representation is that it is very convenient to work with, as checking for existing edges, adding and removing an edge is done by examining the (i, j)th cell in the matrix. The limitations are that it will consume a huge amount of memory [40], so they are often preferred when the graph is dense. It is also less efficient for analysing each of the incident vertices since it has to traverse all the coloumns in one row. It is also difficult to add or remove a vertex, so using matrices for a dynamic structure is quite slow.

$$\begin{array}{c|cccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ \hline v_1 & 0 & 1 & 0 & 0 & 1 & 1 \\ v_2 & 1 & 0 & 1 & 0 & 1 & 0 \\ v_3 & 0 & 1 & 0 & 1 & 1 & 0 \\ v_4 & 0 & 0 & 1 & 0 & 0 & 0 \\ v_5 & 1 & 1 & 1 & 0 & 0 & 1 \\ v_6 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Figure 25: Adjacency matrix for graph in Figure 18

$$\begin{array}{c|cccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ \hline v_1 & 0 & 0 & 0 & 0 & 1 & 0 \\ v_2 & 1 & 0 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 1 & 0 & 1 & 1 & 0 \\ v_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_5 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_6 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Figure 26: Adjacency matrix for digraph in Figure 19

Incidence matrix

The second matrix representation is *incidence matrix*. This is a matrix that represents a graph on the ordering of the set of vertices V and set of edges E, so there is a 1 when edge e_j is incident with v_i. This results in a $n \times m$ matrix where n is the number of vertices (|V|) and m is the number of edges (|E|). Incidence matrix also can represent multiple edges and loops.

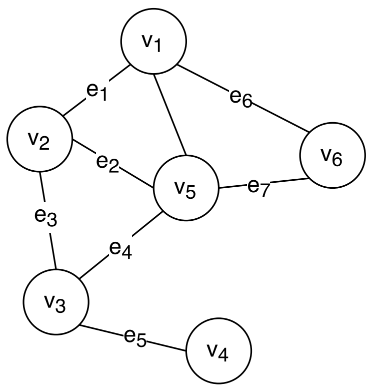

Figure 27: Graph with edge labels for Figure 18

$$\begin{array}{c|ccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ v_1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ v_2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ v_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ v_5 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ v_6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Figure 28: Incidence matrix for Figure 27

2.4 Social network analysis

The two previous chapters with ML (Section 2.2) and graph theory (Section 2.3) sections laid the foundation for understanding graph construction algorithms and SNA respectfully. They will be discussed in this section.

SNA is a multidisciplinary area involving social, mathematical (graph theory), statistical and computer science. It uses methods and analytical techniques to uncover the social relations that form from individuals and groups, the structure of those relationships, and how relations and their structure influence (or are influenced by) social behaviour, attitudes, beliefs and knowledge [38]. Social relationships such as family, work colleagues and friends form different *social networks*, and it is normal for a person to be a member of many different social networks. For example, a friendship network would be one relation, work colleagues another relation and so on.

Social networks can be formally defined as a graph, with a set of *relations* (edges) and a set of *actors* (vertices). SNA encourages the separating of various social networks according to different relations. However, networks can also represent multi-relational data. The terminology in SNA is more orientated towards social terms but refers to the same abstract objects as in the mathematical terminology.

SNA is usually conducted via interviews with the actors of interest [38], where specifically crafted questions is necessary to create a *boundary* around a particular network. A boundary defines actors that can be considered to be inside the network and which ones are outside the network. The *population* of interest have been selected by specifying the boundary.

There are two types of social networks. The first is an *ego network* – which is comprised of a focal actor (called *ego*) and the people to whom ego is directly connected is referred to as *alters* [38], including the edges between the alters. Alters in an ego network has their own ego network. The second is a *complete* *network* – where an entire set of actors and the ties linking these actors together are studied.

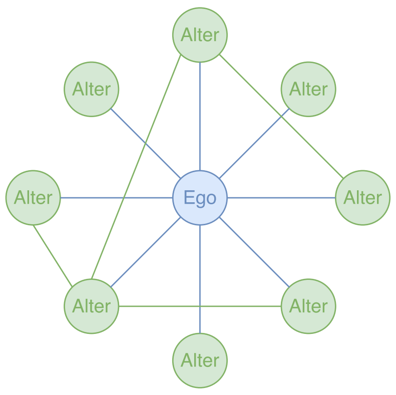

Figure 29: Ego network

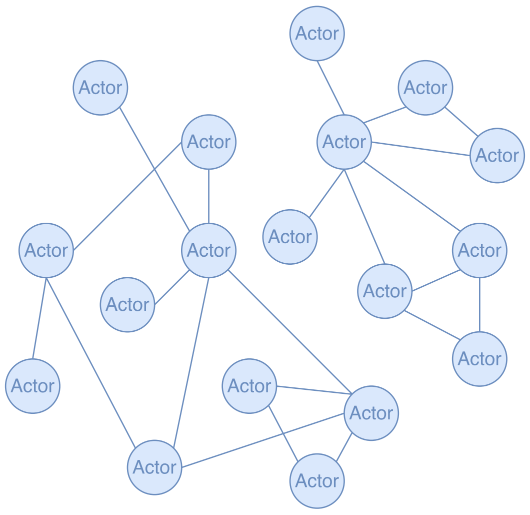

Figure 30: Complete network

The edges in both ego and complete networks can be directed or undirected. The edges represent a relation; then the arrowheads indicate the direction of the relation. When the graph is directed, it shows *directed relations* in SNA terms.

Each actor is represented twice in an adjacency matrix, once in the row and once in the column. The senders are found in rows and receivers in columns. The diagonal of the matrix, therefore, represents a sender's relations with herself. An actor's tie with themselves (self-loops) are usually not of interest when analysing a network, and therefore, the diagonal of the adjacency matrix tends to be ignored [38].

2.4.1 Levels of analysis

Network analysis breaks down the notion of a network as being composed of a series of levels such as actors (i.e. individuals); dyads (i.e. pairs of actors); triads (i.e. structures consisting of three actors); subgroups; and complete networks. Actor level analysis focuses on the individual, whereas dyad and triad level analysis focuses on the social structures. Dyad and triads can be viewed as building blocks for larger network structures. This section will cover the actor level methods used in the thesis, and briefly mentioning the other levels of analysis.

Actor level

The analysis of a network at the actor level is to describe the position of an actor within a network according to these methods outlined in this section. Actor level analysis helps to understand how individual actors are positioned within a particular network by answering the questions of who is important to that network, who makes things "happen" in the network or holds the network together in times of distress [38].

Measures of centrality are used to uncover the answers to these questions. Centrality is measured by an actor's *degree*, *betweenness*, *closeness* and *eigenvector*. Degree is seen as a more local measure as it only takes into account directly connected edges. Whereas betweenness, closeness and eigenvector are seen as global measures because they also take into account directly linked actors.

Degree centrality is calculated by counting the number of directly adjacent vertices an actor has in a network, i.e. $deg(v)$ as described in Section 2.3.1. The formula to calculate the degree centrality from a symmetric adjacency matrix A is given by Equation 2.4, where i is the actor of interest, A_{ij} is the cell value from actor i to actor j and n is the total number of vertices.

$$C_D(i) = \sum_{j=1}^{n} A_{ij} = \sum_{j=1}^{n} A_{ji} = deg(i) \tag{2.4}$$

When degree centrality is calculated on a graph, it can only be viewed as a measure of an actor's level of involvement or activity in the network.However, the direction of the edges can often show interesting insights into group structure and the placement of individuals within this structure [38]. Therefore, in-degree and out-degree centrality has been developed for digraphs. In-degree centrality is the number of edges adjacent from other vertices. It is often used as a measure of prestige or popularity [38]. The formula for in-degree centrality for an asymmetric adjacency matrix A is given by Equation 2.5:

$$C_{in}(i) = \sum_{j=1}^{n} A_{ij} = deg^{-}(i) \tag{2.5}$$

Out-degree centrality is the number of edges adjacent to other vertices. It is often used as a measure of expansiveness [38]. The formula for out-degree centrality from an asymmetric adjacency matrix A is given by Equation 2.6:

$$C_{out}(i) = \sum_{j=1}^{n} A_{ji} = deg^{+}(i) \tag{2.6}$$

A disadvantage of degree, in-degree and out-degree is that these centrality measures can only be used to make meaningful comparisons among actors in the same network or networks of the same size [38], i.e. they have the same number of actors. Linton C. Freeman [41] developed a way to convert these centrality measures into proportions by normalising the scores. Normalising is done by taking the original centrality scores and dividing them by the total number of actors, not including the actor in consideration. Normalising degree centrality is given by Equation 2.7. By normalising the centrality scores, it becomes in proportion to the network. Therefore, they then can be compared to centrality scores from other networks or networks of different sizes.

$$C'_D(i) = \frac{C_D(i)}{n-1} \tag{2.7}$$

The equation is the same for normalising in-degree and out-degree, given by Equation 2.8 and Equation 2.9 respectfully.

$$C'_{in}(i) = \frac{C_{in}(i)}{n-1} \tag{2.8}$$

$$C'_{out}(i) = \frac{C_{out}(i)}{n-1} \tag{2.9}$$

Degree, in-degree and out-degree centrality measures are not considered the most powerful measures of centrality. Since they do not consider the rest of the network; they only look at the immediate edges of each actor. In other words, they ignore the other actors and edges in the network.

Betweenness centrality takes into consideration the rest of the network when computing a score for an individual actor. Betweenness capture the dimension of centrality that it is not important how many people they know, but rather where they are placed within that network [38]. Betweenness centrality looks at how often an actor rests between two other actors. More formally, it calculates how many times an actor sits on the *geodesic* (i.e. the shortest path) linking two other actors together. There may be more than one shortest path between two vertices. If an actor rests between many other actors in the network, then this actor can significantly influence the network by choosing to withhold or distort information she or he receives [38].

It is computed by counting the number of times actor k rests on a geodesic for actors i and j. The adjacency matrix can be both asymmetric and symmetric, but the data should be binary. The formula is given in Equation 2.10, where ∂_{ij} is the total number of shortest paths from vertice i to j and ∂_{ikj} is the number of those paths that pass through k. Normalised formulas for betweenness centrality is given for digraphs in Equation 2.11 and graphs in Equation 2.12.

$$C_B(k) = \sum \frac{\partial_{ikj}}{\partial_{ij}}, i \neq j \neq k \tag{2.10}$$

$$C_B' = \frac{C_B(k)}{(n-1)(n-2)} \tag{2.11}$$

$$C_B' = \frac{C_B(k)}{(n-1)(n-2)/2} \tag{2.12}$$

Betweenness centrality score can be normalised to compare it with another network of different size. Research has shown that betweenness centrality results in a larger amount of variance (difference) among actors than other centrality measures. Therefore, it best captures the most important actors in the network [38], i.e. leaders or the most influential network actors. Those who is more important to the network thus stands out much more clearly using betweenness centrality than any other forms of centrality.

Closeness centrality (*farness*) capture an actor's potential pendency. Closeness centrality is defined as the sum of distances between one actor and all the other actors. The distance measure is defined by the length of their *shortest paths*. The formula is given in Equation 2.13, where $d(ij)$ is the distance (length of the shortest path) connecting actor i to actor j. The normalised formula for closeness centrality is given in Equation 2.14.

$$C_C(i) = \sum_{j=1}^{n} d(ij) \tag{2.13}$$

$$C_C'(i) = [C_C(i)]^{-1}(n-1) \tag{2.14}$$

The network must hold binary data, and any isolates must be removed from the graph [38]. Isolated vertices would hold an infinite distance from all other actors in the network because isolates can never be reached. Closeness centrality can therefore only be measured on fully connected and binary networks. It can be used in a directed or undirected graph, but the graph must also be strongly connected when the graph is directed. Strongly connected graph means that all the actors can reach one another by some path, paying attention to the direction of the edge. A weakly connected graph would be one where all actors are linked by some path, ignoring the direction of the edges. Higher values of closeness centrality score indicate higher centrality in the network.

Eigenvector centrality expands on the notion of degree centrality to also include their alters' edges. Eigenvector centrality is defined as the sum of an actor's edges to other actors, weighted by the alters' degree centrality. Therefore, the centrality score becomes weighted towards those actors who have alters with high degree centralities [38]. Alters can drastically affect actors eigenvector centrality score as one's importance is based on their friend's importance. The formula is given in Equation 2.15, where $\lambda \neq 0$.

$$C_E(v) = \frac{1}{\lambda}\sum_{u=1}^{n} x_{u,v} C_E(x_u) \tag{2.15}$$

Dyadic level

A *dyad* consists of two actors and the edge that link these actors together. When pairs of actors are studied, the focus will often look at how ties are initiated, continued and terminated; what kind of resources get exchanged between them; reciprocity; and the strength of ties [38]. Not many analyses take place on the dyad level, but it is important to describe as one gathers data on an entire network on the dyadic level.

As previously mentioned, actors often have many (and different) relations with other individuals. These relations also differ in strength and direction. For example, a person often has multiple individuals that they consider their friends, but friendship does not have to be reciprocated. On Facebook, friendship is reciprocated via accepting a friend request, and they can, therefore, be considered undirected. However, on Twitter, persons can follow another without requiring that person to reciprocate this action, and they can, therefore, be considered directed.

One friendship relation does not equal another relation. For example, out of all those that they consider their friends, there are only one (or a few) that they consider their best friend(s). Therefore, the edges in both of these scenarios can be weighted, to measure the level of closeness, familiarity, warmth, affection and so forth of the friendship. The edge towards the best friends will be of greater strength than the rest.

The properties of dyads such as the kind of edge relation, strength, direction and duration can be understood through the use of a *dyad census* [38].

A dyad census categorises the edges in dyads into some state. In a graph, there is a binary relationship between the two actors. Binary dyads consist of relations that exist (they are connected), or the relation does not exist (isolates). A digraph, however, has three possible states: M (mutual), A (asymmetric) and N (null). These states are illustrated in Figure 31. A mutual edge has a bidirectional edge between two actors; an asymmetric edge is unidirectional, and null has no connection between the actors.

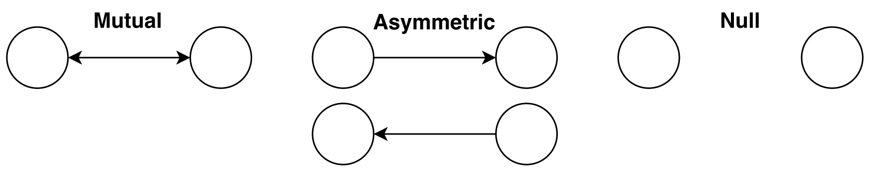

Figure 31: Dyad census MAN states

Triadic level

A *triad* consists of three actors and all the edges between them. It can, therefore, be seen as being composed of different kinds of dyads among a set of three actors [38]. Triads are also the simplest structures in which we can see the emergence of hierarchy [42]. A *triad census* can be viewed as an extension of the dyad census, as it looks at the number of mutual, asymmetric and null states between the actors. A triad census consists of a three digit number. The first number indicate the number of mutual dyads, the second indicate the number of asymmetric dyads, and the third indicate the number of null dyads. Figure 32 illustrate the four possible representations of undirected triads; they are closed triad, open triad, connected pair and unconnected.

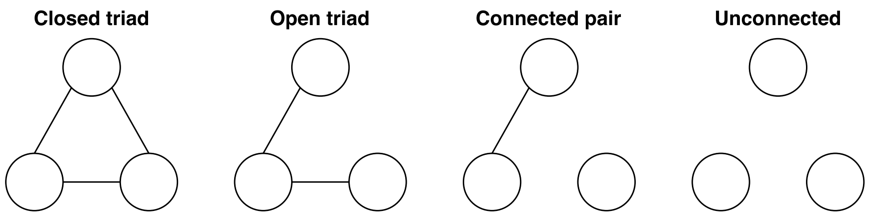

Figure 32: Representations of undirected triads

For digraphs, there are 16 possible triads, as illustrated in Figure 33. The letters U, D, C, and T is used to distinguish between different triads in which these numbers are the same, where "U" is for up, "D" for down, "C" for cycle and "T" for transitive. *Transitivity* of a relation means that when there is an edge from v to u, and also from u to w, then there is also a tie from v to w.

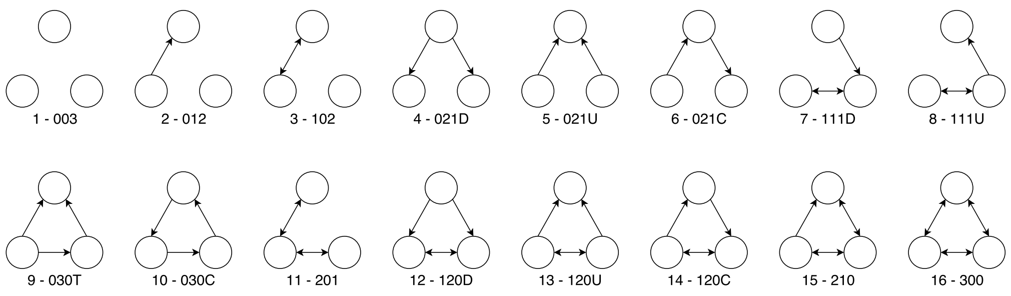

Figure 33: Triad census

Subset level

Any group from the set of vertices set can form a subset, as long as all the edges linking them together is included in the subgraph. So an ego network can be seen as a subgraph that is centred on a certain vertex.

Identifying the components of a network is important, as some network analyses require the network being connected. For example, closeness is best computed on strongly connected graphs [38]. Any network that can be seen as a disconnected graph might be treated as several separate networks for conducting certain analysis, such as closeness centrality.

Alternative techniques must be used to uncover subsets within the network when the network cannot be broken into different components. Clique, n-cliques and k-core are three techniques to find subsets within a graph.

A clique (described in Section 2.3.2) consists of a subgroup of people who are directly connected to one another through mutual edges. Cliques are usually analysed on graphs, but when cliques are analysed on a digraph, it is important to pay attention to the direction of the edges. Cliques would only include those edges that are mutual, and as such, a digraph can be expected to hold fewer cliques than a graph [38]. Analysis of cliques is interested in mutuality of ties, as all actors are having ties to all other actors in the clique. This approach has the problem of *clique overlap*, as it can cause confusion in interpreting the results of the clique analysis. A clique overlap occurs whenever one (or more) actor(s) from one clique can be included as a member(s) of another clique.

Therefore, the *n-clique* technique was developed as the definition of a clique is often seen as being overly strict. A clique will include or exclude actors based on their directly connected vertices, whereas n-cliques is linked by a path of length n or less. The rule by which one allows an actor to join a clique has been relaxed. The value of n should be kept at three or less, as one move away from the sociological understanding with any number higher than tree [38].

K-core is another way to define a subset within a graph. It is built on the concept of degree centrality, as an actor is part of a k-core if they have (at least) a degree centrality of k within that group [38]. It is easier for an actor to join

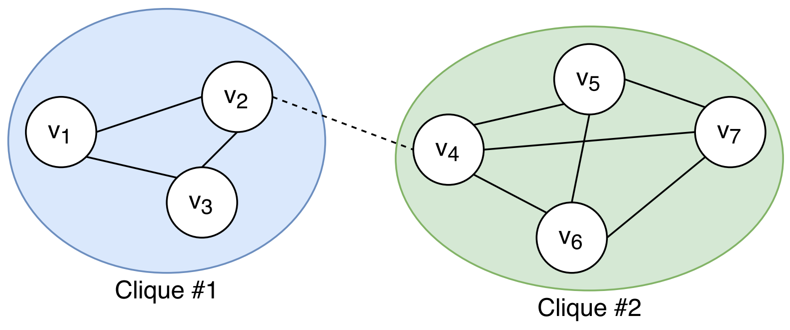

Figure 34: Clique

a subgroup when the value of k is low, and the subgroup will increase in size. It becomes more difficult for an actor to join the group when the value of k increases.

Identifying actors that act as a bridge between different components can help taking down a criminal network. By removing either vertex v_2 or v_4 in Figure 34 will create two disconnected components.

Network level

Network level analysis looks at the graph as a whole. This type of analysis hope to uncover some features of the network that characterise the network as a whole [38], such as density. *Density* refers to the proportion of edges in a network that are present, by counting the number of edges that exist in a network in a proportion of the potential edges that could exist. The formula for a graph is given in Equation 2.16, where L is the actual number of lines present in the graph and n is the number of vertices in the graph. Each node is potentially connected to all others (except themselves). In digraphs, we are interested in the direction of the edges, reflected in the denominator of density formula for digraphs in Equation 2.17.

$$d = \frac{L}{n(n-1)/2} \qquad (2.16)$$

$$d = \frac{L}{n(n-1)} \qquad (2.17)$$

One issue with density is that one (or more) person(s) can raise the density score when they have a disproportionately higher number of edges to others in the network [38]. *Degree centralisation* score is a good way to see the extent to which a graph's density score depends on the edges of one vertex. Degree centralisation is based on degree centrality, and the formula is given in Equation 2.18. Where $C_D max$ is the largest degree centrality score, $C_D(n_i)$ is the degree centrality of actor n_i, and $max \sum C_D max - C_D(n_i)$ is the theoretical maximum possible sum of differences in actor centrality [38]. Note that it should be computed on undirected and binary graphs.

$$C = \frac{\sum C_D max - C_D(n_i)}{max \sum C_D max - C_D(n_i)} \tag{2.18}$$

2.4.2 Graph construction approaches

Graph construction algorithms have been applied to the problem om label propagation, as they can automatically suggest labels for social media objects (music, video, pictures). Manually annotating these objects becomes infeasible as the volume of the data grows. Additionally, users may provide incorrect annotations.

Label propagation algorithms make two assumptions [43]. First, the labels of the initially labelled data object should remain unchanged. Second, data objects that are similar to each other should be assigned the same label. Semi-supervised learning algorithms, introduced in Section 2.2, assign labels to unseen data objects based on known labels for other data objects. This assignment can be performed through graph paths that connect labelled to unlabeled vertices [43]. The edge weights in these *similarity graphs* may represent pairwise similarities (distances) between the data objects. These graphs should reflect the relationships between the entities being labelled [43]. Therefore, the construction of the similarity graphs is critical to the label propagation performance.

This section will focus the discussion on the *neighbourhood* graph construction algorithms: k-NN, e-neighbourhood and b-matching. Which can be found in the work from Pitas [43]. He also discusses more advanced graph construction algorithms for label propagation.

In *neighbourhood methods*, the graph is constructed by connecting each node to its closest ones. Closeness between nodes is determined by a *distance* or *similarity function*, as discussed in Section 2.2.1. Neighbourhood methods use the distance between vertices to construct a similiarity graph, and they construct sparse graphs [43].

k-Nearest neighbour method

In k-NN graphs, each vertex is connected to its k nearest vertices (called *neighbours*). This approach often constructs asymmetric graphs [43], since it is not guaranteed that vertex v is the closest neighbour of u, when vertex u is the nearest neighbour of v. It also produces *irregular* graphs, since certain vertices end up with a higher degree than others. A *regular* graph, in contrast, is a graph where each vertex has the same number of neighbours.

See the original graph in Figure 35, where the evaluated vertex is v_i. There is a directed edge from v_i to u whenever u is one of the k nearest neighbours. Figure 36 illustrate the three ($k = 3$) nearest vertices of vertex v_i.

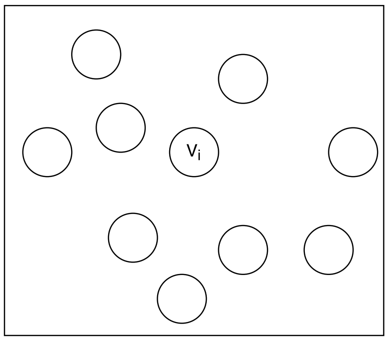

Figure 35: k-NN original graph for vertex v_i

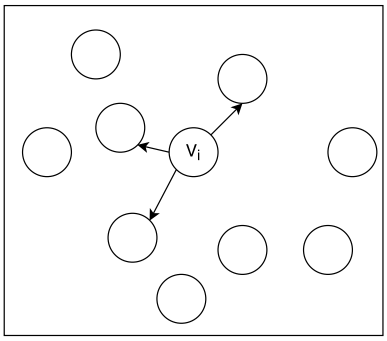

Figure 36: 3-NN neighbourhood for vertex v_i

e-neighbourhood method

In the e-neighbourhood method, a vertex is connected to all the vertices within a predefined distance e, where $e > 0$. Such a graph construction is sensitive to parameter e selection, and it often leads to graphs having disconnected components [43]. For these reasons, the k-NN method shows advantages over the e-neighbourhood method, and it is usually preferred in practice.

See the original graph in Figure 37, where the evaluated vertex is v_i. There is a directed edge from v_i to u whenever the distance to vertex u is less than or equal to e. Figure 38 show the e-neighbourhood to vertex v_i, where e is unspecified.

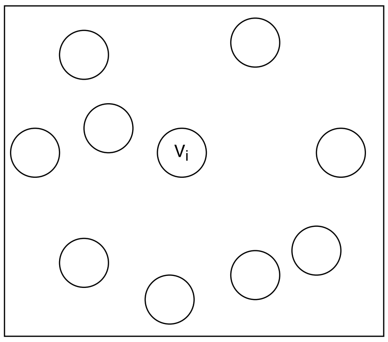

Figure 37: e-neighbourhood original graph for vertex v_i

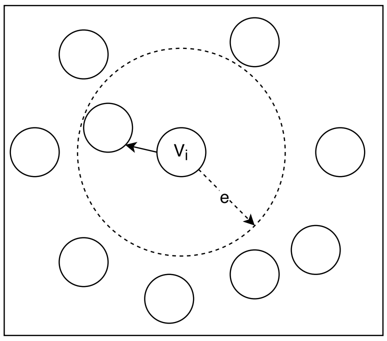

Figure 38: e-neighbourhood for vertex v_i

b-matching method

k-NN and ϵ-neighbourhood graphs do not guarantee that the result will be a regular graph. However, the b-matching method proposed in [44] guarantees graph regularity. This approach involves two steps [43, 44]:

1. *Graph sparsification* – from starting with a complete graph, the edges are selected that will be present in the final, sparse graph P.
2. *Edge re-weighting* – weights are learned for the edges that were selected in the previous step.

The first step generates a binary matrix $P \in \{0,1\}^{N \times N}$, where the cells P_{iij} determine the presence of an edge between data samples x_i and x_j in the final matrix. The calculation of matrix P is performed by minimising the following function:

$$\min_{P} \sum_{ij} P_{ij} Dij \tag{2.19}$$

subject to

$$\sum_{j} P_{ij} = b, P_{ii} = 0, P_{ij} = Pji, \forall_i, j = 1, \ldots, N \tag{2.20}$$

The distance matrix D is calculated from $D_{ij} = \sqrt{W_{ii} + W_{jj} - 2W_{ij}}$, where W denotes the weight matrix. For the last step, the weights for the selected edges are learned by using one of three different schemes [43, 44]:

- Binary weights, $W = P$
- Gaussian kernel weights $W_{ij} = P_{ij} exp\left(-\frac{d(x_i,x_j)}{2\sigma^2}\right)$, where $d(x_i, x_j)$ is distance function between the vertex feature vectors x_i and x_j and σ is the kernel bandwidth
- Locally Linear Reconstruction (LLR)

2.4.3 Applications and challenges

SNA have mostly been applied in social sciences [38], but it has been implemented in both economy studies and digital forensic investigations. It has been used to study the lives of 172 terrorists in the Hamburg Cell [45]. This cell started the planning for the terrorist attacks on 9/11, and ultimately participated in the attack. Sageman [45] found the most common factor driving them was the social ties within their cell.

SNA is a tool that can be used to understand the social relations that form within a network of people. It represents a resource that can be exploited to facilitate the activities of the forensic analyst. As with any tool, the researcher needs to understand when it can be applied to the underlying problem and how best to avoid the common challenges of the field. This section will cover the SNA challenges of data acquisition and privacy and the graph construction challenge of distance metric selection.

Data acquisition

The first challenge of SNA is the data acquisition step. This step is crucial since it is hard to know all the actors beforehand. It will also define the boundary around the population of interest. We do not cover typical data anonymisation techniques that typically are carried out after the data acquisition process. Canali et al. [46] highlighted three main techniques to acquire data from social networks. They were network traffic analysis, ad-hoc applications and crawling the user graph.

Network traffic analysis is well known to information security professionals, as this technique is commonly found in Intrusion Detection System (IDS). It is a typical traffic sniffing and analysis technique that capture network packets on the wire. From these tools, it is possible to analyse the browsing of social networks to obtain the relations between the users.

Ad-hoc applications are a third-party application that uses the social network private Application Programming Interface (API). This allows the collection of user information through interacting with registered user profiles. However, it is often limited to the registered users for that third-party application, and therefore, the information is dependent on the population of the application.

Crawling is the most popular solution for data acquisition in social networks [46]. This technique crawls through the publicly available information about users. Crawling may take advantage of the publicly available APIs that some social network operators provide [46].

Privacy laws

All of the previously mentioned data acquisition techniques have their problems, but the problem they share is the issue of privacy laws. For the data to be meaningful, the researcher must know who the respondent was in order to establish a link to those other actors they indicated having a relationship with [47]. It is hard to balance the protection personal or sensitive data against the benefits of the study. Therefore, anonymity can not be garanteed during the data collection stage.

In 2006, Netflix published 10 million movie rankings by 500 000 customers, as part of a challenge to come up with a better movie recommendation system. This dataset was protected by removing personal details and replacing names with random numbers, and the dataset was thought to be anonymised. However, Narayanan and Shmatikov [48, 49] showed that this dataset could be de-anonymised by correlating the data with the public available IMDB[1] database. A significant portion Netflix customers could be re-identified because they had

[1]`http://www.imdb.com/`

posted their e-mail address information online.

Even though viewing habits is less sensitive than medical records, it shows that de-anonymisation of large datasets is possible. Therefore, the researcher has to be clear on who will see the collected data and what can happen as a result of someone seeing what they responded.

Selection of distance metrics

The choice of the distance metric strongly affects the resulting graph [43]. When no information is provided for the data samples, graph construction algorithms will usually use Euclidean distance to calculate the distance between the vertices. It is important for the selected distance metric to capture the underlying structure that may exist in the data. When prior knowledge of the data samples is available, for example, labels or clustering information, it can be incorporated into the distance metric to provide a better result [43].

2.5 Summary

This chapter has provided a thorough understanding of the various fields that is used in this thesis. So this section summarises the relevant topics that are covered in this chapter. The first chapters provided the knowledge and understanding of SNA and graph construction algorithms.

From section about organised crime (Section 2.1), it is important to understand how criminals organise in networks. These networks provide services to the rest of CaaS business model. By understanding the organisational structure, it will be possible to identify a point of attacks to dismantle these groups.

ML (Section 2.2) is not directly used in this thesis, but the ideas behind clustering can be transferred to graph construction. It is, therefore, important to understand data preprocessing methods, to avoid the curse of dimensionality. A few distance measures were discussed and according to the “no free lunch” theorem, no distance metric that outperforms the other metrics in any given task. The Euclidean distance measure is often preferred, but it is necessary to compare several distance measures to select the metric that performs the best.

The section about graph theory (Section 2.3) provided the necessary mathematical understanding of graphs. It is necessary as SNA methods use graph theory to analyse and understand social networks. The centrality measures will be used when solving the first research question, and graph construction algorithms will be used to solve the second research question.

3 Previous work

In the previous chapter, an introduction to the thesis was given, and theory on several central topics was presented. The following chapter presents previous work related to the contributions of this thesis. It provides an overview of the state of the art in social causes for organised crime, and a discussion of related work. Further, the current use of SNA in practice is presented.

Sociological causes for organised crime

In 1924, Sutherland recognised that most communities, independent of their income, race or sociological background, are organised for both criminal and anti-criminal behaviour. Sutherland [50] investigated the interactions and learning aspects that underlie criminal activities. He proposed the *differential association theory* that advocates that interaction with others who are offenders increase the likelihood of someone becoming and remaining an offender. Social peers can play a crucial role in the development of values and beliefs favourable to law violation [50, 51]. In this theory, Sutherland elaborated on nine postulates to discriminate at the individual level between lawbreakers and law-abiding citizens, where three is especially relevant for this thesis [50]:

- *Criminal behaviour is learned (behaviour)*
- *Criminal behaviour is learned in interactions with other persons in a process of communication*
- *The principal part of the learning of criminal behaviour occurs within intimate personal groups*

Differential association theory proposed individuals learn the values, attitudes, techniques and motives for criminal behaviour through interaction with others. However, one critique against this theory is the idea that people can be independent, rational actors and can have individual motivations [52]. The critique is that the theory does not take into account individual personality traits that might affect their susceptibility to environmental influences.

Learning through communication

Assuming that criminal behaviour can be learned through communication, we can, therefore, induce that learning takes place in underground forums. These forums can be viewed as social networks [53]. Where entities know each other through their pseudonyms, as they communicate through public threads or via private messages.

Previous work has been focused on the social aspect of organised crime. However, a few of them have suggested graph-based algorithms and techniques for intelligence gathering. Graph theory and graph-based algorithms have been widely used in digital forensics for network forensic analysis [54] and malware detection and classification [55]. The growth of cybercrime put challenges the ability for investigators to apply the process of digital forensics to obtain timely results. Irons and Lallie [56] discussed the need for intelligent techniques to address the challenges of the broad and complex cyber domain, where they suggest network analysis techniques.

Sparrow [37] discussed the application of SNA techniques as a new intelligence gathering technique. SNA is the study of relations, ties, patterns of communication, and behavioural performance within a social network [38, 57]. It uses graphs to model the entities and the relationship between them.

To the authors knowledge, it has not been studied how SNA can be applied for intelligence gathering about an underground forum. However, the information that is of interest to the forensic examiner are among the following [58]:

- Key actors
- Entity relationships in the network at specific times
- Changes over time (e.g. pre- and post-criminal activity)

Social network analysis of the Enron dataset

Due to the difficulty of acquiring datasets containing underground forums, we substitute it with the Enron dataset. This dataset, discussed in Section 4.1.2, is an e-mail corpus containing the e-mail messages for many employees of Enron. The value of SNA in investigations has been demonstrated on research conducted on the Enron corpus. Disener and Carley [59] examined the structural properties of the Enron network to identify key players. They showed that, in general, people with higher positions was more likely to be key players in the organisation. Their research also indicated that people with higher positions sent more e-mails than they received before the Enron crisis. Communication was more diverse concerning formal positions during the crisis than during regular months.

Researchers have been able to [58]:

- Discover key players [57]
- Discover hidden group
- Discovering organisational structures
- Demonstrate how communication networks change during an emerging situation

Social network analysis in digital forensic

Abraham et al. [57] used centrality measures for discovering influential individuals. These measures tries to describe an actor's relative position within his or hers network. Sparrow discussed that betweenness centrality could be a measure of significance within communication networks [37]. Methods and ideas for SNA has been applied to the analysis of physical OCGs [51, 53]. However, none of them have applied it to the new challenge of online OCGs.

It is more common for investigators to collect social network data for specific suspects, since they are already under investigation. However, we focus on methods for gathering intelligence of previously unknown (social) networks. The underground forums used by cybercriminals can be viewed as a social networks [53]. They frequently only know each other online via pseudonyms, as they communicate with each other in public threads or via private messages.

The research has been focused on how these underground forums operate. Motoyama et al. [53] characterised different underground forums based on the social networks they formed and how individuals gain and lose trust. They found that forum members gains trust by participating in traiding and discussion threads. However, the forums they inspected did not appear as "organised", as there were few people that issued a friend request to other users. The goal for these forums is to expand the knowledge based of their members, through learning in interactions with other members.

Modeling the interconnections of social networks has recently attracted attention in computer science [51]. A social network can be represented by graph-based structures, where users are represented by vertices and the relationships between users are represented by edges.

IBM i2 Analyst's Notebook

Some popular tools such as EnCase, FTK, The Sleuth Kit, Redeye and Volatility is regularly found in a digital forensic investigator's toolbox. Also, there is the IBM i2 Analyst's Notebook [60]. It is a visualisation tool that tries to optimise the value of big data. It allows analysts to collate, analyse and visualise data from disparate sources while reducing the time required to discover key information in complex data [60].

The author got a demonstration of Analyst's Notebook at a law enforcement agency, where they used it for visualisation and analysis. Analyst's Notebook treat each entity (vertice) and link (edge) as the same objects, with some small variation in the attributes they use. However, its power comes from the ability that these objects can be of anything that has some relationship of some kind. Since around three or four years, this software has come with built-in algorithms for measures of centrality found in the research area of SNA. These algorithms are *degree*, *betweenness*, *closeness* and *eigenvector*. They are discussed in Section 2.4.1. Also, the software also has the *k-core* algorithm, which is a maximum group of actors who are connected to k other members of the group.

Graph construction

Graph construction algorithms are popular in social media and the area of label propagation problems [43], such as the automatically labelling of photos or videos. The problem consists of labelling large amounts of images based on a few images with labels. Graphs are often employed as an effective representation for label propagation [61]. The images are expressed as vertices and edges reflect similarity between them.

It has to the author's knowledge not been studied how graph construction algorithms can be used for digital forensics, more specifically in the area of network analysis.

4 Choice of methods

In the previous chapters, an introduction to the problem, theory on central topics, and the current state of the art have been presented. The following chapter discusses the methodology applied to answer the defined research questions. This includes a selection of different methods, a justification for this selection, as well as the expected results. This chapter clearly states how the activities are performed, ensuring repeatability for future researchers.

4.1 Dataset

Relational data can be collected or randomly generated in numerous ways. Collected data is usually obtained from archival records to recover information such as transactions (for the study of economic transactions), or using questionnaires or interviews to get relations between friends (for the study of social relationships). The nature of the study determines the methods of data collection, the size of the population and the appropriate analytic methods [62].

The concern of which actors to include in the study determines the population (boundary) of interest. Sometimes it is impossible to use analytic measurements on the entire population as the population becomes large. In such situations, a *sample* of the entire population may be taken from the set. The population issue is relatively easy to deal with in a closed set of actors, such as in all employees in a small business or faculty in an academic department. For other studies, the boundary of the set of actors may be difficult (if not impossible) to determine [62].

Social networks form different structures depending on how people interact. For example, Twitter has a directional follower-followee relationship since the follower does not have to reciprocate the followee, whereas Facebook can be viewed as undirected as they require a confirmation of friendship requests. Therefore, it is important for the dataset to be representative of the environment under investigation.

Because of the difficult nature of getting hold of a real dataset over organised criminal groups, the available options for this thesis is to generate a dataset randomly or to find a pre-existing dataset.

4.1.1 Randomly generated datasets

Initially, we had the option of generating a random graph to construct a network consisting of actors operating in the CaaS business model. In the research of Newman et al. [63] they constructed random graphs and compared them to real-world network data.

Their research showed that in some cases, the constructed graph was in agreement with real-world networks. In other graphs, the agreement was poorer. This suggests that there was "some presence of additional social structures in the network that was not captured by the random graph, or social forces at work that shape the network" [63].

Since randomly generated graphs sometimes have a poor representation of real-world network data, we are more interested in networks that already contain the social forces that shape it.

4.1.2 Enron-email corpus dataset

Due to privacy and legal restrictions, it is very tough to get hold of large and realistic datasets. The exception is the Enron corpus dataset, as it contains e-mails generated by 158 senior executives of the Enron Corporation [64], which was acquired by the Federal Energy Regulatory Commission (FERC) during the criminal investigation of Enron Corporation.

After the investigation, FERC made the decision to publish the Enron e-mails from senior executives from 1998 through 2002. The version FERC had posted the e-mails in an unusable format, Leslie Kaelbling (at Massachusetts Institute of Technology), purchased the raw files and spent the time to clean up the data – removing duplicates, organising folders and taking out the remaining private attachments and e-mails [65].

The original dataset with 517 431 e-mails have gone through some numerous revisions to clean and structure the data, and it was eventually reduced down to 252 759 by 2004. It is available at `http://www.cs.cmu.edu/~enron/`. Professor William W. Cohen posted the dataset result from 2004 at Carnegie Mellon University (CMU) [66]. This dataset of 2004 by Cohen is widely accepted by many researchers [65]. This version excludes attachments, and some messages have been deleted upon the request of Enron employees. The Enron dataset has been applied to get a better understanding of how language is used and how social networks function, and it has improved efforts to uncover social groups based on e-mail communication [59].

Figure 39: Enron logo

The original dataset is distributed into 3500 subfolders and is, therefore, difficult to use in computational tasks. We will, therefore, need it to be in a more computational-friendly format, as in a Database Management System (DBMS). Shetty and Adibi [67] from the University of Southern California processed the Enron corpus in 2004 and released a MySQL 4.0 dump version. We will be using their MySQL database, with 252 759 e-mails generated from 151 employees. The

MySQL dump is available from `https://www.cs.purdue.edu/homes/jpfeiff/enron.html` [68].

Enron database schema

Figure 40 shows the schema of the database that shows it contains four tables.

`employeelist` provides information about all of the 151 employees, including full name and e-mail address.

`message` contains information about all e-mail messages, including the sender, subject, date and time sent, message content and other information.

`recipientinfo` contains information about which e-mail message was sent, the e-mail address of the recipient and the type (TO, CC, BCC).

`referenceinfo` contains information about all those messages that have been referenced after being sent once, either as a forward or reply [67].

The syntax from MySQL v4.0 and v5.7 has changed, generating an error when trying to import their MySQL dump. More specifically, the cause of this error is a keyword in "`TYPE=MySIAM`" from v4.0. By changing this line to "`ENGINE=MySIAM`" makes it compatible with v5.7 and solve the importing error.

Database engine `MySIAM` do not support foreign keys, so the arrows in Figure 40 illustrates where this field can be found in other database tables. The database engine should be converted to `InnoDB` to support foreign keys, but this is not necessary for this thesis.

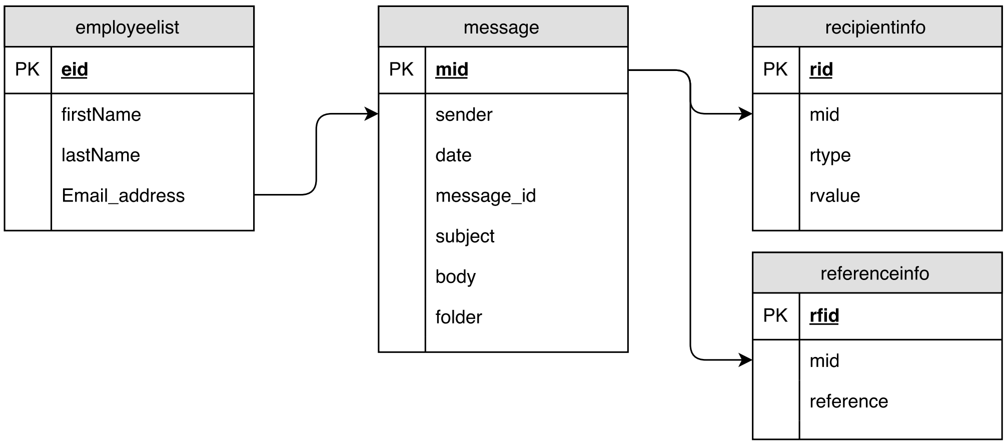

Figure 40: Enron database schema

Enron corpus validity

As previously mentioned, the Enron corpus has gone through numerous revisions to clean the database of duplicate messages, e-mail addresses, removal of personal information and by request from previous Enron employees [64, 65]. These revisions could have resulted in leaving the corpus in an invalid state. However, with the corpus being widely studied and highly accepted by many researchers, it is evaluated to low risk that the corpus is in an invalid state.

Removal of employees and their messages from table “employeelist” and “message” is seen as a reduction of the population boundary for the SNA. Moreover, therefore, not a risk of producing incorrect results in the analysis.

The removal of messages between employees would affect the resulting graph. The edge weights get distorted and will not show the correct frequency of sent and received messages. It is hard to verify as the original dataset, with over 3500 subfolders, as it is very tough to process. To the best of our knowledge, the Enron corpus is valid and will produce a result that reflects the reality within Enron corporation.

4.2 Pre-processing

To explore and understand how factors (such as relationships on an individual and group level) might have impacted the network, we need to extract and analyse this information in an effective and efficient way.

The MySQL database is one step in the direction of a structure that is easier to analyse. However, this format contains unnecessary information such as e-mail message text, and additional processing is required to extract the number of messages between Enron employees.

We first wanted to go through each e-mail in the “message” table and extract to and from e-mail addresses and the frequency. We then had to separate e-mail addresses ending with “@enron.com”, since the table also contain other e-mail addresses for people outside of the Enron corporation. As they are outside of the Enron coorporation, they are also outside of the population of interest for this research. After processing around 26% of the e-mails the program had generated a graph with over 5000 vertices after 6 hours.

Since this creates a graph with too large of a population, we had to select a sample to represent Enron. We choose the 151 employees listed in the table “employeelist” to represent our sample. Selecting this sample makes it easier to calculate the centrality measures and to find their position (job title) within the company. Including the employees position makes it easier to evaluate the results from the SNA.

It is assumed that talented actors in a network tend to have more central roles for the communication. It can be compared to the CaaS business model with criminals of different positions and expertise. Criminals with more desirable expertise will have more attention whereas criminals with less expertise will get less attention.

4.3 Experimental design

This section discusses how this thesis will answer each of the research questions.

4.3.1 Features for identification of individuals and groups

SNA centrality measures are the best indication of how important an actor is. Other levels of analysis (see section 2.4.1) is first interesting to analyse when you only have small groups to analyse (such as k-cliques). The challenge lies in fully understanding the different measures and knowing the context of how to interpret them. This thesis will examine the various centrality measures on the Enron dataset. The goal is to evaluate each measure for their ability to list important actors. The reason for this target is to quickly prioritise a digital forensics investigation on those actors and to continue analysis on those actors with different levels of analysis.

Limitations

The limitation of this analysis is that the e-mail messages was captured over a four-year period. So the results can be skewed towards the amount of time they were active. People that got inactive for some reason (sick leave, pregnancy, changed positions, left the company, and so forth) might affect the result of the analysis. One additional restriction is that the script to extract SQL data only looks at sending e-mails, whereas e-mail communication is more dynamic than one sender and one reciever. Finally, the dataset might poorly representat a real underground forum, so it can only be used as a proof-of-concept.

Results

The results of this analysis will directly impact the next section as it will limit the available features to construct a graph. Features that take the other actors into consideration will be selected for the graph construction. This process of selecting the best features will hopefully aid in the graph construction to understand similarities between actors. It is important to understand the different available features before continuing with other experiments.

4.3.2 Graph-based construction techniques for digital forensics

The goal of this research question is to find whether graph construction algorithms can help in a digital forensic investigation. To achieve this aim, a set of algorithms is implemented (see section 2.4.2). The goal is to see whether changes in graph construction algorithms produces a reliable graphical representation that could be used in an investigation. The algorithms are selected to evaluate how they can be applied to digital forensic investigations rather than finding an algorithm that performs the best.

Limitations

Due to limited time and complexity of particular algorithms, we will be unable to test all the algorithms described in section 2.4.2. Because of this, we limit our experiments to neighbourhood construction algorithms. The reason we do not implement other algorithms is that we could not find previously implemented algorithms, and it takes time to implement each algorithm and to making sure that it work as intended. Implementing more algorithms would also increase the time needed to understand, compare and analyse each result. The specific graph construction algorithms that are tested in our experiments is the following: e-neighborhood and k-NN.

Results

The results of this experiments are not necessarily complex to analyse, but to understand. The reason for this statement is that human interaction is governed by hidden and complex social forces or behaviours. These social forces impact the algorithms used for building the graphs. The goal is to understand how (some) graph construction algorithms can help in a digital forensic investigation. It is expected that these algorithms can assist in the fast identification and prioritisation of important actors in a network. e-neighborhood is expected to produce the least reliable results, as it is heavily dependent on the selected distance (e). k-NN is supposed to produce a more reliable result. However, it can produce irregular graphs.

5 Experiments

In the previous chapters, we have discussed the motivation and contribution of the thesis, theory on central topics, and the current state of the art in topics related to our research questions. Further, we presented our methodologies for solving the research questions. In the following chapter, we will give a description of the experimental environment we used, to provide a guideline for future research and repeatability. We also provide a description of the experiment process and the expected results, in addition to discussing the software implementations responsible for executing the experiments. Further, we review the results related to each research question, and a summary is provided at the end to list the findings.

5.1 Experimental design and environment setup

To adequately represent and analyse the information contained in the corpus we need to extract the necessary data from the database. The data of interest are the Enron employees as vertices and the e-mail exchange as edges. The relation among and between the actors in this dataset are reflected in the e-mail exchanged between employees and the content of those messages. In this thesis, we concentrate on the extraction and explorative analysis of exchanged e-mails.

From a sociological perspective, the relational data in the corpus is considered multi-mode and multi-time. In other words, it contains multiple modes (relations) such as friendship, mentor/mentee and colleague to name a few. It also contains relations over an extended period of time, where these relations could change.

We need a data format that can handle rich network data to represent adequately and analyse the information provided within the corpus. That also can be used as input to multiple analysis tools that we consider using. We chose to use Graph Exchange XML Format (GEXF) as the data format because it meets our requirements. GEXF is a file format similar to Extensible Markup Language (XML) that can represent an arbitrary number of vertex sets and graphs. Also, to represent our required vertices and edges, it can also represent their attributes such as node label, position, colour and edge weight. Listing 5.1 provide an example of the GEXF file format.

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
  <graph mode="static" defaultedgetype="directed">
    <nodes>
      <node id="0" label="Hello" />
      <node id="1" label="Word" />
    </nodes>
    <edges>
      <edge id="0" source="0" target="1" />
    </edges>
    </graph>
</gexf>
```

Listing 5.1: Hello world GEXF example

The advantage of using the GEXF format over other data formats is that many tools, such as Networkx and Gephi, already support this format. Because of its similarity to the XML data format, it can easily be changed to any other required format with an XML parser. Also, the Python package Networkx can read and write this file format, including numerous other output formats. GEXF files for the representation of communication networks require data from three tables in the database: *employeelist*, *message* and *recipientinfo*.

All database work and data extraction were performed on a MacBook Pro machine, with Python scripts that we wrote for this purpose. In addition to the laptop computer, we also had a desktop computer available. However, it was not necessary as the laptop had the necessary processor and memory capacity. Table 4 provide a list of the equipment and Table 5 provide a list of the software required for these experiments. Python was used to extract the Enron dataset, run the SNA and graph construction.

	MacBook Pro (Retina, mid 2012)	Desktop computer
Processor	2,3 GHz Intel Core i7	3,41 GHz Intel Core i7-6700
Memory	8 GB 1600 MHz DDR3	32 GB 1300 MHz DDR4
Storage	250 GB SSD Flash Storage	500 GB SSD Flash Storage
Operating system	OS X 10.11.5	Windows 10 and Ubuntu 15.10

Table 4: Experiment equipment

The Python script `sqlexp.py` (seen in Appendix E) was created to export the necessary information from the dataset. At the start of the script, it generates a list of all the Enron employees found in the employeelist table. It uses their e-mail addresses to find messages (from table message) that have been sent from the employee e-mail address to other employees in the same dataset sample. E-mail correspondence has three distinct types: *TO* (direct to the recipient), *CC* (copy to a recipient) and *BCC* (blind copy to a recipient). We are interested in each e-mail type since it is a direct action taken by the employee. Therefore, we do not discriminate on the type of e-mail correspondence. An edge is added to each recipient of the e-mail message, where the weight is the number of e-mail messages between the same sender and receiver.

This script also generates the vertex colours use throughout this thesis, described in Section 5.3.1. The colouring is based on each employee's job position within the hierarchy of Enron, seen in Figure 42. The output from `sqlexp.py` (Appendix E) is one graph and one digraph, which will be used for SNA. Be aware that this output also includes self-loops (messages sent to oneself), but these edges are discarded before the analysis.

Module	**Version**
Python	2.7.10
Scipy	0.16.0
Numpy	1.9.2
Networkx	1.11
MySQLdb	1.2.5
gexf	0.2.2
lxml	3.5.0
libxml2	2.7.0

Table 5: Python modules requirements

5.1.1 Dataset

Figure 41 show the adjacency matrix over the dataset. It was created with Protovis[1], where a blue square indicates an edge between two actors and a white square indicates the lack of an edge. Protovis also organises the matrix according to some clustering algorithm. The usage of this algorithm and its parameters could not be changed by the researcher, but the figure correctly orders the cells according to the employee IDs.

Figure 41 indicates that the network is sparse. More actors increase the chance for the network to be sparse, but in this case, it can also be because of the extended period for when the e-mail messages were collected (little over four years). Also, some employees could have quit, or been replaced by other employees, and therefore no edge was created between those actors. It can also be because of the fact that corporations are organised, and people will usually have contact with others within the same department.

[1]`http://mbostock.github.io/protovis/`

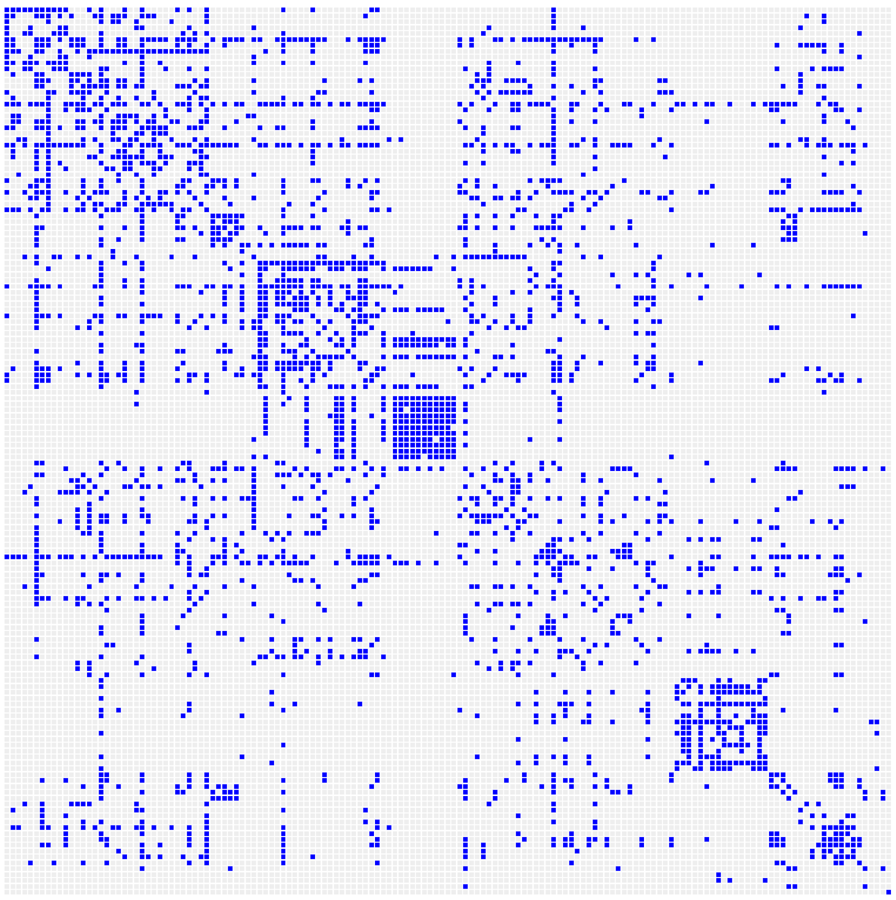

Figure 41: Enron undirected adjacency marix (tool: Protovis)

Results from script `sqlexp.py` (Appendix E):

- File *undirected_colour.gexf* with 151 vertices and 1612 edges
- File *directed_colour.gexf* with 151 vertices and 2235 edges

The naming scheme for these files reflects the direction of the graph, and "colour" refers to the colour scheme for the Enron job title hierarchy in Figure 42.

5.1.2 Dataset validation

Validating the datasets was manually done by randomly selecting a few Enron employees, and check that their count of sent e-mails correlated with the edge weight in the dataset. This validation process was done through querying the database through the MySQL shell. The process of validating the digraph was to insert the sender's employee ID on line three and the receiver's ID on line seven into the query in Listing 5.2. For this example, the sender is ID 9, and the receiver is ID 67. So a valid digraph will have an edge from 9 to 67 with weight 310 whereas a valid graph will add to this weight with the edge weight from 67 to 9 with weight 386, so a total weight of 696.

The script would have generated a valid directed dataset if the result of this query was equal to the edge weight that corresponds to the sender and receiver IDs. For undirected dataset, the query has to be run two times, and the results added. One time with the sender's ID on line three and receiver's ID on line seven, and vice versa for the second time.

```
1   SELECT COUNT(rvalue) FROM recipientinfo WHERE mid IN (
2     SELECT mid FROM message WHERE sender LIKE (
3       SELECT Email_id FROM employeelist WHERE eid LIKE 9
4     )
5   )
6   AND rvalue IN (
7     SELECT Email_id FROM employeelist WHERE eid LIKE 67
8   );
9
```

Listing 5.2: Query to validate dataset

Before using the datasets generated from the `sqlexp.py` for SNA, self-loops can be removed without affecting the results. As one's relationship to themselves are not interesting. The affected vertices with self-loops are:

2	3	6	7	11	13	14	15	16	17	18	20	23
26	27	28	29	30	31	32	33	35	36	37	38	39
40	42	44	45	48	49	51	52	53	55	57	58	59
61	65	66	67	68	69	70	71	72	73	75	76	78
79	80	84	91	92	97	98	100	101	103	105	107	109
111	112	113	114	116	120	122	123	125	126	127	128	129
131	135	137	139	140	142	149	150					

Table 6: Enron employees with self-loops

Also, some centrality measures require the graph to be connected (discussed in 2.3.2). A graph is connected when there is an edge between every pair of vertices, so there is no unreachable vertex. Employee 116 is the only one that is isolate from the others in both the graph and digraph. Since she is both isolate and has self-loops, she is removed from the dataset before SNA.

5.2 Experiment execution

This section discusses the Python scripts used to execute the experiments. First, the SNA is performed and analysed with the Python package Networkx. Then the feature extraction and graph construction are analysed. Experiment results are first discussed in Section 5.3.

5.2.1 Features for identification of individuals and groups

The first research question is *"which features can be used to identify important and influential individuals within a network?"* The goal is to use graph methods to separate important individuals from the rest of the group so that digital forensic investigators can focus their investigation on a subset of actors. SNA have been used in sociological studies to analyse social structures in a network. We want to apply them for analysis of organisational structures to find actors with higher positions within a network. From the discussion of SNA in background theory and previous work, it has been shown that SNA can be used to achieve this goal.

The Python script `sna.py` (found in Appendix F) utilises the Python package Networkx. It is a package for the creation, manipulation, and study of the structure, dynamtics, and functions of complex networks [69]. The selection of the programming language was because of the author's familiarity with it. Also, the Python language has many advanced and popular third-party packages such as SciPy, Numpy and Networkx. SciPy and Numpy will be discussed in the next subsection.

The Networkx package has implemented all of the algorithms for centrality measures (covered in 2.4.1) that are used in SNA. Also Networkx has several other centrality measures such as katz, load, current flow closeness, current flow betweenness and communicability betweenness. These new measures are also included in our SNA to analyse their effect of reaching our research goal. Table 7 lists the different measures analysed for both graph and digraph.

We selected the centrality measures that could measure an actors position within a network as features. Standard measures of centrality in SNA is degree, in-degree, out-degree, betweenness, closeness and eigenvector, and they are therefore included as default in the analysis. However, the new centrality measures need a justification for their inclusion. As each algorithm measures different qualities of each actor, this list gives a justification for including the measures:

Degree is the most simple and basic measure of centrality. It is the number of edges incident to the vertex and is not considered a powerful measure. Therefore, it can only be seen as a measure that emphasises an actor's activity in a network, in other words, their involvement in a network.

In-degree is the number of edges adjacent from the vertex. When edges are associated with some positive aspect such as friendship or collaboration,

Directed graph	**Undirected graph**
Degree	Degree
In-degree	
Out-degree	
Closeness	Closeness
Betweenness	Betweenness
Eigenvector	Eigenvector
Katz	Katz
Load	Load
	Current flow closeness
	Current flow betweenness
	Communicability betweenness

Table 7: Centrality measures analysed by `sna.py` (Appendix F)

in-degree is often interpreted as a form of popularity [38].

Out-degree is the number of edges adjacent to the vertex. When edges are associated with some positive aspect, out-degree is often interpreted as a form of outgoing or social person [38].

Betweenness quantifies the number of times a vertex acts as a bridge along the shortest path between two vertices. It emphasises the potential control over information flow in the network.

Closeness is the distance (shortest path) from all the other actors. Thus, the more central a vertex is, the lower its total distance from all other vertices.

Eigenvector is the measure of the influence of a vertex in a network. It measures an actor's involvement in a network based on how many edges their alters have. Eigenvector is often interpreted as a form of popularity; an actor is considered important when they also have many important friends.

Katz is a generalisation of the eigenvector centrality. Eigenvector works well if the (di)graph is (strongly) connected. It counts the number of total walks between actors, where each connection is given a weight based on the walk length. It measures an actor's involvement in a network based on the total amount walks between a pair of actors. This is an additional centrality measure that Networkx lists, and could potentially be better to achieve our goal.

Load is the fraction of all shortest paths that pass through that node. It is slightly different from betweenness. Networkx lists this as an additional centrality measure, and could potentially be better to achieve our goal. It can also be compared with betweenness centrality.

Current-flow closeness is a variant of closeness centrality based on effective resistance between nodes in a network [69]. It is also known as information centrality. Networkx lists this as an additional centrality measure, and its results can be compared with closeness centrality.

Current-flow betweenness is a variant of betweenness centrality which uses an electrical current model for information spreading, whereas betweenness centrality uses shortest paths [69]. Networkx lists this as a centrality measure, and its results can be compared with betweenness centrality.

Communicability betweenness uses the number of walks connecting every pair of vertices as the basis of a betweenness centrality measure [69]. Networkx lists this as an additional centrality measure, and its results can be compared with betweenness centrality.

The SNA script analyses both files resulted from the exportation script, as discussed in Section 5.1. The (di)graph is read from a file, and Networkx converts it into a dataformat that it can process. Note that this method also preserves the colours for the employee job titles.

The SNA process begins after removing isolates and vertices with self-loops. It analyses the dataset for each of the measures of centrality, by following the list provided in Table 7. Each centrality value is saved to a data structure that assigns it to the correct vertex index and holds the ordering of the values. This data structure is easier to write to a Comma-Separated Values (CSV) file.

However, the script will only display the 30 highest values to the terminal window. This number was selected based on these criteria:

- Our goal is to find features that can be used to identify important and influential individuals within a large group of people. It is most convenient to select a subset from the set of actors, which allow digital forensic investigators to focus on those actors. Therefore, selecting 30 actors to analyse allow us to evaluate if the feature is consistent with finding actors according to our goal.
- Some actors with lower organisational positions might occopy the highest values when the limit is set to a low number. Therefore, the centrality measure could be incorrectly evaluated as worse than other measures.
- Setting the limit to a higher number will go against our goal of finding a subset of focal actors, as it provides a longer list of actors to investigate. It will, therefore, occupy more of an investigator precious time.

The output from the `sna.py` script (Appendix F) is two CSV files for one graph and one digraph dataset; the first is called *undirected_features.csv* and the latter is called called *directed_features.csv*. We chose to use CSV files as they provide the necessary order for vertices and their features. Each vertex is therefore represented with one row in the CSV file. Also, each row can also be seen as a feature vector (discussed in 2.2.2) for each vertex. These files will be used as input for the graph construction algorithms.

We discussed in Section 2.2.2 about Feature quality measures that it was necessary to evaluate each feature because not all features provide equal quality to the problem at hand. By selecting a few different measures of centrality, for two types of graphs, it can increase the likelihood of finding features that best captures the most important actors in a network. From previous research, we expect that betweenness centrality best captures them [38]. Because of the similarities of centrality measures for each graph type, we also assume that their ability to identify important actors will be quite similar between the types of graphs.

5.2.2 Graph-based construction techniques for digital forensics

The second research question is "how can graph construction techniques be applied in digital forensics for identifying targets?" The goal is to use graph construction techniques to aid an investigator through the target identification process. Several types of graph construction techniques have been applied to the problem of label propegation [43], but with the limited time, we are going to focus on the neighbourhood approaches: k-NN and e-neighbourhood. Each of these technqiues is implemented in the Python script *neighborhood_approaches.py* (seen in Appendix G).

The "no free lunch" theorem (discussed in Section 2.2.4) states that there is no technique or function that outperform the other techniques or functions for all tasks. Therefore, we are implementing two graph construction techniques and using several distance measures to compare and evaluate them to each other. The plan was to also include the b-matching algorithm. However, it proved challenging as it takes into account more constrains than the two other techniques. The implementations for each algorithm had to be tested to be verified that they worked as described in Section 2.4.2.

The script uses the Python packages SciPy and Numpy. Numpy[2] contains a powerful N-dimensional array that is used to contain the feature vectors. Moreover, SciPy[3] is a popular package that contains scientific computing tools for Python; it includes several functions to calculate the distance between two feature vectors.

For this experiment, we are interested in functions that can compute the distance for numerical feature vectors (no-boolean vectors). The functions had to not contain additional parameters beyond the feature vectors, as we are not so familiar with the Enron dataset that we can customise the parameters to optimise

[2]http://www.numpy.org
[3]http://www.scipy.org/

the results. More specifically, we selected these eight distance functions: *braycurtis*, *canberra*, *chebyshev*, *cityblock* (Manhattan), *correlation*, *cosine*, *euclidean* and *hamming*.

The script begins by parsing the two files, *directed_features.csv* and *undirected_features.csv*, with feature vectors generated from the SNA script discussed in the previous section. These feature vectors are saved in a N-dimensional Numpy array, where N correspond to the number of features in Table 7. It also saves the vertex colours used in this thesis, to be consistent of the colour coding which follows the Enron hierarchy in Figure 42.

Then, for each distance function, it generates a graph based on the graph construction algorithm. For e-neighbourhood method, there is an edge between two vertices when their feature vector distance is less than a specifided value e. Smaller values for e is more preferable, as it will be easier to connect similar actors. Whereas with higher values for e, actors will be connected to more actors and thus be more difficult to analyse. We select and evaluate the values 0.01, 0.02, 0.03, 0.04, 0.05 and 0.06 for e.

For k-NN method, there is en edge between one vertex and its k most similar actors for a specified value k. Again, smaller values for k is more preferable, as it does not connect as many actors together. k-NN method guarantees that each vertex have at least k out-bound edges, however, each vertex is not guaranteed to have k in-bound edges. The in-bound edges are most likely in a range equal to or higher than zero. We select and evaluated the values 1, 2 and 3 for k.

For each technique, the verification process was to randomly select a vertex u. We calculated the distance between u and all the other vertices and put these values into a list. Then, we sorted this list in ascending order, with smaller values first of the list. For either e-neighbourhood or k-NN graph to be constructed sucessfully, the vertex u has to be connected to the appropriate number of closest (similar) actors. For e-neighbourhood this would be all actors below a threshold e, and for k-NN this would be its closest k neighbours. We compared the distance lists for vertex u with the appropriate neighbourhood approach and distance function. After trying several randomly selected vertices we could not find one inconsistent constructed graph.

The output from this Python script is several GEXF files with the filename-pattern: *graph direction_value_construction method_distance function.gexf*. Where "graph direction" refers to the two graph types graph and digraph, "value" to the total nine values for e and k, "construction method" for the two methods e-neighbourhood and k-NN, and "distance function" to the eight distance functions. This results in $2 * 9 * 8 = 144$ graph and 144 digraphs used for analysis.

The script allows for both binary and weighted output of graph edges. We expect to see some similarities in graph and digraphs, with the digraph performing better since it contain more information in its edges. However, we are going to focus on the binary graph output for our analysis.

Since this script produces similarity graphs, where similar feature vectors for actors are connected to other similar actors, we expect to find some pattern in the resulting graph. We think that this pattern will allow us to prioritise actors in a better way than the lists generated by the SNA. Because it takes into account all of the features, which is the centrality measures. There should be little variation in the selection of distance measure as they all find the similarity between 1-dimensional feature vectors. However, a distance measure that divides or segregates the actors according to their centrality or that best prioritises the actors will fit our goal the best.

5.3 Experiment results

Before conducting the experiments, as outlined in Chapter 4, we can import both the graph and digraph into Gephi[4]. It is an open-source software program for visualisation and exploration tool for graphs. The graph has 151 vertices and 1612 edges, and the digraph has 151 vertices and 2235 edges. The illustration for both of these graphs is presented in Appendix B.

The experiment results for each research question is presented in this section. First, it present the results from the SNA, where the focus is on the centrality measures to prioritise actors before a digital forensic investigation. An investigator will be more efficient with their time by focusing on a few focal actors, rather than trying to study the complete graph. Then the results from the various graph construction algorithms are presented. Where we focus on how these algorithms can be used with features extracted from the SNA process.

5.3.1 Results of Social Network Analysis

In this subsection we provide an overview of the results from SNA on the data subset, which consists of 151 Enron employees from the *employeelist* table. The result is from SNA for both graph and digraph, and a description of generating these graphs was provided in Section 5.1. This section is presenting the summary from the SNA result, but the results is presented in their entirety in Appendix C.

To evaulate how centrality measures can help in the identification of interesting or important persons, we will also need their job title (or position) within the company. The complete list of Enron employees, including their job titles, used for this experiment can be viewed in Appendix A. A description of the process for gathering these job titles is found in the same appendix. Table 8 show a summary of the Enron employee job titles, where 138 employees had their job positions listed, and only 13 was not included in the list. These 13 employees will be treated as Not Available (N/A) during the analysis.

[4]`https://gephi.org`

43 employees (28,48% of 151) with job titles in the Enron dataset have what we regard as a high position within the company. We assume that higher positions are considered to be more important or influential within the network than employees with lower positions. A summary of the Enron employee list is found in Table 8. Whereas the lower positions represent the majority of the dataset with 108 employees (71,52% of 151). Where 13 of them have unknown job titles.

Position	**# Employees**	**Percent**
CEO	4	2,90%
President	4	2,90%
Vice President	21	15,22%
Director	12	8,70%
Managing Director	2	1,45%
In-House Lawyer	1	0,72%
Manager	12	8,70%
Trader	11	7,97%
Specialist	0	0.00%
Analyst	0	0.00%
Employee	35	25,36%
N/A	36	26,08%
Total	138	100.00%
Empty	13	
Total	151	

Table 8: Number of individuals per job title

It is to be expected that lower job positions is more represented in the dataset, as they fulfill more of the jobs of an organisation. Therefore, the centrality measures should find active and important actors with higher positions. The distinction between high and low positions is based on the Enron hierarchy present in the work from Gilbert [70]. Figure 42 illustrate colour coding for the different job titles. The colours in this figure is used throughout this thesis, including the generation of graphs.

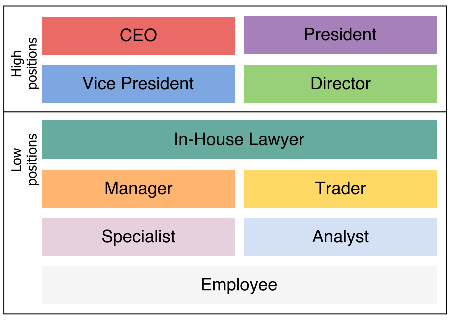

Figure 42: Enron hierarchy of job titles [70]

The identification of high positions is preferable over low positions. Also, within high positions, the identification of CEO and president is preferable over vice president and director (or managing director).

We chose to inspect the top 30 highest ranking actors for each measure of centrality. A summary of the top 30 graph features are found in Table 9, and the full results can be found in Appendix C, in Tables 20, 21, 22, 23 and 24.

The summary in Table 9 show the SNA results for a graph. The first column shows the centrality measures selected for the graph (discussed in Section 5.2.1). The columns from two to six show the amount of high positions that was found

Centrality measure	C	P	VP	D	MD	Total	% top 30	% 43 HP
Degree	2	3	5	1	1	12	40.00	27.906976744
Closeness	2	3	8	1	1	15	50.00	34.88372093
Betweenness	2	2	6	0	1	11	36.67	25.581395349
Eigenvector	0	1	7	0	1	9	30.00	20.930232558
Katz	2	2	2	1	0	7	23.33	16.279069767
Load	2	2	7	0	1	12	40.00	27.906976744
Current flow closeness	0	2	7	0	0	9	30.00	20.930232558
Current flow betweenness	1	1	8	0	0	10	33.33	23.255813953
Communicability betweenness	3	2	10	3	1	19	63.33	44.186046512

Table 9: High level positions found by centrality measures for undirected graph

for that measure, where the job titles have been reduced to initials. Where “C” stands for CEO, “P” for president, “VP” for vice president, “D” for director and “MD” for managing director. Column seven show the total amount of high positions actors found by the centrality measure, with column eight displaying the percentage of the top 30 list that holds high positions. At last, coloumn nine (*43 HP*, where HP is “high position”) hold the percentage of how many of those 43 we consider high position that are within those top 30 actors.

Centrality measure	C	P	VP	D	MD	Total	% top 30	% 43 HP
Degree	3	3	10	0	1	17	56.67	39.534883721
In-degree	1	3	11	2	1	18	60.00	41.860465116
Out-degree	3	2	8	1	0	14	46.67	32.558139535
Closeness	3	3	11	0	1	18	60.00	41.860465116
Betweenness	1	3	8	1	0	13	43.33	30.23255814
Eigenvector	0	2	8	0	1	11	36.67	25.581395349
Katz	1	1	1	6	0	9	30.00	20.930232558
Load	1	3	9	0	0	13	43.33	30.23255814

Table 10: High level positions found by centrality measures for directed graph

Table 10 presents the SNA results for a directed gprah. The description of columns from the previous table is also applied to this table. All results of digraph SNA analysis is found in Appendix C, in Tables 25, 26, 27 and 28.

5.3.2 Results of graph-based construction techniques

In this subsection, we provide an overview of the results from graph construction. The result has been 144 graphs and 144 digraphs, and a description of generating these graphs was provided in Section 5.2.2. This section is presenting the summary of the graph construction neighbourhood approaches, but the result is presented in its entirety in Appendix D.

In the previous sections, the relationship between vertices had always been some communication between them. However, for graph construction approaches this is no longer the case. The relationship between vertices now represents the similarity between their feature vectors. This similarity is calculated by several distance functions, as listed in Section 5.2.2. Table 11 show the minimum and maximum distance over all feature vectors.

An initial evaluation of these values indicates that some minium values has

	Graph		**Digraph**	
Distance measure	**Min**	**Max**	**Min**	**Max**
Braycurtis	0.0135893026624	0.813441493401	0.00831285077795	1.50270295281
Canberra	0.518823901069	9.27871034156	0.436380271907	7.93760289983
Chebyshev	0.013422818792	0.782908318632	0.006711409396	0.900192949238
Cityblock	0.0350083484866	2.91255273106	0.00949945718664	2.50614556103
Correlation	0.000112208245271	0.599616961817	2.51215982707e-05	1.93466245485
Cosine	7.08791772255e-05	0.480578481897	1.87554821347e-05	1.89933105094
Euclidean	0.0169881059058	1.20179754992	0.00692017593907	1.09143782882
Hamming	0.5	1.0	0.5	1.0

Table 11: Distance measures minimum and maximum values

some value higher than our smallest e distance, which is $e = 0.01$. We expect there to be an edge between some arbitrary vertices u and v when the distance is less than e. Therefore, the minimum values suggest that the distance functions Correlation and Cosine will start to produce a similarity graph the beginning, where $e = 0.01$. The other distance measures, Braycurtis, Chebyshev and Euclidean will start producing similarity graphs when $e = 0.02$. Whereas Canberra and Hamming will never produce similarity graphs as their minimum distance is 0.5 and above.

The same initial evaluation can also be done for the digraph. Where Correlation and Cosine will start by producing the similiarity digraphs, and the other distance measures will soon follow. Except for Canberra and Hamming as they have a minimum distance of 0.5 and above.

Tables 14, 15, 14, 15 and 16 show the summary of the different distance functions for both e-neighbourhood graph and digraph. These tables confirm our initial evaluation of how the distance measures could behave.

Graph									
	0.01-neighbourhood			0.02-neighbourhood			0.03-neighbourhood		
Distance	**Nodes**	**Edges**	**Weak**	**Nodes**	**Edges**	**Weak**	**Nodes**	**Edges**	**Weak**
Braycurtis	0	0	0	5	3	2	35	20	16
Canberra	0	0	0	0	0	0	0	0	0
Chebyshev	0	0	0	19	11	9	54	41	20
Cityblock	0	0	0	0	0	0	0	0	0
Correlation	140	758	5	149	1883	5	149	2830	3
Cosine	146	999	4	149	2321	3	149	3496	2
Euclidean	0	0	0	5	3	2	21	12	10
Hamming	0	0	0	0	0	0	0	0	0

Table 12: Graph construction summary e-neighbourhood part 1

Graph									
	0.04-neighbourhood			0.05-neighbourhood			0.06-neighbourhood		
Distance	**Nodes**	**Edges**	**Weak**	**Nodes**	**Edges**	**Weak**	**Nodes**	**Edges**	**Weak**
Braycurtis	68	53	21	95	125	16	121	245	13
Canberra	0	0	0	0	0	0	0	0	0
Chebyshev	94	119	16	118	242	14	126	387	8
Cityblock	5	3	2	7	4	3	15	8	7
Correlation	149	3683	2	150	4493	1	150	5137	1
Cosine	149	4518	1	150	5374	1	150	6089	1
Euclidean	38	25	15	78	61	24	104	138	19
Hamming	0	0	0	0	0	0	0	0	0

Table 13: Graph construction summary e-neighbourhood part 2

Digraph								
	0.01-neighbourhood				0.02-neighbourhood			
Distance	**Nodes**	**Edges**	**Weak**	**Strong**	**Nodes**	**Edges**	**Weak**	**Strong**
Braycurtis	6	6	3	3	20	22	9	9
Canberra	0	0	0	0	0	0	0	0
Chebyshev	16	18	7	7	66	146	13	13
Cityblock	2	2	1	1	8	8	4	4
Correlation	134	3406	5	5	147	6022	6	6
Cosine	143	4218	7	7	145	7038	3	3
Euclidean	4	4	2	2	30	40	12	12
Hamming	0	0	0	0	0	0	0	0

Table 14: Digraph construction summary e-neighbourhood part 1

Digraph								
	0.03-neighbourhood				0.04-neighbourhood			
Distance	**Nodes**	**Edges**	**Weak**	**Strong**	**Nodes**	**Edges**	**Weak**	**Strong**
Braycurtis	42	64	15	15	74	192	13	13
Canberra	0	0	0	0	0	0	0	0
Chebyshev	104	616	5	5	116	1210	4	4
Cityblock	15	16	7	7	34	46	13	13
Correlation	147	8102	4	4	148	9782	4	4
Cosine	148	9368	4	4	149	11048	4	4
Euclidean	77	196	12	12	95	512	4	4
Hamming	0	0	0	0	0	0	0	0

Table 15: Digraph construction summary e-neighbourhood part 2

Digraph								
	0.05-neighbourhood				0.06-neighbourhood			
Distance	**Nodes**	**Edges**	**Weak**	**Strong**	**Nodes**	**Edges**	**Weak**	**Strong**
Braycurtis	97	428	6	6	115	754	7	7
Canberra	0	0	0	0	0	0	0	0
Chebyshev	127	2224	5	5	134	3068	5	5
Cityblock	53	104	14	14	72	180	10	10
Correlation	149	11192	4	4	149	12354	4	4
Cosine	149	12588	3	3	149	13726	3	3
Euclidean	111	906	5	5	121	1500	6	6
Hamming	0	0	0	0	0	0	0	0

Table 16: Digraph construction summary e-neighbourhood part 3

Evaluating the k-NN approach is different to evaluating e-neighbourhood, as k-NN guarantees k outgoing-edges. Since there is no boundary for selecting the nearest neighbour, only that its the k closest neighbour, they should produce graphs with more edges and more connected graphs. Tables 17, 18 and 19 show the summary of the different distance functions for both k-NN graph and digraph. They show that k-NN produces more edges, however, all the actors clusters together to form smaller connected components. It is first when we increase k that the components gets fewer, but they contain more actors. See 2-NN and 3-NN in Table 17 compared to 1-NN.

Graph									
	1-NN			2-NN			3-NN		
Distance	**Nodes**	**Edges**	**Weak**	**Nodes**	**Edges**	**Weak**	**Nodes**	**Edges**	**Weak**
Braycurtis	150	107	43	150	210	4	150	302	2
Canberra	150	114	36	150	210	3	150	307	1
Chebyshev	150	101	49	150	206	5	150	306	5
Cityblock	150	110	40	150	213	2	150	303	2
Correlation	150	109	41	150	210	5	150	309	3
Cosine	150	108	42	150	208	5	150	308	4
Euclidean	150	107	43	150	209	4	150	302	4
Hamming	150	123	27	150	239	3	150	343	1

Table 17: Graph construction summary k-NN

Digraph								
	1-NN				2-NN			
Distance	**Nodes**	**Edges**	**Weak**	**Strong**	**Nodes**	**Edges**	**Weak**	**Strong**
Braycurtis	150	150	34	116	150	300	2	75
Canberra	150	150	40	110	150	300	2	61
Chebyshev	150	150	40	110	150	300	4	62
Cityblock	150	150	35	115	150	300	1	75
Correlation	150	150	42	108	150	300	5	64
Cosine	150	150	40	110	150	300	4	76
Euclidean	150	150	42	108	150	300	4	62
Hamming	150	150	26	124	150	300	3	99

Table 18: Digraph construction summary k-NN part 1

Digraph				
	3-NN			
Distance	**Nodes**	**Edges**	**Weak**	**Strong**
Braycurtis	150	450	1	37
Canberra	150	450	1	21
Chebyshev	150	450	1	33
Cityblock	150	450	1	35
Correlation	150	450	2	40
Cosine	150	450	1	42
Euclidean	150	450	2	32
Hamming	150	450	1	50

Table 19: Digraph construction summary k-NN part 2

An intial analysis of the k-NN approach shows an improved graph structure which includes all the actors. There is small differences in the number of edges they generate. However, the hamming distance generate more edges than the rest. This is propably caused by its unusual minimum and maximum values found in Table 11. Where the minimum is 0.5 and maximum is 1.0. k-NN graph, where $k = 1$, also produces more disconnected components, which can negatively affect the prioritisation. However, k-NN graphs show more advantages for implementation into digital forensic processes.

5.4 Experiment discussion

The experiments conducted in this chapter tested whether an actors communication behaviour can be used to identify them according to their organisational job positions and whether graph construction could be used for this purpose. This section discusses each experiment result and research question separately. First, we discuss the best features for identifying individuals of importance. Then we discuss the best methods for graph-based construction techniques for digital forensic.

5.4.1 Features for identification of individuals and groups

The purpose of this experiment is to see whether SNA methods can find many of the interesting actors within a network of communicating actors. We evaluate how “interesting” an actor is based on their job title, where we assume that job titles can reflect over to desired services provided by entities in the CaaS model. Therefore, we consider more central job such as a CEO or president will to be more interesting than (for example) managers and employees. The features that are used to identify the most interesting actors has been the measures of centrality provided by SNA process. The full result of this analysis is found in Appendix C.

After extracting the features for each Enron employee, we listed each feature individually, with the top 30 highest values for each of them. We have separated the results between graph and digraph, we will, therefore, do the same for this discussion. As they the graphs differ in structure, features and produces different results. We first discuss the results for the graph, before discussing digraph results.

In SNA of the graph, we observed that almost all of the features detected most of the CEOs and presidents. However, the features *communicability betweenness* and *closeness* contained most of the central job positions. For communicability betweennness, there was a total of 19 actors with central job titles, where three of them was CEOs and two presidents. It did also list more vice presidents and directors than any other feature. For closness, there was a total of 15 actors with central job titles, where two of them was CEOs and three presidents. It listed the same number of vice presidents as current flow betweenness, however, it is ranked higher as it listed more actors with central job positions.

In SNA of the digraph, we observed that its features had detected many of the same actors as the analysis of the graph. Those features was degree, out-degree, closeness, betweenness and eigenvector. However, the ordering of the actors was different. In-degree was the only that had a different set of actors displayed. The features *closeness* and *degree* contained most of the central job positions. For closeness, there was a total of 18 actors with central job titles, where three of them was CEOs and three presidents. Also, it listed as many vice presidents as in-degree, which is more than any other feature. For degree, there was a total of 17 actors, where three of them was CEOs, three presidents, ten vice presidents and one managing director. In-degree is listed with more high positions actors. However, degree found more of the job titles we are interested in.

Even though the centrality measures had a good detection rate of high job positions, some actors were either employees or had been listed as N/A. For example, the top two actors for each feature was either employee (ID 122) or N/A (ID 150). A closer inspection of actor with ID 150 showed that it was an assistant to president [71]. This shows that SNA can also find secondary targets, which can be taken down to cause a major disruption to the operations for the president. In addition, employee ID 122 is Chief Operating Officer (COO) and is responsible for the daily operation of the company, and usually routinely reports to a CEO.

From previous work, we expected to find most of the high ranking actors in betweennness centrality. Since it has been argued over its capability to capture the most important actors in a network [38, 37]. However, this assumption was proved to be false according to our experiment and dataset. It found 11 actors in the analysis of the graph and 13 actors for the digraph. Performing almost as bad as the worst features, eigenvector, katz and load.

We expected that there would be a difference between the two types of graphs as e-mail communication do not have to be reciprocated. However, it was not found any significant difference between graph and digraph when comparing them to each other. Digraph performed slightly better as it found on average 47,08% high position actors in its top 30 list, whereas graph found 39,58% in its lists.

5.4.2 Graph-based construction techniques for digital forensics

It is hard to read and compare each actor from the numbers and tables produced by SNA. Therefore, the purpose of this experiment is to see whether graph construction algorithms can find a better way of prioritising interesting actors. Where we prioritise according to the results from the SNA. The features used to construct the graphs is the measures of centrality, according to Table 7. As the graph construction resulted in 288 graphs, where 144 is graphs and 144 is digraphs, we have to limit the selection of graphs to analyse. A selection of the analysed graphs is found in Appendix D.

The ϵ-neighbourhood approaches for graph and digraph, where $\epsilon = \{0.01,$

$0.02, 0.03, 0.04, 0.05, 0.06\}$, was not included in this analysis. The reason for this is that they produce sparse and disconnected graphs, with many few components. A graphical example produced by e-neighbourhood is found in Figure 43. Sparse and disconnected graphs are difficult to analyse for some reasons:

- Not all actors are represented in the graph, only those actors with a similarity smaller than e.
- Increasing the value of e resulted in more dense and connected graphs, however, the edges gets connected to many other actors.
- The selection is therefore highly dependent on the selection of the value e.

For sparse and disconnected graphs, see Figure 43, the analysis difficulty lies in not knowing how the different components relate to another. It is necessary to know how the disconnected components are connected to have a prioritised ordering of actors. In the opposite case with a dense and connected graph, the difficulty is that it contains too many edges between the actors. It is, therefore, difficult to differentiate which actor is most similar to other actors.

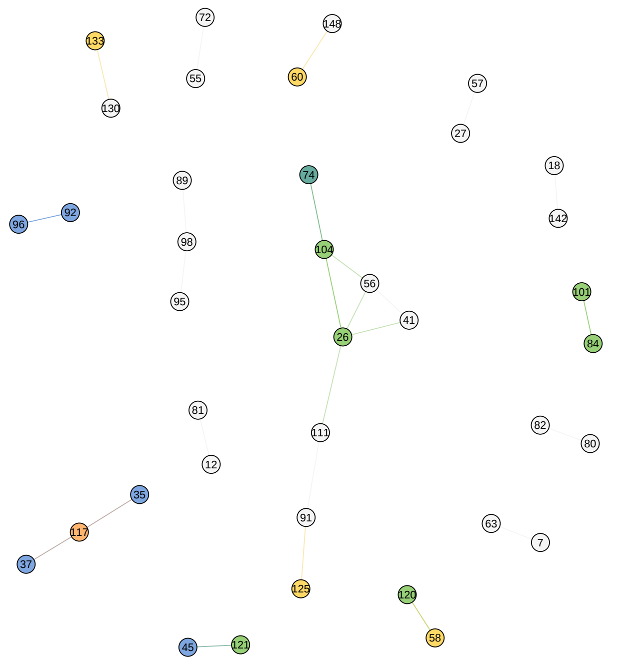

Figure 43: e-neighbourhood Euclidean, where $e = 0.04$, connected components

For the reasons mentioned above, k-NN approach will be better suited for prioritising actors as it includes all the actors in the graph. However, there is still $2 * 3 * 8 = 48$ k-NN graphs, where 24 is graphs and 24 is digraphs. This produces more over 40 pages with graphs for analysis and appendix section, so we are going to focus on the analysis of (undirected) graphs. Therefore, by reducing the analysis to one graph type we can compare the different values for k and each distance metric. The digraphs can be produced by following the steps in this chapter.

It is important to note that the relationship between the vertices do not reflect e-mail communication between the employees, but rather the similarity of the feature vectors. Where k-NN produces an edge between a feature vector and its closest k neighbour vectors. In Section 2.2.1 we discussed that a small distance was equivalent to large similarity. This similarity graph can be used to understand the whole picture given by the SNA tables and values.

Lists produced by SNA are good for an digital forensic investigator to follow. However, they should not rely on following one list that has worked in the past. As each measure of centrality capture one dimension of the communication pattern, there would be variations when some features work better than others. Therefore, a list loses the context of the other features, whereas the advantage with a similarity graph is that it takes into account all features when generating the graph.

According to Table 17, 1-NN produces disconnected graphs, where the smallest number of disconnected (weak) componets was 27. This is the same problem as with ϵ-neighbourhood approach, where the problem was of knowing the relations between the components. However, when increasing to 2-NN, the graph become more connected with a maximum of five components. We are going to focus our discussion on the 2-NN graph produced by Cityblock (Manhattan) with two components, where the graph is found in Figure 59.

The top five actors, ID 53 (CEO), 107 (President), 122 (Employee), 127 (CEO) and 150 (N/A) are connected into a tight cluster in all the 2-NN graphs, independent of the distance measures. These actors also appear together frequently in measures degree, closeness, betweenness, load and communicability betweenness. This suggests that the graph construction algorithm k-NN have captured the patterns found via SNA. Figure 44 illustrate one such component, from 2-NN Cityblock graph.

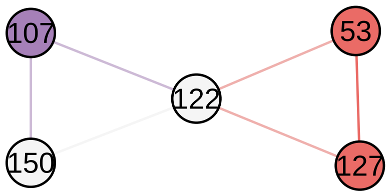

Figure 44: Top five actors component in 2-NN Cityblock

Figure 45 is a subgraph of the 2-NN graph, with Cityblock distance function. Note that the distances between vertices in this figure does not reflect the actual distance, it is only for illustration purposes. The complete graph can be found in Figure 59. It shows one complete graph component, and only parts of a second component, illustrated by the empty edges from vertices 21 and 59.

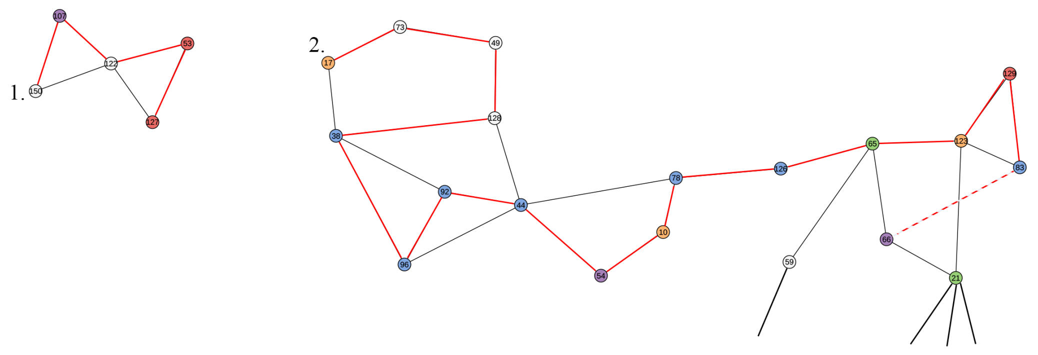

Figure 45: Investigation path

Our suggestion is that an investigator selects a start vertex based on experience or via our suggestion, to start with unexplored vertices from the SNA lists. We suggest starting with closeness centrality list, which is the best feature for both graph and digraphs to list actors with central positions. With this suggestion, we start with the first unexplored actor, with label 150, indicated with number one (1.) in the figure. The red edges is a proposed path that they can take to continue to investigate unexplored actors. This path is arbitrary, but it can be improved with adding weights to the edges, to illustrate the similarity between two actors. With weights is involved, we propose that an investigator follows the path with least weight as those actors are more similar.

When an investigator has visited each vertex in a connected component, we suggest that they check the SNA lists again to find the next unexplored actor. In this example, the next actor, with label 17, in indicated with number two (2.) in the figure. Because of the irregular graph generated by the k-NN approach, it is difficult to know which path to select as next actor. This could be solved by selecting the smallest weight in a weighted graph, or use an algorithm that guarantees regular graphs such as b-matching. Again, the red edges is a proposed path for an investigator.

Best distance measures are Braycurtis, Cityblock, Cosine and Euclidean, as they provided graphs with more chains than graphs with paths. As this will allow an investigator to follow one specific path rather than choosing between different paths.

6 Discussion, conclusion and further work

The previous chapters have presented theory and state of the art related to our research questions. Further, we presented our methodologies for solving the research questions, and the results of the experiments. The experiments focused on the issues of identifying important actors within a communication network. This chapter provides discussions of the implications as well as a summary of the thesis. The theoretical and practical implications of the obtained results are discussed before ending the discussion with a conclusion.

6.1 Theoretical implications

We wanted to demonstrate how people of importance could be identified via the communication patterns in a network. This identification was performed by analysing the e-mail communication inside the Enron corporation. We are not the first to use SNA centrality measures on the Enron dataset. However, to our knowledge we are the first to perform a systematical evaluation of the performance of each measure of centrality. Furthermore, we suggest a graph-based method for prioritising actors according to their importance.

Research question 1

> *Which features can be used to identify important and influential individuals within a network?*

We performed a literature review to determine the features that would highlight important individuals within a communicating network. SNA centrality measures were proposed as a solution to this problem. This thesis has used algorithms and methods from graph theory and SNA to develop the proposed method of identifying important actors. It includes several types of centrality measures, where each of them provides a different solution to what it means to be important.

The current situation for a digital forensic investigation is to manually analyse a suspicious underground forum to study actor's behaviours and decide who is important to that network. It can be concluded from the experiments that some SNA methods are better for finding actors with importance to the network.

The proposed algorithms are based on literature and have a strong theoretical foundation. They are widely used in several different research areas, most significantly found in sociological studies. So the investigators can benefit from the use of SNA methods as they fulfil the criteria in the *Daubert standard*. However, it is important to highlight that our dataset contained structures of an organisation and that we assume these same structures are found in organised crime.

The proposed centrality measures have been shown to automate the forensic analysis to collect intelligence about which individuals is considered important in a communication network. The remaining task of an investigator is to perform a manual investigation on each entity, to determine which individual they should prioritise to take down. They can benefit from a list of this sort, so they can prioritise other entities when there are jurisdictional or other issues during the investigation. Finding secondary entities to take down is a matter of following the same prioritisation list and find frequent communication parties with the one they intend to disrupt.

Also, these techniques can give a greater insight into the investigation as they help to identify entities who may not have been part of the original investigation.

Research question 2

How can graph construction techniques be applied in digital forensics for identifying targets?

From the different SNA centrality measures, we observed that some features performed better for the task of identifying important individuals. A challenge, however, is that none of the features alone will provide a reliable indicator of importance, instead, they can be combined for greater accuracy. Graph construction algorithms, more specifically neighbourhood approaches, have been shown to create similarity graphs.

To improve the identification process, with respect to the “no free lunch” theorem, we have conducted experiments using a variety of neighbourhood approaches and distance functions. Our method provides a better approach to the identification of targets to take down. An investigator can better follow a similarity graph’s edges to identify actors of importance since it takes into consideration all the features. However, the problem of finding the right starting point remains. Therefore, we suggest they should use experience and the list from SNA as a starting point in the graph.

6.2 Practical recommendations

The dataset used in this thesis was the Enron corpus, which is a popular dataset commonly used in scientific research. The Enron corporation contained original organisational structures and positions. It is uncertain if these assumptions about the network structure can be transferred to underground forums, such as those found in Darknet. As the dataset is over 900 MB, containing over two hundred thousand e-mail messages, it is possible that it may contain inconsistencies. In addition, our proposed methods do not cover all necessary aspects of being used in a complete digital forensic solution.

Changes in the criminal landscape require changes in the law enforcement skill sets. They can no longer afford to ignore the less visible, but no less damaging, phenomena such as cybercrime. However, it is no longer possible or efficient to seek to identify and prosecute all suspects for these crimes. Our proposed method have the potential to be a more efficient way to disrupt or prevent cybercriminal activities.

The experiments should be easy to reproduce as both the dataset and software tools is open source and free to use. The tools we used was mostly our Python scripts that used Networkx, Numpy, SciPy and other popular third packages. With tightening budgets, these algorithms and methods allow to effective routinely investigate these underground forums. In addition, the common forensic tool IBM i2 Analyst's Notebook already provide some of the SNA algorithms. We suggest that closeness centrality should be used for the analysis as it found most of the employees with higher positions within the undirected and directed version of the Enron dataset.

The type of dataset is very important when using SNA algorithms. We are not aware of any other ways to interpret their results, other than the sociological explanation of them, discussed in Section 2.4.1. Therefore, it is important to consider the dataset used with this approach. For example, entities with star-like (or fan-like) structures should have their neighbours removed to avoid generating an incorrect result. Star-like entities are graph structures with one vertex and k leaves, and they will, therefore, get an artificial high centrality score. The structure is known as a "star" in graph terminology.

6.3 General discussion

Graph-based algorithms and methods have been successfully used in digital forensic for malware analysis and network forensics. We used graphs for their power of abstraction, to solve the issue of identifying important actors within a communication network. The advantage with graphs is that vertices gain the expressive power of edge weight such as frequency and direction. This allows a graph to express the overall pattern of the network, which can be studied by investigators. However, the problem with this approach is that vertices loose the e-mail content, but this could be included in the vertices as features.

We successfully identify actors with high job titles in our experiment. Unfortunately, the dataset used in our experiment might not be a good substitution of an underground forum. Real organisations have a hierarchy they follow, whereas it has been suggested that cybercriminals are more loosely structured. Therefore, the problem with our approach is that it is unknown that real organisational structures can relate to such forums or entities within the CaaS business model. The SNA approach will also not discover if an entity has used a middle-man for his activities, without an investigation of those suspects.

Entities can hide behind multiple pseudonyms and across multiple underground forums. Our approach does not take this situation into consideration.

However, it should be possible to correlate entities in underground forums of different sizes. Comparisons of networks with different sizes is discussed in Section 2.4.1. Also, entities with multiple pseudonyms could be identified through analysing their area of interest, expertise or via their communication pattern.

The "garbage in, garbage out" principle (discussed in Section 2.2.3) says that feeding data of poor quality into the algorithm will produce a result of poor quality. Therefore, the preprocessing of the data play a central role in how data can be understood. We used the Enron dataset for our experiments, and we focused on undirected graphs in our experiment with graph construction algorithms. However, the process of identification and prioritisation could benefit from knowing the direction of similarity and how similar they are (weight). An investigator could then focus on the direction of the edge, and select those actors which are most similar to previously analysed actors. Therefore, this research could benefit from the analysis of directed graphs.

6.4 Conclusion

Anonymisation techniques allow cyber criminals to communicate with peers in digital undergrounds known as Darknets. Where the small group of technology-skilled individuals provide their services to potentially thousand other criminals, that do not possess the same level of technological skills. Since they often specialise in different tasks, they can also provide service to skilled criminals. Therefore, the initial goal of the thesis was to present a way of analysing such unknown underground forums for digital forensic investigators. To combat the threat from these forums and the challenge of analysing them.

In this thesis, we have shown that SNA centrality measures and similarity graph construction can aid investigators in this process of identification. From the research questions in Chapter 1, we first wanted to identify features that would identify important actors found in the smaller population of the CaaS business model. We assumed their identification is equivalent to the hierarchical structure found in organisations and corporations. Our assumption is based on the similarity between skilled individuals and employees with higher job positions. They are both regarded as having control over information in a communication network. In addition to being more sought after for their expertise.

The features we selected was the centrality measures in SNA, discussed in related work [37] and [56] as a solution for gathering intelligence about communication networks. We used analysed these centrality measures for an undirected and directed version of the Enron dataset. For the graph, we identified communicability betweenness and closeness as the best features. Where 63.33% and 50.00% of the first 30 employees had higher job titles. For the graph, it was in-degree and closeness that was the best features. Where both features had 60.00% of the first 30 employees had higher job titles. Our results show that the application of centrality measures find higher ranking actors within patterns found in the communicating network.

The second research question was related to how graph-based methods can improve the identification and prioritisation of entities. We used the neighbourhood approaches ϵ-neighbourhood and k-NN, each with eight distance functions. This graph improves on SNA centrality measures as it takes into account all of the features, to produce a similarity graph. Our result shows that ϵ-neighbourhood worked poorly as a way to prioritise actors, as it produced graphs with many disconnected components and not all actors were included. Therefore, only similar (under the threshold ϵ) was included, and this does not necessary equate to higher job positions. In addition, the disconnected components were difficult to analyse how they would relate do each other.

k-NN proved easier to analyse as it contained all the actors, where each actor was guaranteed to have k out-going edges. When $k = 1$ it had the same problem as ϵ-neighbourhood that created a graph with many disconnected components. However, when $k = 2$ the graph had become more connected, with maximum five components. We recomment the use previous experience and lists generated from the SNA process to identify a vertex for a starting point of investigation. After this initial starting vertex, the graph allow the investigator to follow the graph edges to other similar actors.

Law enforcement agencies are in the blind when they encounters an unknown network, such as those found in the Darknet. It is a challenging task to begin to understand the network, to identify targets of importance with the goal of stopping their criminal activity. Our work shows this work can be effectively done through analysing graphs, that represents such underground networks. It has contributed towards the goal of finding central actors within a communication network. We have bridged the gap between law enforcement agencies and cybercriminals by applying graph-based algorithms and methods on real world networks, and by proposing an improved way of prioritising users within those networks.

6.5 Further work

Based on our experimental results and proposed graph-based method for prioritising important actors, we propose several further research areas. We hope that our research motivates future work.

Differentiate between e-mail types, and including message content
We did not distinguish between e-mails types sent as TO, CC or BCC when creating our dataset. As these types are used differently, they should, therefore, not be treated equally. Sending an e-mail to another actor should count more toward the edge weight than copy or blind-copy.

In addition, the message content could also introduce a score to the edge weight. Where more casual messages had less weight than more important messages, thereby allowing the edge weights to reflect more of the real communication between actors. We propose the investigation on whether different weighting of e-mail types can result in increased accuracy of SNA methods.

Weighted voting of SNA centrality measures
In our experiments, we assumed that all measures of centrality were equally good at finding important actors within a network. However, our experiment showed that they were not equally in finding important actors. Therefore, we propose to investigate each centrality measure to produce weights that can be applied to the edges. We recommend that this be done through a dataset that already contain an evaluation of how important the actors is.

Dynamically changing communication graph
In our experiments, we used e-mail communication over four years. Human interaction is dynamically and always changing, and our experiment does not reflect these changes. We propose the investigation on whether dynamical graphs can better reflect the intercommunication between actors. Gephi already has support for viewing dynamically graphs. We have not looked into this, but it should require some additional attributes for the nodes and edges such as timestamp or time-period. Additional attributes such as geolocation, gender, language, and so forth could also provide valuable information for an investigator.

Real underground forum dataset
The Enron dataset is an old e-mail communication dataset. However, it is the best-case scenario that an investigator can get, with all the e-mail messages between individuals available. We propose the investigation of a worst-case scenario, where only public available discussion threads are analysed. During this thesis, there was a database leakage of a real hacker forum called Nulled.io, in addition to datasets available at `www.azsecure-data.org/dark-web-forums.html`. We recommend that these new datasets should be used for this research.

Communication over different types of mediums
Technology allows for multiple ways of communication, such as Voice over IP (VoIP), instant messaging and forum threads. We propose the investigation of finding how these types of communication medias differ concerning SNA. This is to effectively prepare for new types of technology, and to develop new techniques for analysing them.

Other graph construction algorithms
We focused on a particular set of neighbourhood approaches for graph construction. There exist many different algorithms; particularly b-matching could produce a better result for the prioritisation of important actors. We propose the investigation on whether other graph construction algorithms does have a better effect. We recommend that the implementation of these algorithms should be publicly published so the community can easily use them in other research areas.

Method for path selection
We have used neighbourhood approaches in our experiments. Since they are known to produce irregular graphs, this make it harder for an investigator to select the path they want to investigate. An irregular graph is where each vertex not necessarily have equal number of edges. We propose the investigation on finding methods for path traversal that best accomplishes the goal of prioritising entities. We recommend to use a weighted similarity graph, so to select a path with the smallest distance (equivalent to largest similarity). Alternatively, using b-matching algorithm as it will produce a regular graph.

Graph construction for file reconstruction
Graph construction proved a good way to illustrate similarities between feature vectors. As file (fragment) reconstruction is mostly based on clustering, from ML, it has to compare each fragment to all objects in a cluster. A similarity graph would produce paths that a computer can follow to reconstruct files, potentially reducing the number of comparisons and time for reconstruction.

Bibliography

[1] Europol. September 2014. The internet organised crime threat assessment (iocta). https://www.europol.europa.eu/sites/default/files/publications/europol_iocta_web.pdf. Online; accessed 29 September 2015.

[2] Manky, D. 2013. Cybercrime as a service: a very modern business. *Computer Fraud & Security*, 2013(6), 9–13.

[3] Europol. September 2015. The internet organised crime threat assessment (iocta). https://www.europol.europa.eu/sites/default/files/publications/europol_iocta_web_2015.pdf. Online; accessed 31 October 2015.

[4] Mallory, S. L. 2011. *Understanding Organized Crime*. Jones & Bartlett Learning.

[5] UN General Assembly. January 2001. United nations convention against transnational organized crime : resolution / adopted by the general assembly. Available at http://www.refworld.org/docid/3b00f55b0.html. Online; accessed 4 February 2016.

[6] Boister, N. 2003. 'transnational criminal law'? *European Journal of International Law*, 14(5), 953–976. URL: http://ejil.oxfordjournals.org/content/14/5/953.abstract, doi:10.1093/ejil/14.5.953.

[7] Tabansky, L. December 2012. Cybercrime: A national security issue? *Military and Strategic Affairs*, 4(3), 9–13.

[8] Choo, K.-K. R. 2008. Organised crime groups in cyberspace: a typology. *Trends in Organized Crime*, 11(3), 270–295. URL: http://dx.doi.org/10.1007/s12117-008-9038-9, doi:10.1007/s12117-008-9038-9.

[9] ITU. November 2014. Itu releases annual global ict data and ict development index country rankings. Available at http://www.itu.int/net/pressoffice/press_releases/2014/68.aspx#.VriGXJMrLfB. Online; accessed 8 February 2016.

[10] Warren, P. & Streeter, M. 2005. *Cyber Alert: How the World is Under Attack from a New Form of Crime*. Vision. URL: https://books.google.no/books?id=ThdIHgAACAAJ.

[11] Charlton, J. 2005. Al qaeda buys cyber criminal expertise. *Computer Fraud & Security*, 2005(3), 2 –. URL: http://www.sciencedirect.com/science/article/pii/S1361372305701664, doi:http://dx.doi.org/10.1016/S1361-3723(05)70166-4.

[12] Report, D. T. August 2001. A road map for digital forensic research. Available at https://www.dfrws.org/2001/dfrws-rm-final.pdf. Online; accessed 29 May 2016.

[13] Altheide, C. & Carvey, H. 2011. Digital forensics with open source tools. Syngress, Boston. doi:http://dx.doi.org/10.1016/B978-1-59749-586-8.00015-7.

[14] Alpaydin, E. 2004. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[15] Franke, K. May 2015. Digital & computational forensics. Available at http://www.fagforbundet.no/file.php?id=18286. Online; accessed 26 April 2016.

[16] Jain, A. K., Duin, R. P. W., & Mao, J. January 2000. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1), 4–37. URL: http://dx.doi.org/10.1109/34.824819, doi:10.1109/34.824819.

[17] Kononenko, I. & Kukar, M. 2007. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited.

[18] Mohri, M., Rostamizadeh, A., & Talwalkar, A. 2012. *Foundations of Machine Learning*. The MIT Press.

[19] Kohavi, R. & Provost, F. 1998. Glossary of terms. *Machine Learning*, 30(2-3), 271–274.

[20] Rouse, M. March 2008. Garbage in, garbage out (gigo). Available at http://searchsoftwarequality.techtarget.com/definition/garbage-in-garbage-out. Online; accessed 25 February 2016.

[21] Elomaa, T. & Rousu, J. a 2004. Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. *Data Min. Knowl. Discov.*, 8(2), 97–126. URL: http://dx.doi.org/10.1023/B:DAMI.0000015868.85039.e6, doi:10.1023/B:DAMI.0000015868.85039.e6.

[22] Kotsiantis, S. & Kanellopoulos, D. Discretization techniques: A recent survey.

[23] Theodoridis, S. & Koutroumbas, K. 2008. *Pattern Recognition, Fourth Edition*. Academic Press, 4th edition.

[24] Bellotti, T., Nouretdinov, I., Yang, M., & Gammerman, A. 2014. Chapter 6 - feature selection. In *Conformal Prediction for Reliable Machine Learning*, Balasubramanian, V. N., Ho, S.-S., & Vovk, V., eds, 115 – 130. Morgan Kaufmann, Boston. URL: `http://www.sciencedirect.com/science/article/pii/B9780123985378000067`, `doi:http://dx.doi.org/10.1016/B978-0-12-398537-8.00006-7`.

[25] Bermingham, M. L., Pong-Wong, R., Spiliopoulou, A., Hayward, C., Rudan, I., Campbell, H., Wright, A. F., Wilson, J. F., Agakov, F., Navarro, P., & Haley, C. S. May 2015. Application of high-dimensional feature selection: evaluation for genomic prediction in man. *Scientific Reports*, 5(10312).

[26] Brownlee, J. October 2014. An introduction to feature selection. Available at `http://machinelearningmastery.com/an-introduction-to-feature-selection/`. Online; accessed 29 February 2016.

[27] Powell, V. & Lehe, L. 2015. Principal component analysis: Explained visually. Available at `http://setosa.io/ev/principal-component-analysis/`. Online; accessed 12 April 2016.

[28] Ariu, D., Giacinto, G., & Roli, F. 2011. Machine learning in computer forensics (and the lessons learned from machine learning in computer security). In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, 99–104, New York, NY, USA. ACM. URL: `http://doi.acm.org/10.1145/2046684.2046700`, `doi:10.1145/2046684.2046700`.

[29] de Vel, O., Anderson, A., Corney, M., & Mohay, G. December 2001. Mining e-mail content for author identification forensics. *SIGMOD Rec.*, 30(4), 55–64. URL: `http://doi.acm.org/10.1145/604264.604272`, `doi:10.1145/604264.604272`.

[30] Li, W.-J., Wang, K., Stolfo, S. J., & Herzog, B. June 2005. Fileprints: identifying file types by n-gram analysis. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, 64–71. `doi:10.1109/IAW.2005.1495935`.

[31] Domingos, P. c 2012. A few useful things to know about machine learning. *Commun. ACM*, 55(10), 78–87. URL: `http://doi.acm.org/10.1145/2347736.2347755`, `doi:10.1145/2347736.2347755`.

[32] Wikipedia. February 2016. Cross-validation (statistics) — wikipedia, the free encyclopedia. Available at `https://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics)&oldid=705824849`. Online; accessed 3 March 2016.

[33] Sammut, C. & Webb, G. I. 2010. *Encyclopedia of Machine Learning*. Springer US.

[34] Cai, E. January 2014. Machine learning lesson of the day – the 'no free lunch' theorem. Available at `http://www.statsblogs.com/2014/01/25/machine-learning-lesson-of-the-day-the-no-free-lunch-theorem/`. Online; accessed 21 May 2016.

[35] Lischinsky, P. January 2014. Machine learning lesson of the day – the "no free lunch" theorem. Available at `https://chemicalstatistician.wordpress.com/2014/01/24/machine-learning-lesson-of-the-day-the-no-free-lunch-theorem/`. Online; accessed 3 March 2016.

[36] Rosen, K. H. 2007. *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 6th edition.

[37] Sparrow, M. K. 1991. The application of network analysis to criminal intelligence: An assessment of the prospects. *Social Networks*, 13(3), 251 – 274. URL: `http://www.sciencedirect.com/science/article/pii/037887339190008H`, `doi:http://dx.doi.org/10.1016/0378-8733(91)90008-H`.

[38] Prell, C. 2011. *Social Network Analysis: History, Theory and Methodology*. Sage Publications Ltd.

[39] Weisstein, E. W. 2015. Clique. Online; accessed 2 December 2015. URL: `http://mathworld.wolfram.com/Clique.html`.

[40] Algolist.net. 2015. Undirected graphs representation. Online; Accessed 1 December 2015. URL: `http://www.algolist.net/Data_structures/Graph/Internal_representation`.

[41] Freeman, L. C. 1978. Centrality in social networks conceptual clarification. *Social Networks*, 1(3), 215 – 239. URL: `http://www.sciencedirect.com/science/article/pii/0378873378900217`, `doi:http://dx.doi.org/10.1016/0378-8733(78)90021-7`.

[42] Hanneman, R. & Riddle, M. 2005. *Introduction to Social Network Methods*. University of California. URL: `https://books.google.no/books?id=wAHaygAACAAJ`.

[43] Pitas, I. 2016. *Graph-Based Social Media Analysis*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press. URL: https://books.google.no/books?id=BvYYCwAAQBAJ.

[44] Jebara, T., Wang, J., & Chang, S.-F. 2009. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 441–448, New York, NY, USA. ACM. URL: http://doi.acm.org/10.1145/1553374.1553432, doi:10.1145/1553374.1553432.

[45] Sageman, M. 2004. *Understanding Terror Networks*. University of Pennsylvania Press, Incorporated. URL: https://books.google.no/books?id=SAQ8Oa6zWF4C.

[46] Canali, C., Colajanni, M., & Lancellotti, R. Dataacquisition in social networks: Issues and proposals.

[47] Borgatti, S. P. & Molina, J. L. 2003. Ethical and strategic issues in organizational social network analysis. *The Journal of Applied Behavioral Science*, 39(3), 337–349.

[48] Klosek, J. 2010. *Protecting Your Health Privacy: A Citizen's Guide to Safeguarding the Security of Your Medical Information*. Praeger. URL: https://books.google.no/books?id=K0ua0gm-acUC.

[49] Narayanan, A. & Shmatikov, V. 2008. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, 111–125, Washington, DC, USA. IEEE Computer Society. URL: http://dx.doi.org/10.1109/SP.2008.33, doi:10.1109/SP.2008.33.

[50] Sutherland, E., Cressey, D., & Luckenbill, D. 1992. *Principles of Criminology*. The Reynolds Series in Sociology. AltaMira Press. URL: https://books.google.no/books?id=JVB3AAAAQBAJ.

[51] Calderoni, F. *Encyclopedia of Criminology and Criminal Justice*, chapter Social Network Analysis of Organized Criminal Groups, 4972–4981. Springer New York, New York, NY, 2014. URL: http://dx.doi.org/10.1007/978-1-4614-5690-2_239, doi:10.1007/978-1-4614-5690-2_239.

[52] Boundless. May 2016. Differential association theory. Available at https://goo.gl/TPHRVk or https://www.boundless.com. Online; accessed 26 May 2016.

[53] Motoyama, M., McCoy, D., Levchenko, K., Savage, S., & Voelker, G. M. 2011. An analysis of underground forums. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, 71–80, New York, NY, USA. ACM. URL: `http://doi.acm.org/10.1145/2068816.2068824`, `doi:10.1145/2068816.2068824`.

[54] Wang, W. *A graph oriented approach for network forensic analysis*. PhD thesis, Iowa State University, 2010.

[55] Ding, Y., Dai, W., Yan, S., & Zhang, Y. 2014. Control flow-based opcode behavior analysis for malware detection. *Computers & Security*, 44, 65–74.

[56] Irons, A. & Lallie, H. S. 2014. Digital forensics to intelligent forensics. *Future Internet*, 6(3), 584–596.

[57] Ortiz-Arroyo, D. 2010. Discovering sets of key players in social networks. In *Computational Social Network Analysis*, Abraham, A., Hassanien, A.-E., & Snáŝel, V., eds, Computer Communications and Networks, 27–47. Springer London. URL: `http://dx.doi.org/10.1007/978-1-84882-229-0_2`, `doi:10.1007/978-1-84882-229-0_2`.

[58] Clarke, N., Furnell, S., & Katos, V. 2013. *Proceedings of the European Information Security Multi-Conference (EISMC 2013)*. Lulu.com. URL: `https://books.google.no/books?id=x2SiBQAAQBAJ`.

[59] Diesner, J., Frantz, T. L., & Carley, K. M. 2005. Communication networks from the enron email corpus "it's always about the people. enron is no different". *Computational & Mathematical Organization Theory*, 11(3), 201–228. URL: `http://dx.doi.org/10.1007/s10588-005-5377-0`, `doi:10.1007/s10588-005-5377-0`.

[60] IBM Software. May 2016. i2 analyst's notebook. Available at `http://www-03.ibm.com/software/products/no/analysts-notebook`. Online; accessed 30 May 2016.

[61] Chen, X., Mu, Y., Liu, H., Yan, S., Rui, Y., & Chua, T.-S. December 2013. Large-scale multilabel propagation based on efficient sparse graph construction. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(1), 6:1–6:20. URL: `http://doi.acm.org/10.1145/2542205.2542209`, `doi:10.1145/2542205.2542209`.

[62] Wasserman, S. & Faust, K. 1994. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press.

[63] Newman, M. E. J., Watts, D. J., & Strogatz, S. H. 2002. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1), 2566–2572. URL: `http://www.pnas.org/content/99/suppl_1/2566.abstract`, `arXiv:http://www.pnas.org/content/99/suppl_1/2566.full.pdf`.

[64] Klimt, B. & Yang, Y. 2004. The enron corpus: A new dataset for email classification research. 217–226.

[65] Zhou, Y., Goldberg, M., Magdon-Ismail, M., & Wallace, W. A. 2007. Strategies for cleaning organizational emails with an application to enron email dataset. 5th Conf. of North American Association for Computational Social and Organizational Science (NAACSOS 07).

[66] Cohen, W. W. 2004. Enron email dataset. `http://www.cs.cmu.edu/~enron/`. Online; accessed 1 October.

[67] Shetty, J. & Adibi, J. The enron email dataset database schema and brief statistical report. Technical report, 2004.

[68] Pfeiffer, J. Enron mysql 5.5 dump file overview. Available at `https://www.cs.purdue.edu/homes/jpfeiff/enron.html`. Online; accessed 3 May 2016.

[69] Networkx. May 2016. High-productivity software for complex networks. Available at `https://networkx.github.io/`. Online; accessed 15 May 2016.

[70] Gilbert, E. 2012. Phrases that signal workplace hierarchy. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, 1037–1046, New York, NY, USA. ACM. URL: `http://doi.acm.org/10.1145/2145204.2145359`, `doi:10.1145/2145204.2145359`.

[71] Zhang, H., Spiliopoulou, M., Mobasher, B., Giles, C., McCallum, A., & Nasraoui, O. 2009. *Advances in Web Mining and Web Usage Analysis: 9th International Workshop on Knowledge Discovery on the Web*. Lecture Notes in Artificial Intelligence. Springer. URL: `https://books.google.no/books?id=8me70KLsvw4C`.

A Enron employee list

This list of Enron employees contains all employee ID and label (which is their Enron e-mail address). Center for Imaging Science (CIS) have the same list including each employees position within the company, available from `http://cis.jhu.edu/~parky/Enron/employees`. Some employees was (as what we believe) sensored with the string "xxx" after the employee e-mail address (or username). They are converted to N/A in this table. Other employees had full e-mail address/username, first and last name but "N/A" in their position within the company. We suspect they were not recorded when the table was first created. This list over employees follow the list found in the Enron database table.
These employees are not listed in the database *employeelist* table:

andrew.lewis@enron.com	Director
brenda.whitehead@enron.com	N/A
clint.dean@enron.com	N/A
daren.farmer@enron.com	Manager
darron.giron@enron.com	Employee
david.delainey@enron.com	CEO
debra.bailey@enron.com	N/A
d..martin@enron.com	Vice President
e.taylor@enron.com	Employee
f..keavey@enron.com	Employee
fletcher.sturm@enron.com	Vice President
gretel.smith@enron.com	N/A
hunter.shively@enron.com	Vice President
james.steffes@enron.com	Vice President
jeffrey.hodge@enron.com	Managing Director
jeffrey.shankman@enron.com	President
j.harris@enron.com	N/A
j..kaminski@enron.com	Manager
joannie.williamson@enron.com	N/A
john.forney@enron.com	Manager
john.lavorato@enron.com	CEO
judy.hernandez@enron.com	N/A
kevin.presto@enron.com	Vice President
larry.campbell@enron.com	Employee
margaret.carson@enron.com	N/A
mark.e.haedicke@enron.com	Managing Director
mark.haedicke@enron.com	Managing Director
mark.taylor@enron.com	Employee
michele.lokay@enron.com	Employee
mike.mcconnell@enron.com	N/A
m..scott@enron.com	N/A
m..smith@enron.com	N/A
m..tholt@enron.com	Vice President
patrice.mims@enron.com	N/A
paul.thomas@enron.com	N/A
phillip.allen@enron.com	Manager
phillip.love@enron.com	N/A
randall.gay@enron.com	N/A
richard.sanders@enron.com	Vice President
rob.gay@enron.com	N/A
sandra.brawner@enron.com	Director
steven.kean@enron.com	Vice President
susan.pereira@enron.com	Employee
s..ward@enron.com	N/A
t..hodge@enron.com	Managing Director
thomas.martin@enron.com	Vice President
vince.kaminski@enron.com	Manager
v.weldon@enron.com	N/A

These employees from the database table was not recorded in the CIS employee list:

45, 56, 76, 84, 89, 92, 93, 109, 110, 118, 122, 127, 143, 147, 148

They are listed as blank in this table, but treated as not available.

#	Label	Position
1	robert.badeer@enron.com	Director
2	kevin.hyatt@enron.com	Director
3	tracy.geaccone@enron.com	Employee
4	teb.lokey@enron.com	Manager
5	richard.ring@enron.com	Employee
6	taylor@enron.com	Employee
7	theresa.staab@enron.com	Employee
8	w..pereira@enron.com	Employee
9	stephanie.panus@enron.com	Employee
10	k..allen@enron.com	Manager
11	mark.whitt@enron.com	N/A
12	t..lucci@enron.com	Employee
13	marie.heard@enron.com	N/A
14	b..sanders@enron.com	Vice President
15	monika.causholli@enron.com	Employee
16	michelle.cash@enron.com	N/A
17	mike.grigsby@enron.com	Manager
18	lynn.blair@enron.com	N/A
19	lindy.donoho@enron.com	Employee
20	kimberly.watson@enron.com	N/A
21	keith.holst@enron.com	Director
22	joe.quenet@enron.com	Trader
23	d..steffes@enron.com	Vice President
24	m..forney@enron.com	Manager
25	jay.reitmeyer@enron.com	Employee
26	frank.ermis@enron.com	Director
27	elizabeth.sager@enron.com	Employee
28	darrell.schoolcraft@enron.com	N/A
29	danny.mccarty@enron.com	Vice President
30	bill.rapp@enron.com	N/A
31	benjamin.rogers@enron.com	Employee
32	shelley.corman@enron.com	Vice President
33	kim.ward@enron.com	N/A
34	juan.hernandez@enron.com	Employee
35	joe.stepenovitch@enron.com	Vice President
36	a..shankman@enron.com	President
37	jane.tholt@enron.com	Vice President
38	barry.tycholiz@enron.com	Vice President
39	dana.davis@enron.com	Vice President
40	tori.kuykendall@enron.com	Trader
41	l..gay@enron.com	N/A
42	martin.cuilla@enron.com	Manager
43	f..campbell@enron.com	Employee
44	john.arnold@enron.com	Vice President
45	harry.arora@enron.com	Vice President
46	w..delainey@enron.com	CEO
47	brad.mckay@enron.com	Employee
48	tana.jones@enron.com	N/A
49	susan.scott@enron.com	N/A
50	susan.bailey@enron.com	N/A
51	j..kean@enron.com	Vice President
52	kay.mann@enron.com	Employee
53	lavorato@enron.com	CEO
54	greg.whalley@enron.com	President
55	paul.y barbo@enron.com	
56	jason.wolfe@enron.com	N/A
57	jason.williams@enron.com	N/A
58	jim.schwieger@enron.com	Trader
59	monique.sanchez@enron.com	N/A
60	kevin.ruscitti@enron.com	Trader
61	matthew.lenhart@enron.com	Employee
62	peter.keavey@enron.com	Employee
63	scott.hendrickson@enron.com	N/A
64	tom.donohoe@enron.com	N/A
65	e..haedicke@enron.com	Managing Director
66	stanley.horton@enron.com	President
67	sara.shackleton@enron.com	N/A
68	rod.hayslett@enron.com	Vice President
69	richard.shapiro@enron.com	Vice President
70	michelle.lokay@enron.com	Employee
71	matt.smith@enron.com	N/A
72	mark.mcconnell@enron.com	N/A
73	jeff.dasovich@enron.com	Employee
74	james.derrick@enron.com	In-House Lawyer
75	gerald.nemec@enron.com	N/A
76	debra.perlingiere@enron.com	N/A
77	john.zufferli@enron.com	Vice President
78	andy.zipper@enron.com	Vice President
79	w..white@enron.com	N/A
80	charles.weldon@enron.com	N/A
81	judy.townsend@enron.com	Employee
82	d..thomas@enron.com	N/A
83	j..sturm@enron.com	Vice President
84	geoff.storey@enron.com	Director
85	geir.solberg@enron.com	Employee
86	cara.semperger@enron.com	Employee
87	holden.salisbury@enron.com	Employee
88	eric.saibi@enron.com	Trader
89	andrea.ring@enron.com	N/A
90	cooper.richey@enron.com	Manager
91	dutch.quigley@enron.com	N/A
92	m..presto@enron.com	Vice President
93	phillip.platter@enron.com	Employee
94	vladi.pimenov@enron.com	N/A
95	joe.parks@enron.com	N/A
96	scott.neal@enron.com	Vice President
97	matt.motley@enron.com	Director
98	l..mims@enron.com	N/A
99	albert.meyers@enron.com	Employee
100	errol.mclaughlin@enron.com	Employee
101	jonathan.mckay@enron.com	Director
102	larry.may@enron.com	Director
103	a..martin@enron.com	Vice President
104	mike.maggi@enron.com	Director
105	m..love@enron.com	N/A
106	h..lewis@enron.com	Director
107	louise.kitchen@enron.com	President
108	jeff.king@enron.com	Manager
109	j.kaminski@enron.com	Manager
110	john.hodge@enron.com	Managing Director
111	john.griffith@enron.com	N/A
112	c..giron@enron.com	Employee
113	doug.gilbert-smith@enron.com	
114	chris.germany@enron.com	Employee
115	lisa.gang@enron.com	N/A
116	mary.fischer@enron.com	Employee
117	j..farmer@enron.com	Manager
118	chris.dorland@enron.com	Manager
119	mike.carson@enron.com	Manager
120	f..brawner@enron.com	Director
121	robert.benson@enron.com	Director
122	sally.beck@enron.com	Employee
123	rick.buy@enron.com	Manager
124	don.baughman@enron.com	Trader
125	eric.bass@enron.com	Trader
126	s..shively@enron.com	Vice President
127	kenneth.lay@enron.com	CEO
128	kam.keiser@enron.com	Employee
129	jeff.skilling@enron.com	CEO
130	sean.crandall@enron.com	
131	ryan.slinger@enron.com	Trader
132	mike.swerzbin@enron.com	Trader
133	diana.scholtes@enron.com	Trader
134	craig.dean@enron.com	Trader
135	bill.williams@enron.com	N/A
136	stacy.dickson@enron.com	Employee
137	drew.fossum@enron.com	Vice President
138	mark.guzman@enron.com	
139	mary.hain@enron.com	
140	lysa.akin@enron.com	
141	steven.harris@enron.com	
142	dan.hyvl@enron.com	Employee
143	eric.linder@enron.com	
144	steven.merris@enron.com	
145	robin.rodrigue@enron.com	
146	steven.south@enron.com	N/A
147	carol.clair@enron.com	
148	chris.stokley@enron.com	
149	kate.symes@enron.com	Employee
150	liz.taylor@enron.com	N/A
151	rosalee.fleming@enron.com	

B Complete graphs

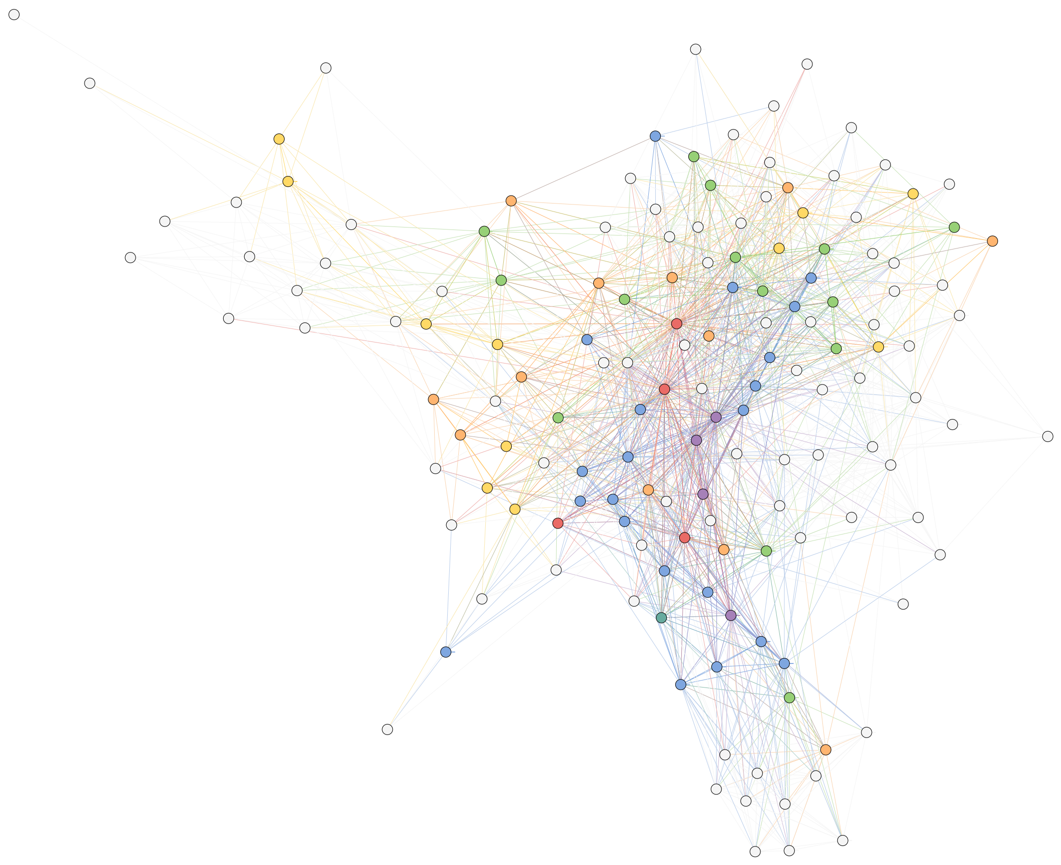

Figure 46: Complete graph of population sample

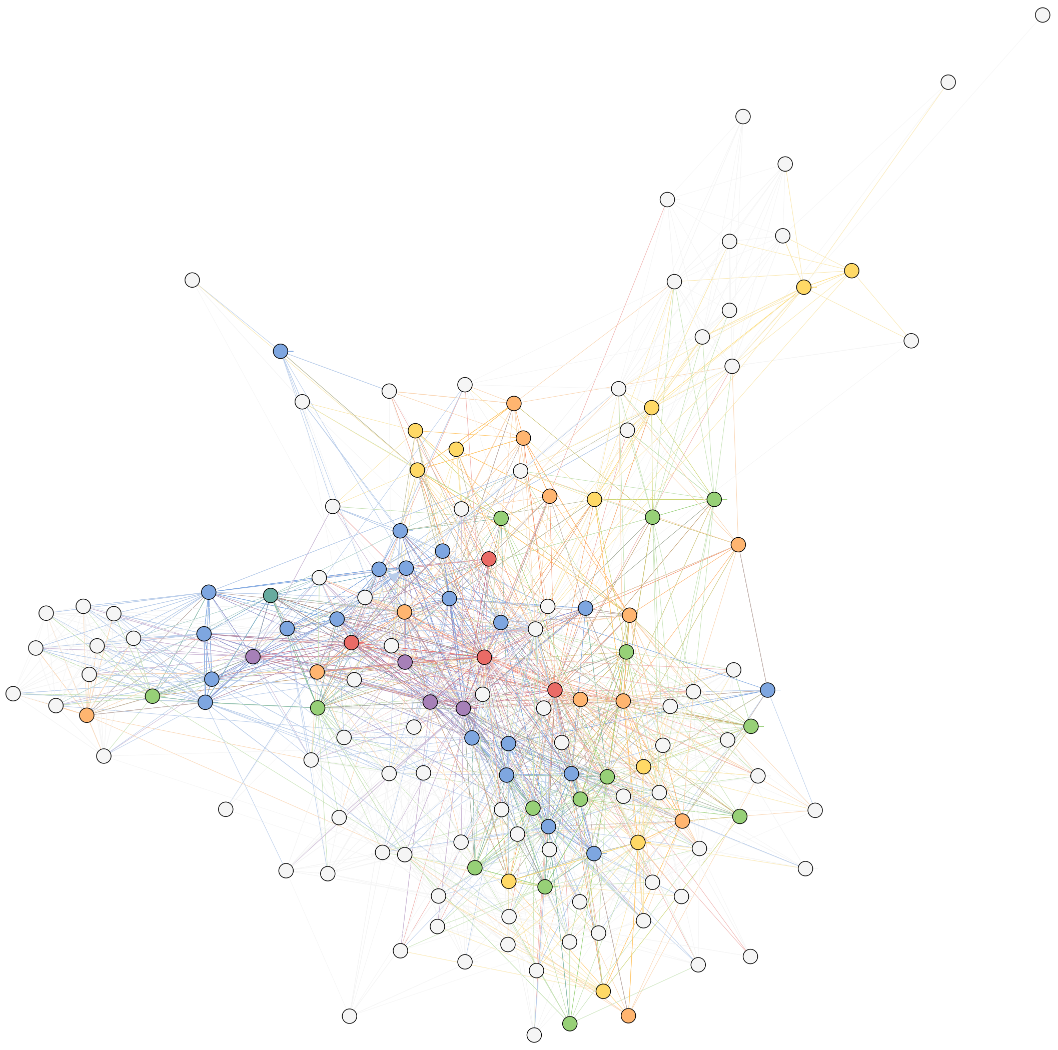

Figure 47: Complete digraph of population sample

C Social network analysis results

Position	ID	Degree	Position	ID	Closeness
N/A	150	0.4966442953020134	N/A	150	0.6592920353982301
Employee	122	0.4899328859060403	Employee	122	0.645021645021645
CEO	53	0.42953020134228187	CEO	53	0.6286919831223629
President	107	0.4228187919463087	President	107	0.6208333333333333
Employee	27	0.40268456375838924	CEO	127	0.6182572614107884
Manager	17	0.2953020134228188	Manager	17	0.5752895752895753
Employee	128	0.2885906040268456	Vice President	92	0.5643939393939394
N/A	49	0.2885906040268456	Vice President	38	0.5601503759398496
Vice President	92	0.26174496644295303	Vice President	96	0.5498154981549815
Vice President	96	0.2550335570469799	N/A	49	0.5498154981549815
Vice President	38	0.2483221476510067	Employee	73	0.5498154981549815
Employee	73	0.2483221476510067	President	54	0.5457875457875457
President	54	0.22818791946308725	Vice President	44	0.5437956204379562
Vice President	44	0.22818791946308725	N/A	48	0.5418181818181819
Vice President	78	0.22818791946308725	Vice President	78	0.5398550724637681
Vice President	126	0.21476510067114093	Manager	123	0.5379061371841155
N/A	48	0.21476510067114093	Employee	128	0.5379061371841155
N/A	59	0.2080536912751678	CEO	129	0.5379061371841155
Manager	10	0.2080536912751678	Manager	10	0.5359712230215827
CEO	129	0.20134228187919462	N/A	33	0.5340501792114696
Vice President	83	0.20134228187919462	Managing Director	65	0.5340501792114696
N/A	33	0.20134228187919462	Vice President	69	0.5340501792114696
N/A	75	0.20134228187919462	Vice President	83	0.5321428571428571
Vice President	137	0.19463087248322147	Employee	27	0.5302491103202847
Employee	27	0.19463087248322147	N/A	75	0.5283687943262412
Director	101	0.19463087248322147	President	66	0.5265017667844523
Managing Director	65	0.19463087248322147	N/A	59	0.5246478873239436
President	66	0.19463087248322147	Director	21	0.5228070175438596
N/A	57	0.18791946308724833	Vice President	23	0.5209790209790209
Manager	123	0.18120805369127516	Vice President	126	0.5209790209790209

Table 20: Graph degree and closeness centrality results

Position	ID	Betweenness	Position	ID	Eigenvector
N/A	150	0.12741903205662952	N/A	67	0.5484543984046928
Employee	122	0.0771999546616338	N/A	48	0.49700038653622225
CEO	53	0.07128931996362767	N/A	147	0.39518085824653326
CEO	127	0.06087954853442533	N/A	13	0.30432015715026534
N/A	49	0.05480578002058777	N/A	50	0.3010855876166442
President	107	0.049624345900325634	Employee	9	0.29458298665378435
N/A	135	0.032366349181675066	Employee	6	0.08423610844462812
Employee	73	0.03046819220287483	Employee	27	0.0756057981648209
N/A	33	0.02565356746845891	Employee	136	0.07040947636419569
Manager	17	0.023740618469121476	N/A	57	0.05817183899861572
Employee	128	0.02106796329345721	President	107	0.0344705160349727
N/A	75	0.019034444664106517	Employee	73	0.02910497832110247
President	66	0.018280869446014737	Vice President	69	0.02693984537623267
Vice President	38	0.018181740027529956	N/A	33	0.017416032369916897
N/A	140	0.01682247962835146	Vice President	23	0.016513291438288048
Vice President	68	0.015177805015212351	Employee	52	0.01466135593533686
N/A	48	0.014017748147200963	N/A	150	0.01270954296430585
Vice President	92	0.012127292160119813	Vice President	44	0.010769154042500612
Employee	52	0.01197444925337974	Managing Director	65	0.00871332127231574
Vice President	96	0.011049030820347389	Employee	142	0.008179348615640637
Employee	27	0.01070099147481135	Vice President	92	0.007965403632577193
Managing Director	65	0.010421681352084381	Vice President	78	0.00796453147706711
N/A	139	0.009844761717010667	Vice President	38	0.007733705214378049
N/A	57	0.008603987492661204	N/A	91	0.00772721408987753
N/A	76	0.008491717923483395	N/A	139	0.006496199473090178
Vice President	23	0.0077490378271448815	Vice President	51	0.005264396884965995
Vice President	32	0.007729347919215445	Employee	122	0.004340854098370539
N/A	59	0.007592212592856629	Manager	17	0.003889894483466084

Table 21: Graph betweenness and eigenvector centrality results

Position	**ID**	**Katz**	**Position**	**ID**	**Load**
Director	106	0.27083515497575944	N/A	150	0.12202317376736826
Manager	108	0.1943822198843708	Employee	122	0.0749447076684403
N/A	148	0.19332995710295922	CEO	53	0.07170193047811683
Trader	60	0.18324243086890468	CEO	127	0.059724024971851586
N/A	113	0.17835118356319138	N/A	49	0.053555813387741416
N/A	79	0.15623876546680746	President	107	0.04905674315636346
Employee	85	0.15471874637429617	N/A	135	0.034558746004214005
N/A	140	0.1424285520623401	Employee	73	0.029936747738610103
Employee	15	0.14094533992454614	N/A	33	0.02581228226931536
Manager	42	0.13787232837246427	Manager	17	0.02303920029683394
Manager	109	0.1361845447599217	Employee	128	0.0209920140199694
CEO	53	0.13561006189208738	N/A	75	0.01931366634632139
N/A	64	0.13231945246309346	President	66	0.01798053109032224
N/A	72	0.12920736554705878	Vice President	38	0.017563251595088373
Manager	10	0.1176746581491969	N/A	140	0.017191862424829202
N/A	80	0.11322472422213939	Vice President	137	0.014505452618174286
N/A	82	0.1126313082113631	Vice President	68	0.014286960620496947
Vice President	51	0.11172423040288532	N/A	48	0.013839326662930345
President	54	0.10690005116331523	Employee	114	0.01366941922233052
Employee	87	0.10331211043591552	Employee	52	0.013327235550118876
N/A	94	0.09740159212627263	Vice President	92	0.012027356643537138
N/A	55	0.09691462482124993	Employee	27	0.010845184878195652
N/A	16	0.09528478632828273	Vice President	96	0.010839189149365639
President	36	0.08989688226411195	Managing Director	65	0.010322754675794492
CEO	127	0.08962345847982207	N/A	139	0.010276975594747113
Employee	43	0.08445535367063431	N/A	76	0.008700937078845608
Vice President	137	0.0842524357824327	Vice President	32	0.00825383446060586
Employee	99	0.06952090790212889	Vice President	23	0.007911441515488607
N/A	145	0.06260104388956907	N/A	59	0.007416120123040826

Table 22: Graph katz and load centrality results

Position	**ID**	**Current-flow closeness**	**Position**	**ID**	**Current-flow betweenness**
President	107	0.6467799882131677	N/A	150	0.20032855968508428
N/A	150	0.6389661879385443	President	107	0.1860438175805059
Manager	17	0.635764500612942	Employee	73	0.1559911048523174
Employee	73	0.6340578529725237	Manager	17	0.15213481265904769
Vice President	69	0.624241308465019	N/A	49	0.12460083975654498
N/A	48	0.6185319433429298	Employee	122	0.10842293886747335
Employee	122	0.6161062882772926	N/A	48	0.07497359104222295
Vice President	38	0.6151686793189958	Employee	114	0.07462935109494692
Vice President	23	0.614519404160276	Vice President	38	0.07376415068088717
N/A	67	0.613287224001205	N/A	135	0.07000278332793267
N/A	13	0.6055026561122205	N/A	139	0.06730254289422952
Employee	9	0.6041223653246481	Vice President	96	0.06471196267723739
N/A	147	0.6022580233576166	Employee	128	0.06456817287350414
N/A	49	0.6002257089330718	N/A	75	0.06026463906777592
N/A	75	0.597027516157679	N/A	151	0.05894353855247506
N/A	50	0.5966946598118354	N/A	140	0.05770949179208586
Employee	128	0.5958701069090415	Vice President	137	0.0555954652831289
N/A	33	0.5956816156223563	N/A	33	0.05435034625519146
N/A	151	0.5934262212595738	Vice President	92	0.05320533073556542
N/A	139	0.5918030447634124	Vice President	69	0.05272553500947823
Vice President	92	0.5905785050567769	Vice President	23	0.052488778468610255
Employee	61	0.5894611799547581	Vice President	44	0.050804019160239015
Vice President	44	0.5894487814880507	Employee	61	0.05028115330852095
President	54	0.5853131066700509	CEO	127	0.04971538820191448
Vice President	51	0.5850549879446145	N/A	141	0.04827370128989803
Vice President	137	0.5843153835450076	Employee	52	0.04717523980139562
Employee	27	0.5827658797646988	Employee	27	0.04631946365119866
N/A	57	0.5827190803124106	N/A	76	0.04612563419976971
N/A	141	0.5799087094586808	Vice President	32	0.04526331173599807
N/A	76	0.5760741190760632	Employee	149	0.04483809986798685

Table 23: Graph current flow closeness and current flow betweenness centrality results

Position	**ID**	**Communicability betweenness**
Employee	122	0.7829800076092372
N/A	150	0.7699423081856573
President	107	0.7229092802664814
CEO	127	0.6769885851312053
CEO	53	0.662217185039693
Manager	17	0.4719586283665264
Vice President	92	0.4338540679053311
Vice President	96	0.4263958374398043
Vice President	38	0.4238913937923879
Vice President	44	0.41269454368465897
Vice President	78	0.40421595632831486
President	54	0.39616785350040484
Vice President	126	0.3793176357645377
Manager	10	0.3749448691117962
Employee	128	0.35191030652519323
Vice President	83	0.34498301350537985
CEO	129	0.3294282073974352
Employee	73	0.3116800168592963
N/A	49	0.3096793664140022
Manager	123	0.27884190219867383
Managing Director	65	0.2621594815928801
N/A	48	0.26164351867758706
N/A	59	0.26050894835002497
Director	101	0.24934151920303904
Director	84	0.2430405738257549
Director	21	0.2298501800537978
Vice President	69	0.22958155898955004
Vice President	51	0.22795060768425818
Employee	27	0.22621011371315286
Vice President	103	0.2253664183665956

Table 24: Graph communicability betweenness centrality result

Position	**ID**	**Degree**	**Position**	**ID**	**In-degree**
President	107	0.7046979865771812	President	107	0.30201342281879195
Employee	122	0.5906040268456376	Manager	17	0.2483221476510067
N/A	150	0.5704697986577181	President	54	0.22147651006711408
Manager	17	0.4966442953020134	Vice President	96	0.22147651006711408
CEO	127	0.4899328859060403	Vice President	38	0.18791946308724833
CEO	53	0.44295302013422816	Vice President	126	0.18120805369127516
N/A	49	0.4228187919463087	Director	84	0.174496644295302
Vice President	96	0.4093959731543624	N/A	33	0.174496644295302
Employee	128	0.40268456375838924	Vice President	92	0.174496644295302
Vice President	92	0.3825503355704698	N/A	57	0.16778523489932887
Employee	73	0.3624161073825503	Vice President	44	0.16778523489932887
Vice President	44	0.35570469798657717	N/A	48	0.15436241610738255
Vice President	38	0.3422818791946309	N/A	49	0.15436241610738255
Vice President	137	0.3221476510067114	N/A	75	0.15436241610738255
President	54	0.3221476510067114	Employee	27	0.1476510067114094
N/A	48	0.3221476510067114	Vice President	68	0.1476510067114094
N/A	75	0.3221476510067114	Vice President	78	0.1476510067114094
N/A	33	0.31543624161073824	Employee	25	0.14093959731543623
Vice President	126	0.3087248322147651	Director	21	0.14093959731543623
Vice President	78	0.3087248322147651	Manager	123	0.14093959731543623
President	66	0.2953020134228188	Vice President	103	0.14093959731543623
N/A	57	0.28187919463087246	Managing Director	65	0.14093959731543623
Managing Director	65	0.28187919463087246	President	66	0.14093959731543623
N/A	76	0.28187919463087246	Manager	10	0.14093959731543623
Vice President	32	0.2751677852348993	Vice President	137	0.1342281879194631
Vice President	68	0.2751677852348993	CEO	127	0.1342281879194631
Employee	27	0.2684563758389262	Employee	128	0.1342281879194631
CEO	129	0.2684563758389262	Vice President	32	0.1342281879194631
N/A	59	0.2684563758389262	N/A	141	0.1342281879194631
Vice President	51	0.2684563758389262	Vice President	23	0.12751677852348994

Table 25: Digraph degree and in-degree centrality results

Position	**ID**	**Out-degree**	**Position**	**ID**	**Closeness**
N/A	150	0.4966442953020134	N/A	150	0.6535087719298246
Employee	122	0.46308724832214765	Employee	122	0.6260504201680672
CEO	53	0.42953020134228187	CEO	53	0.6182572614107884
President	107	0.40268456375838924	President	107	0.5983935742971888
CEO	127	0.35570469798657717	CEO	127	0.5797665369649806
Employee	128	0.2684563758389262	Employee	73	0.5418181818181819
N/A	49	0.2684563758389262	Vice President	92	0.5283687943262412
Manager	17	0.2483221476510067	N/A	49	0.5265017667844523
Employee	73	0.2483221476510067	Manager	17	0.5209790209790209
Vice President	92	0.2080536912751678	Vice President	44	0.5209790209790209
Vice President	137	0.18791946308724833	Vice President	51	0.5068027210884354
N/A	59	0.18791946308724833	N/A	75	0.5050847457627119
Vice President	96	0.18791946308724833	N/A	48	0.5033783783783784
Vice President	44	0.18791946308724833	Vice President	96	0.5016835016835017
CEO	129	0.16778523489932887	CEO	129	0.5
N/A	48	0.16778523489932887	Vice President	38	0.49666666666666665
N/A	140	0.1610738255033557	Managing Director	65	0.49174917491749176
Vice President	78	0.1610738255033557	Vice President	78	0.4837662337662338
N/A	105	0.15436241610738255	Employee	128	0.48220064724919093
Vice President	38	0.15436241610738255	Vice President	69	0.48220064724919093
President	66	0.15436241610738255	N/A	139	0.4806451612903226
N/A	76	0.15436241610738255	N/A	79	0.4806451612903226
Vice President	51	0.1476510067114094	Vice President	126	0.4790996784565916
Director	101	0.1476510067114094	President	54	0.4790996784565916
Employee	61	0.1476510067114094	Vice President	137	0.476038338658147
Employee	9	0.1476510067114094	Vice President	23	0.476038338658147
N/A	139	0.14093959731543623	Vice President	32	0.476038338658147
N/A	33	0.14093959731543623	President	66	0.476038338658147
Vice President	32	0.14093959731543623	N/A	59	0.4745222929936306

Table 26: Digraph out-degree and closeness centrality results

Position	**ID**	**Betweenness**	**Position**	**ID**	**Eigenvector**
President	107	0.10076605522644952	N/A	67	0.5391047010625928
N/A	49	0.06970766699584406	N/A	50	0.45532201973661945
Manager	17	0.06009844962678258	N/A	48	0.3942517012642078
CEO	127	0.04812042737735486	N/A	13	0.35544760602602576
N/A	33	0.0464827946000284	Employee	9	0.30256656815580246
Employee	73	0.0458765540767956	N/A	147	0.26967775214165873
N/A	150	0.04339706334678396	N/A	57	0.12168590576547944
Employee	122	0.043036743753226636	Employee	27	0.11484421567917934
Vice President	92	0.04057671578347909	Employee	136	0.1101009937204844
N/A	135	0.037400982697570646	Employee	6	0.1061614267767172
Vice President	96	0.035132222282581685	President	107	0.03696942158036821
N/A	75	0.034560238742703836	N/A	75	0.029959483325828786
Employee	52	0.029868733280033147	N/A	33	0.020553407080967846
Employee	128	0.02681801013382451	Vice President	44	0.017863588032695655
N/A	139	0.026545374052426123	Employee	52	0.015990845082520355
Vice President	23	0.025982954754434393	Vice President	38	0.01472444832871569
Employee	86	0.02430040581946505	N/A	91	0.013033765763950758
Employee	149	0.02283005291743167	Vice President	78	0.012467377902678819
N/A	48	0.022143753840107923	Vice President	92	0.010740798051241058
Vice President	137	0.021852725292509662	Employee	142	0.00851342395099905
Vice President	44	0.021214955107970737	N/A	11	0.006303248910870136
Employee	114	0.02098511762772163	Vice President	69	0.006126932631440135
Trader	132	0.019170330300736265	President	54	0.00548486733374738
President	54	0.019090605295859202	Vice President	77	0.004671859410378506
Vice President	126	0.017975596242310963	Vice President	39	0.00463946737357979
Vice President	68	0.01755695848498713	Vice President	126	0.004409953458334775
Vice President	32	0.01647685912710031	Manager	17	0.004246902756002734
President	66	0.01634924619001559	Managing Director	65	0.004177535355094588
Director	1	0.01585235296943939	Employee	122	0.004153544111844599

Table 27: Digraph betweenness and eigenvector centrality results

Position	ID	Katz	Position	ID	Load
N/A	143	0.24237422850728338	President	107	0.09589345968071644
N/A	130	0.23537953089929822	N/A	49	0.06562775891450512
Trader	134	0.23321842855683123	Manager	17	0.0549235689618194
Trader	133	0.18490272073304684	CEO	127	0.04864510544420742
Trader	131	0.14847560114705086	N/A	33	0.047666240371718516
Trader	132	0.14216395037258336	Employee	73	0.045360911999927846
Director	1	0.13665227360950677	Employee	122	0.04369300543857222
Employee	85	0.0957221897350964	N/A	150	0.04195730869993218
Director	97	0.08754325544868294	Vice President	92	0.040891667571169146
Director	26	0.08672008514228506	N/A	135	0.039554582574569314
N/A	138	0.07966115848806787	N/A	75	0.034626941887536136
Employee	99	0.05958095358628763	Vice President	96	0.034147323326961784
N/A	56	0.04908151246885697	Employee	52	0.031556689666421986
Employee	136	0.03825420144836524	N/A	139	0.02895107162110891
Employee	25	0.03438250355928042	Employee	128	0.026798206207182377
Director	120	0.032164274331988985	Vice President	23	0.02567056036183881
Director	102	0.02861180884487637	Vice President	38	0.02484820543422453
N/A	135	0.02816877309328466	Employee	86	0.02442509542637905
President	54	0.026407817729662837	Employee	149	0.02372649680414163
N/A	94	0.02484048406979067	N/A	48	0.022691642798503612
Employee	5	0.020932518538827365	Vice President	137	0.021934170297805783
Manager	119	0.020789502185213124	Vice President	44	0.020807063001073236
Vice President	39	0.019834819908128336	President	54	0.018735221321616646
Manager	123	0.01779256550599657	Vice President	68	0.018317786395246102
N/A	41	0.017721429711231716	Trader	132	0.017923264098955963
Manager	4	0.017238968657953928	Vice President	32	0.017370598241606642
Director	106	0.016686015104044077	Vice President	126	0.016659037514068165
CEO	46	0.016498568546175364	President	66	0.015940664353338387
N/A	89	0.013826167717827973	Manager	118	0.014773718438079538

Table 28: Directed katz and load centrality results

D Graph construction results

These (di)graphs was produced by Gephi[1]. The algorithms used to generate them was the *Yifan Hu* one time, followed by *Expansion* times three. Expansion was added as Yifan Hu algortihm could produce graphs with vertices close to each other or overlaying. Expansion, as the name suggests, will pull each vertex from the rest with an equal distance, so to perserve the Yifan Hu organisation. This appendix only contain 24 k-NN graphs, where $k = \{1, 2, 3\}$ for the distance measures *Braycurtis*, *Canberra*, *Chebyshev*, *Cityblock*, *Correlation*, *Cosine*, *Euclidean* and *Hamming*.

[1]`https://gephi.org`

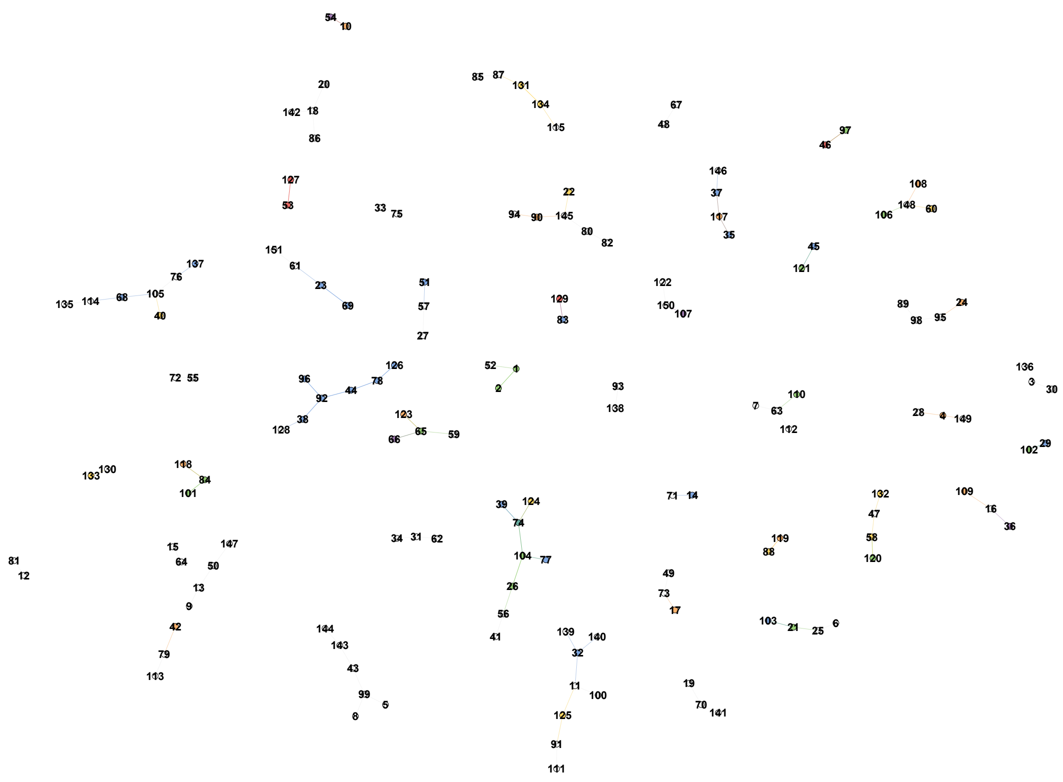

Figure 48: Graph k-NN Braycurtis where $k = 1$

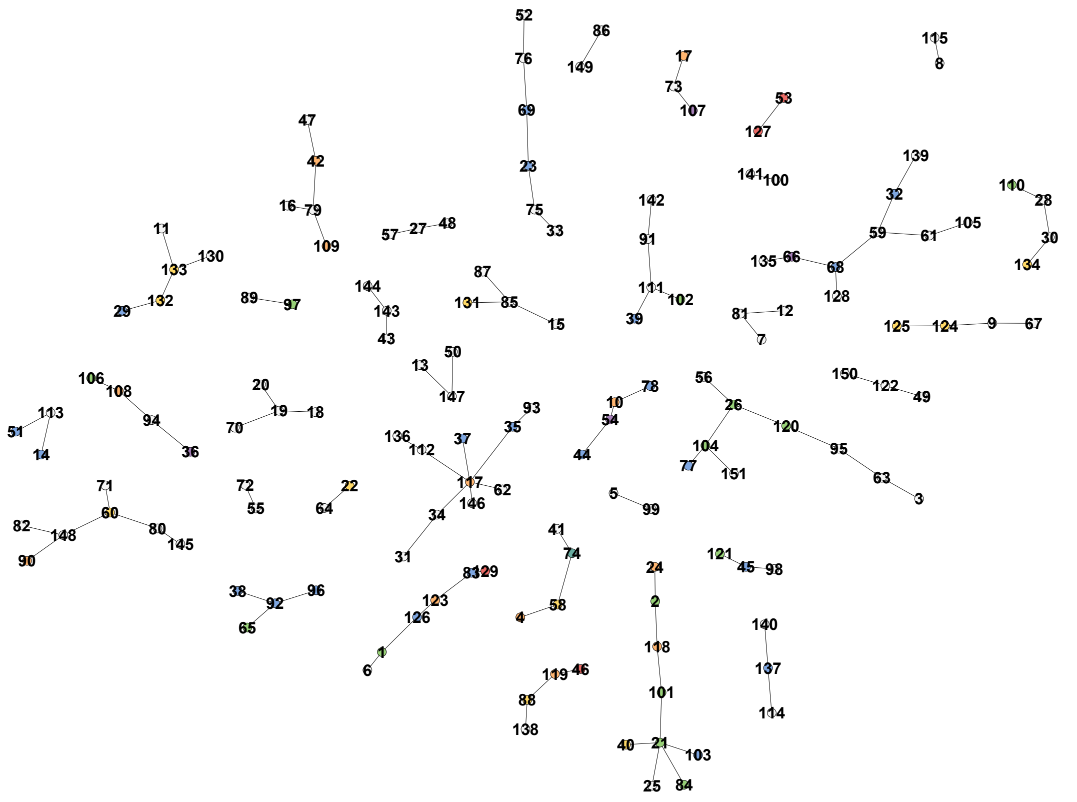

Figure 49: Graph k-NN Canberra where k = 1

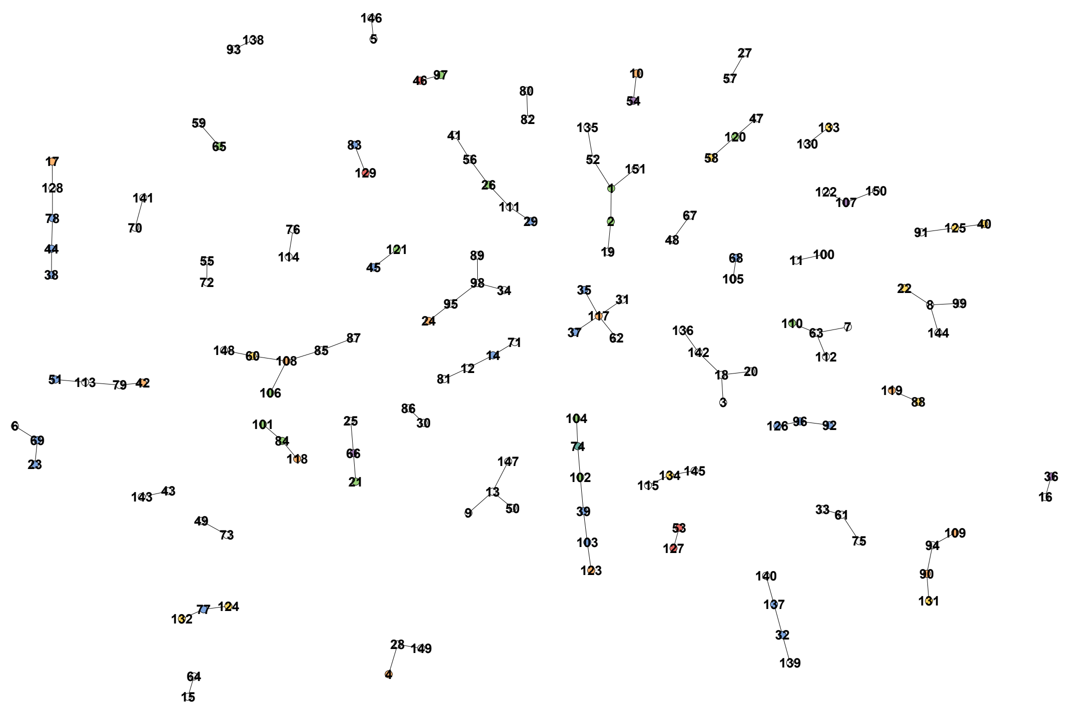

Figure 50: Graph k-NN Chebyshev where $k = 1$

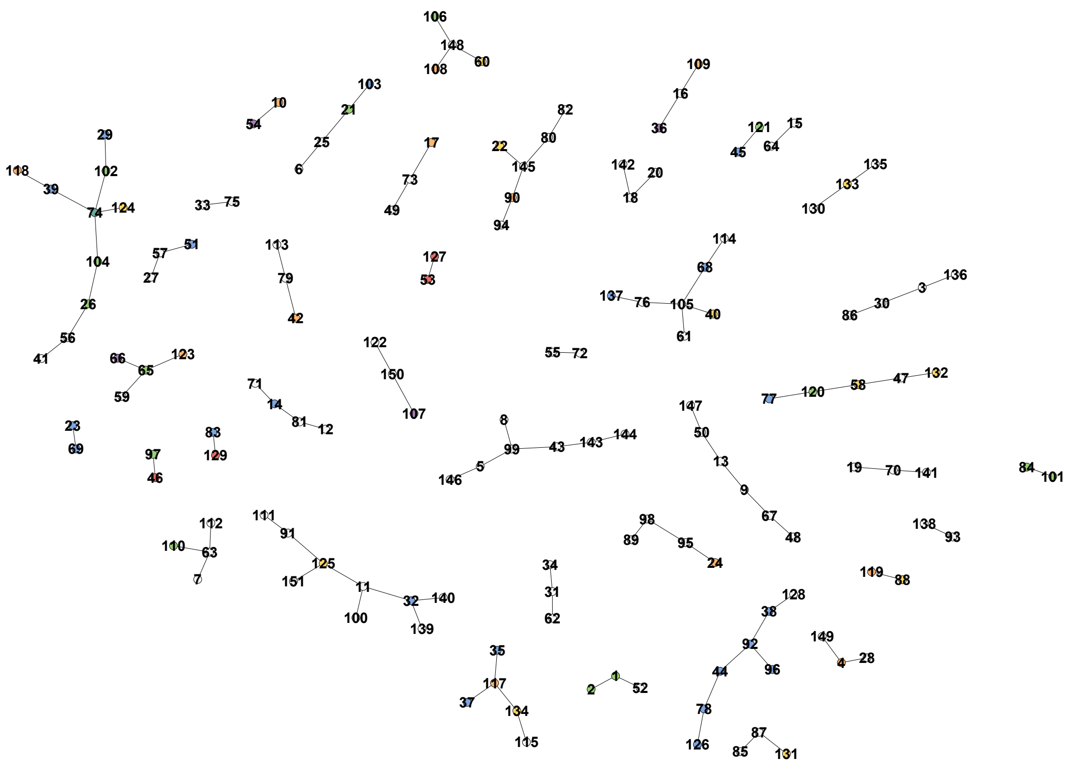

Figure 51: Graph k-NN Cityblock where k = 1

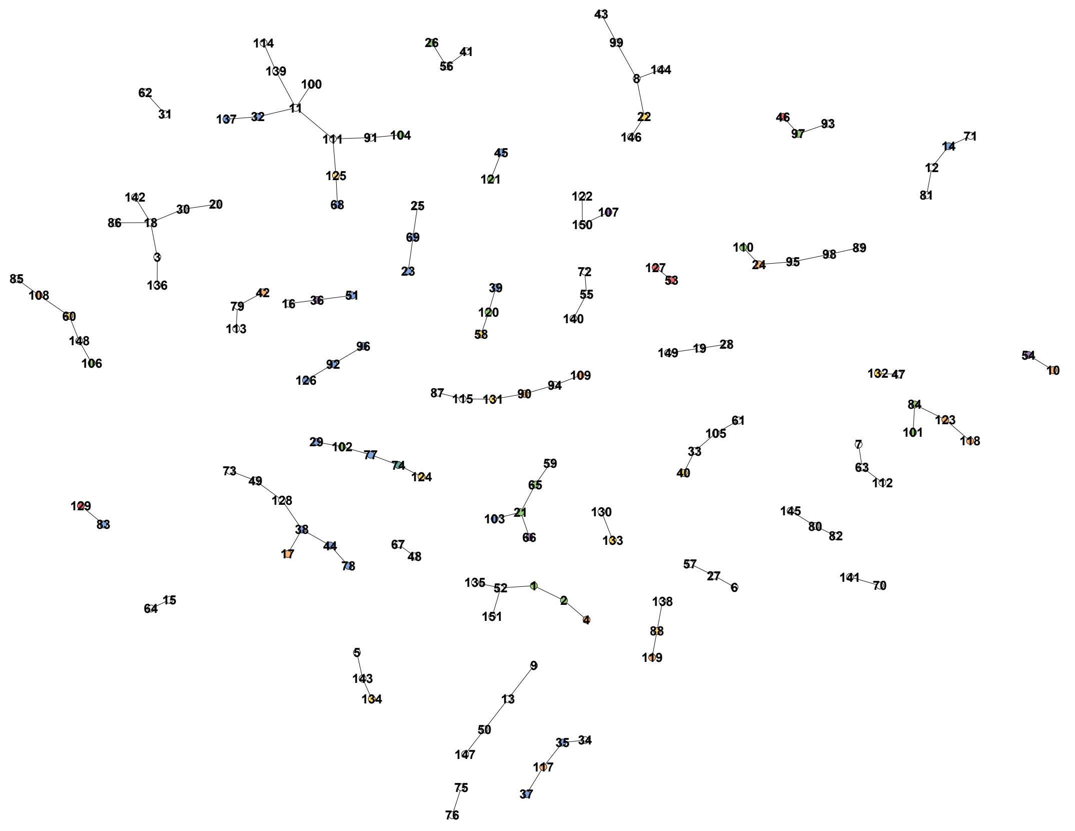

Figure 52: Graph k-NN Correlation where $k = 1$

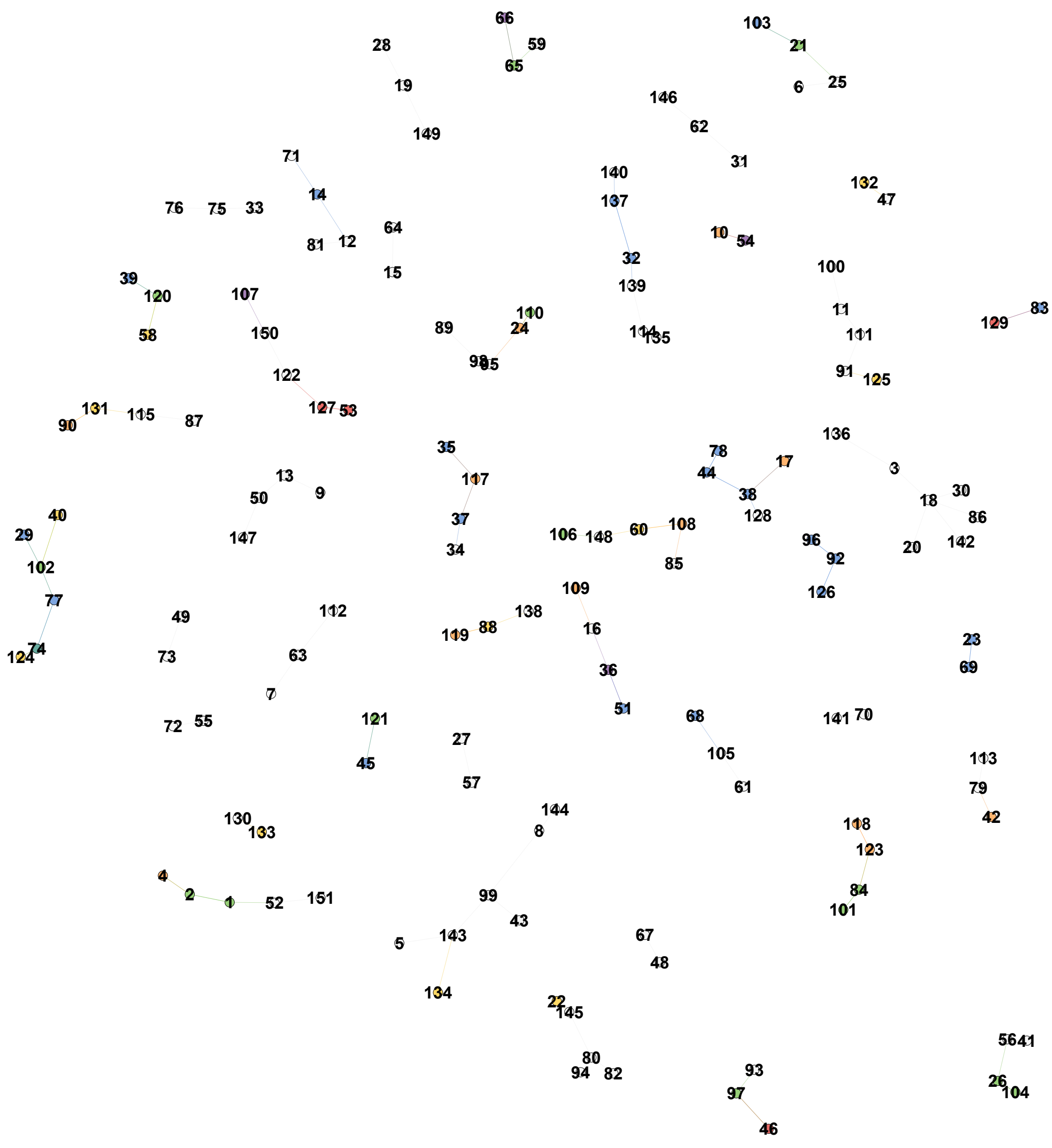

Figure 53: Graph k-NN Cosine where k = 1

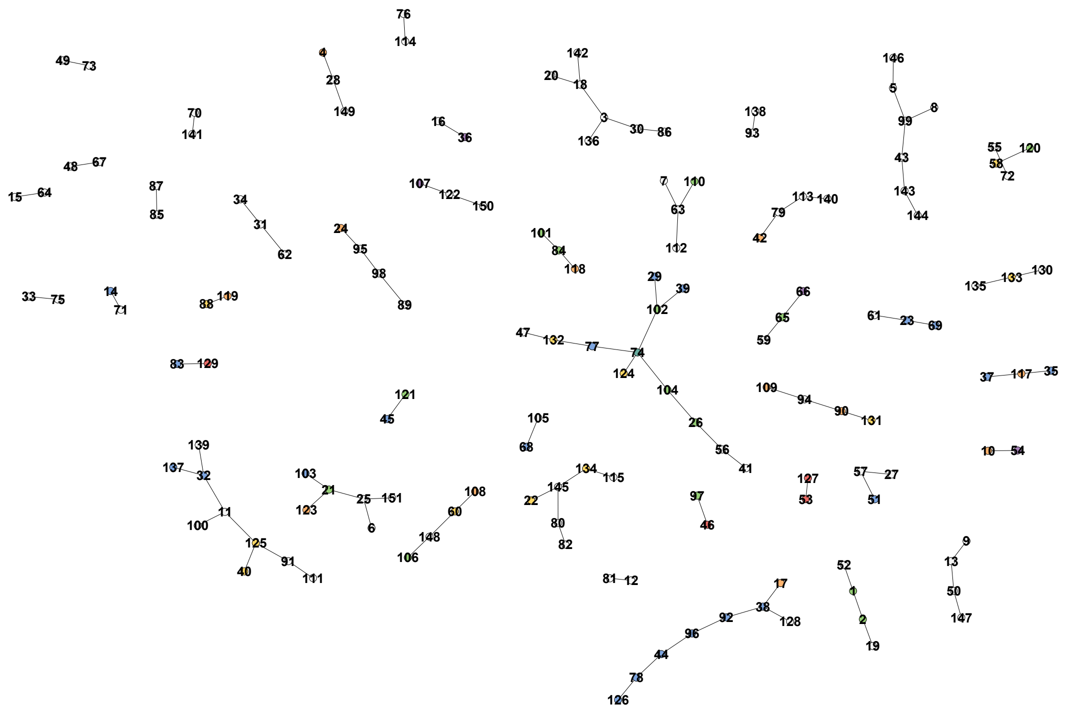

Figure 54: Graph k-NN Euclidean where k = 1

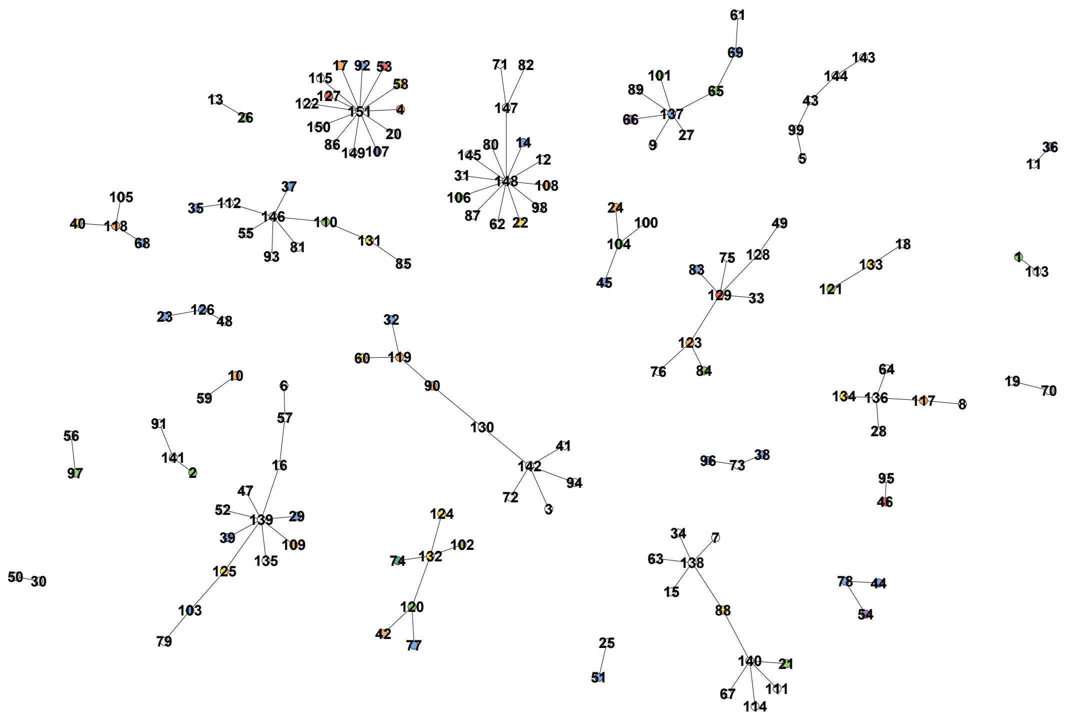

Figure 55: Graph k-NN Hamming where $k = 1$

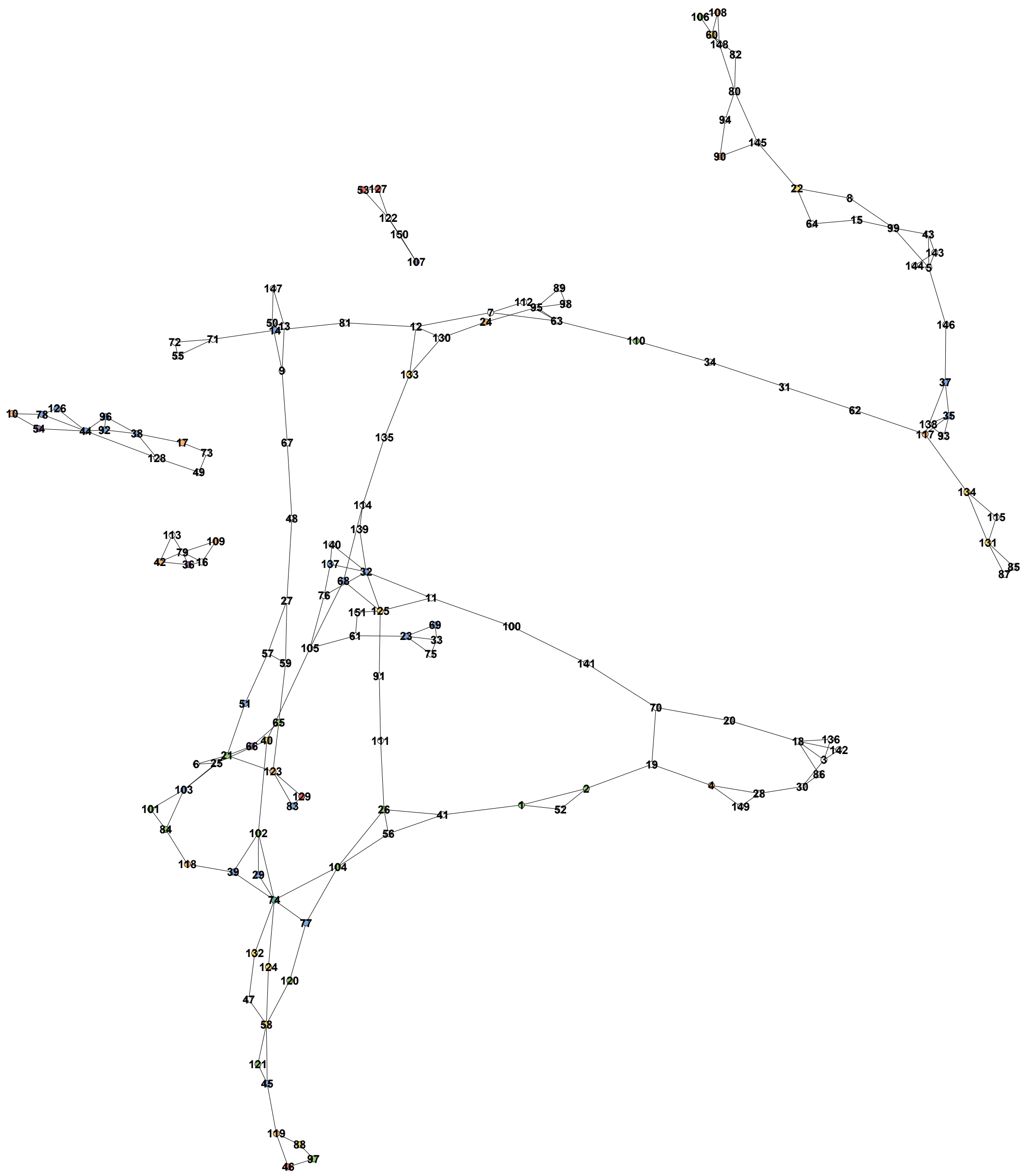

Figure 56: Graph k-NN Braycurtis where $k = 2$

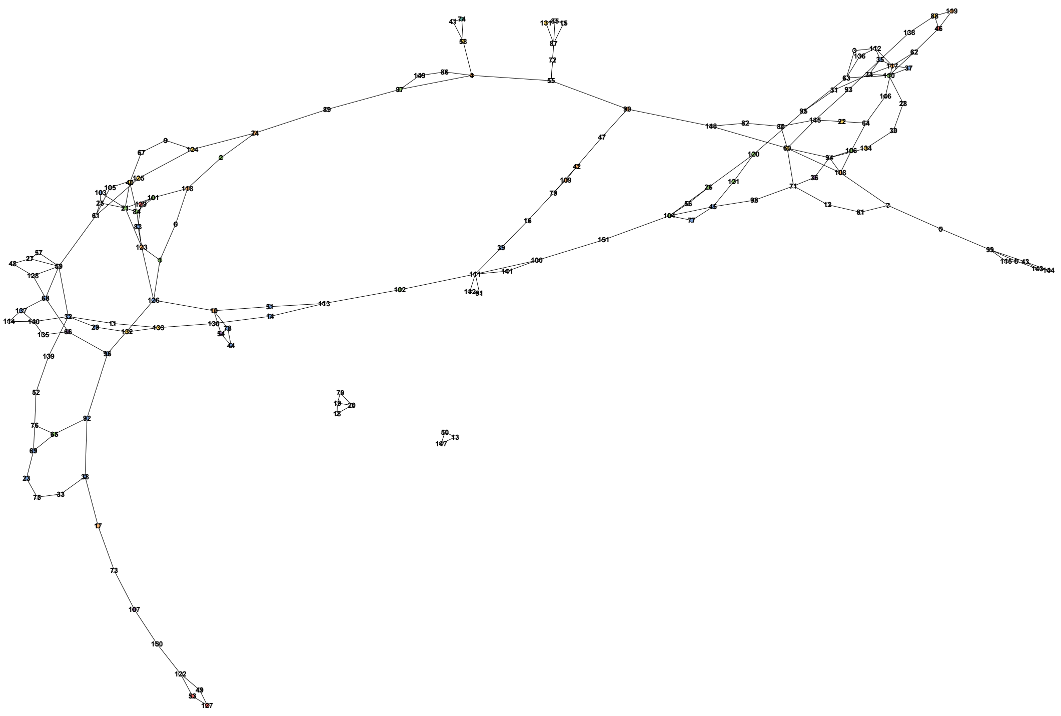

Figure 57: Graph k-NN Canberra where $k = 2$

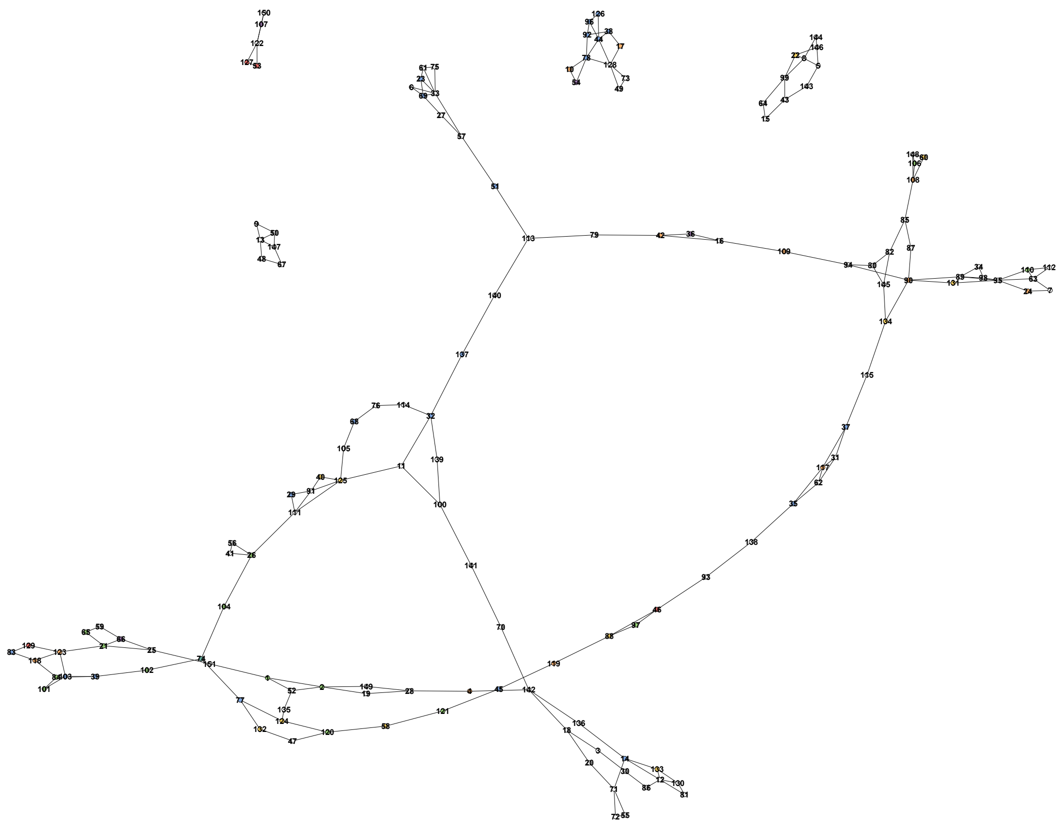

Figure 58: Graph k-NN Chebyshev where k = 2

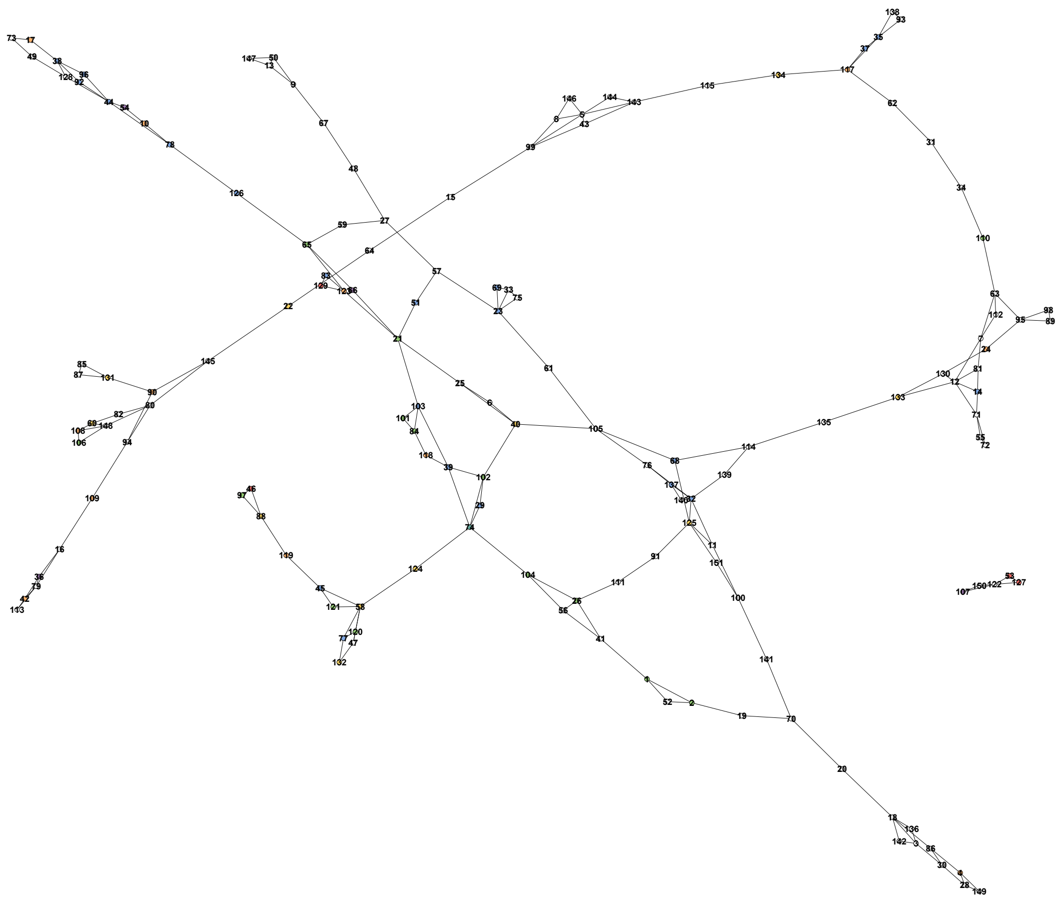

Figure 59: Graph k-NN Cityblock where k = 2

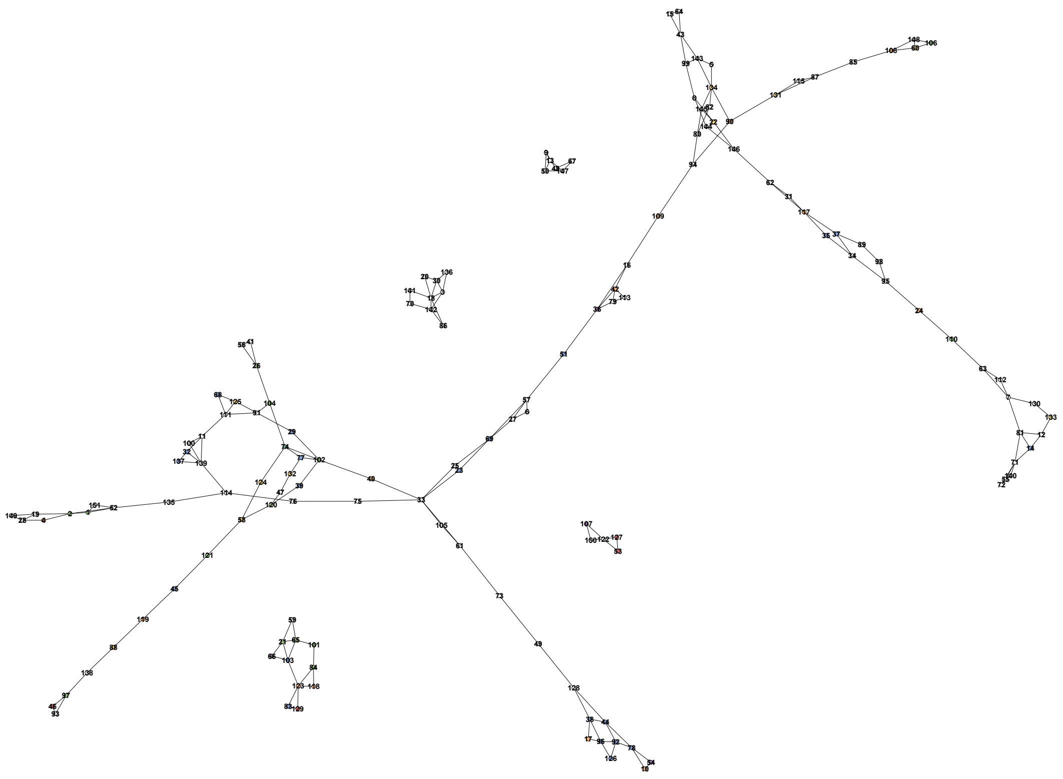

Figure 60: Graph k-NN Correlation where k = 2

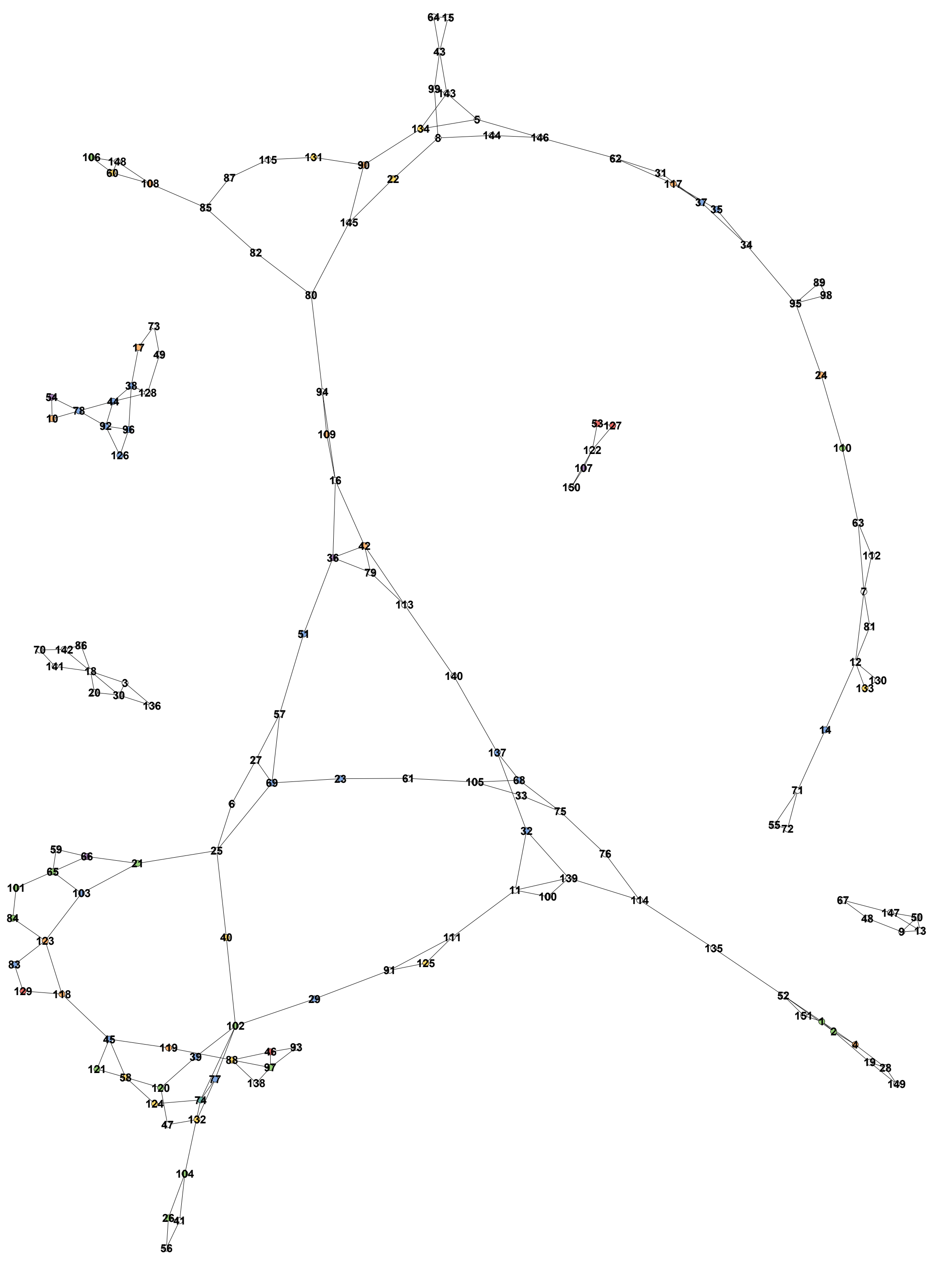

Figure 61: Graph k-NN Cosine where k = 2

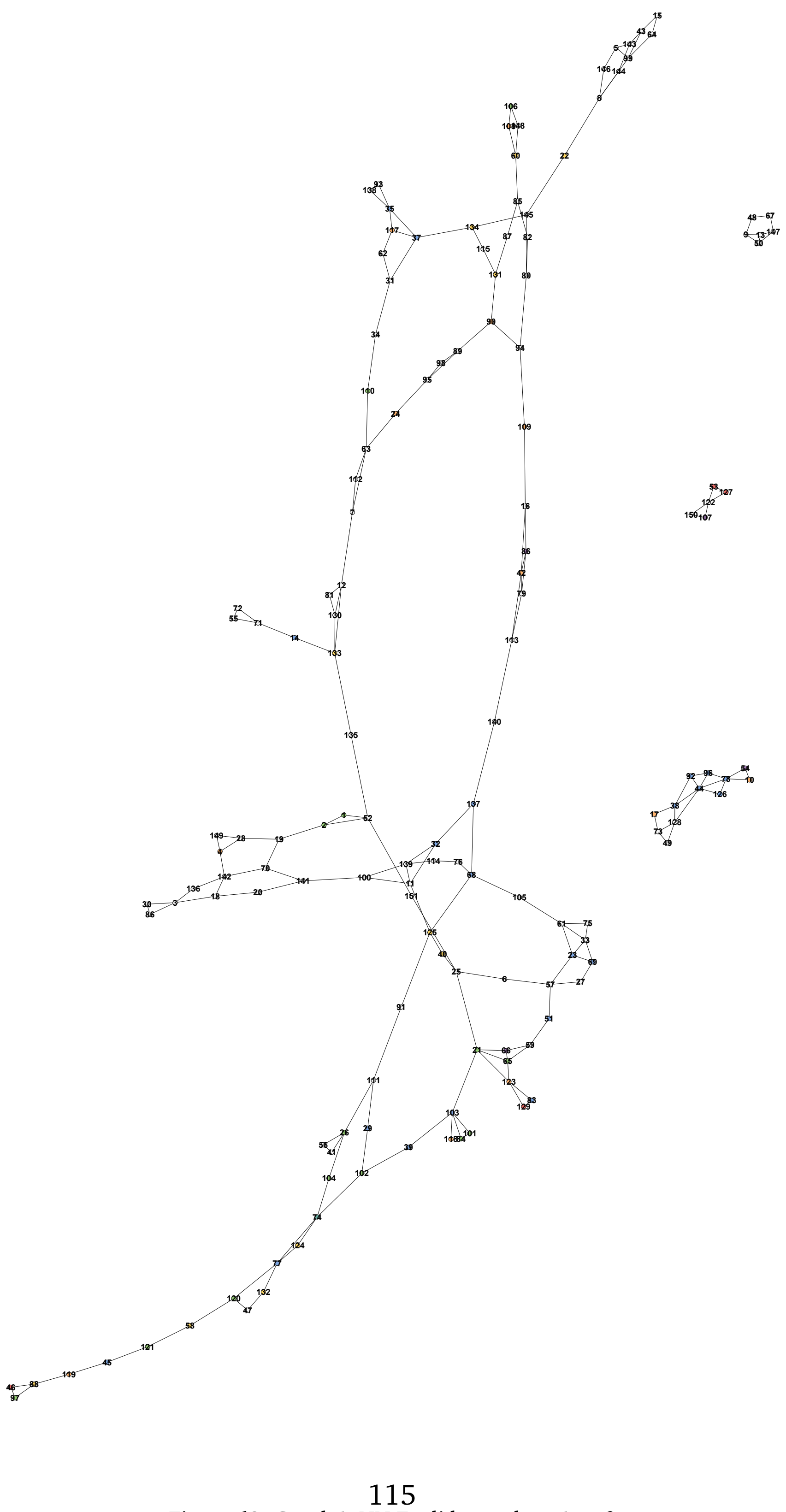

Figure 62: Graph k-NN Euclidean where $k = 2$

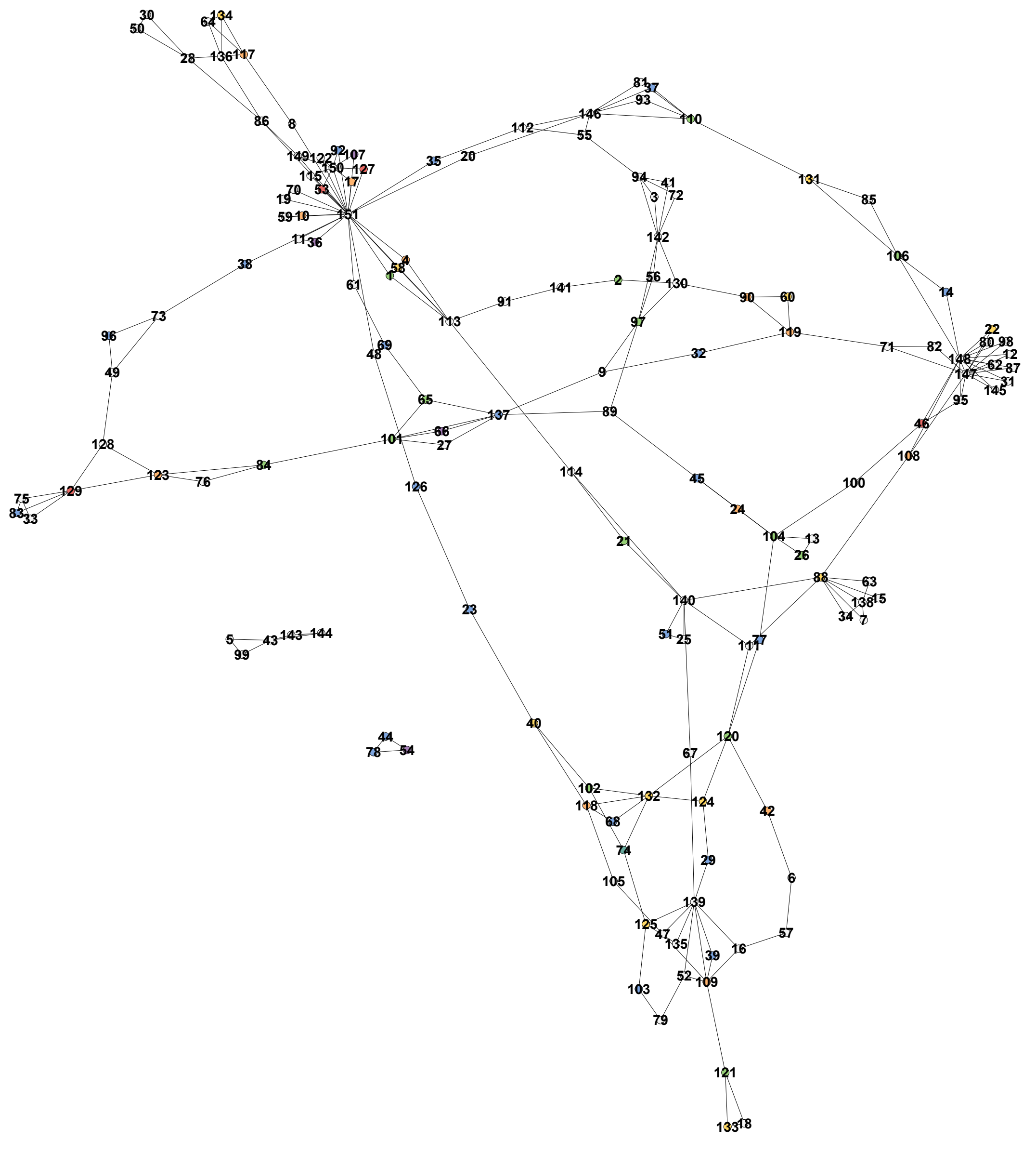

Figure 63: Graph k-NN Hamming where $k = 2$

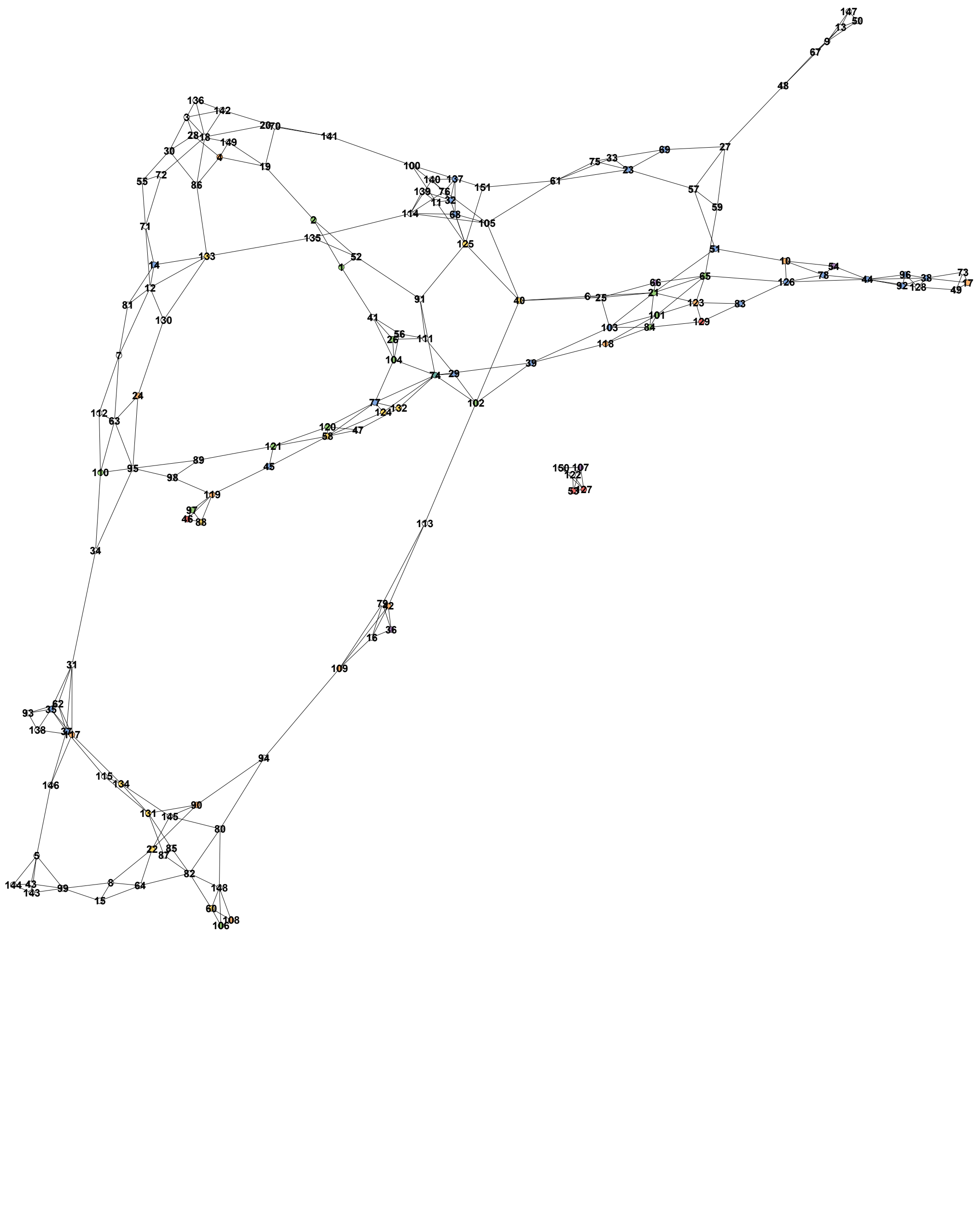

Figure 64: Graph k-NN Braycurtis where $k = 3$

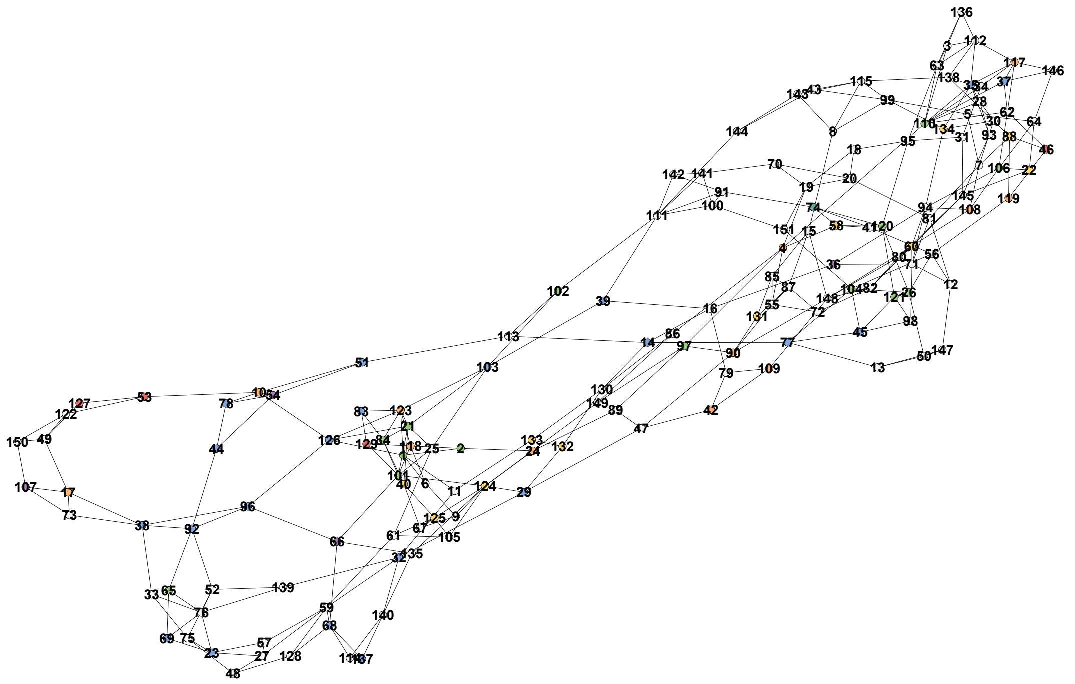

Figure 65: Graph k-NN Canberra where $k = 3$

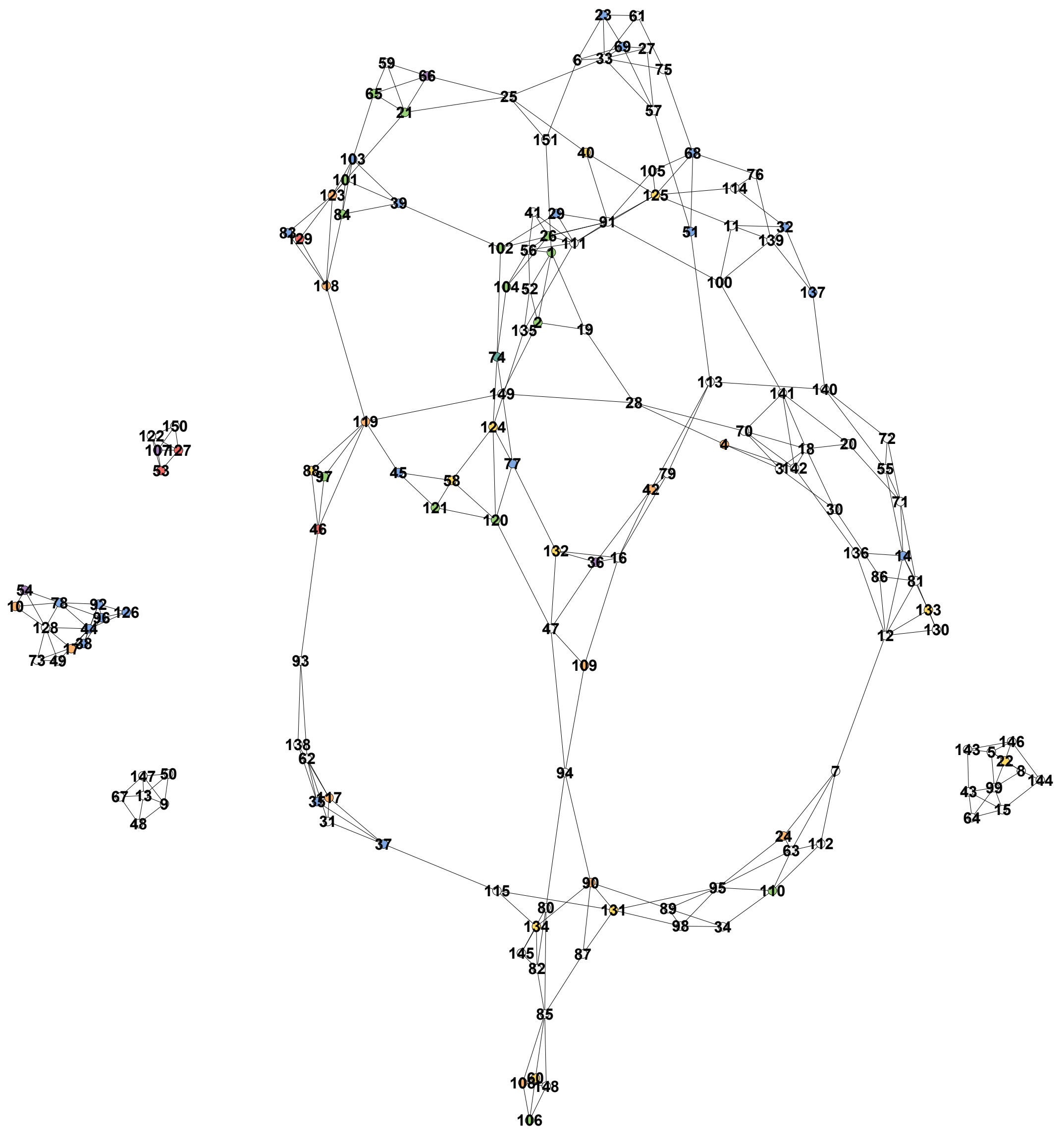

Figure 66: Graph k-NN Chebyshev where k = 3

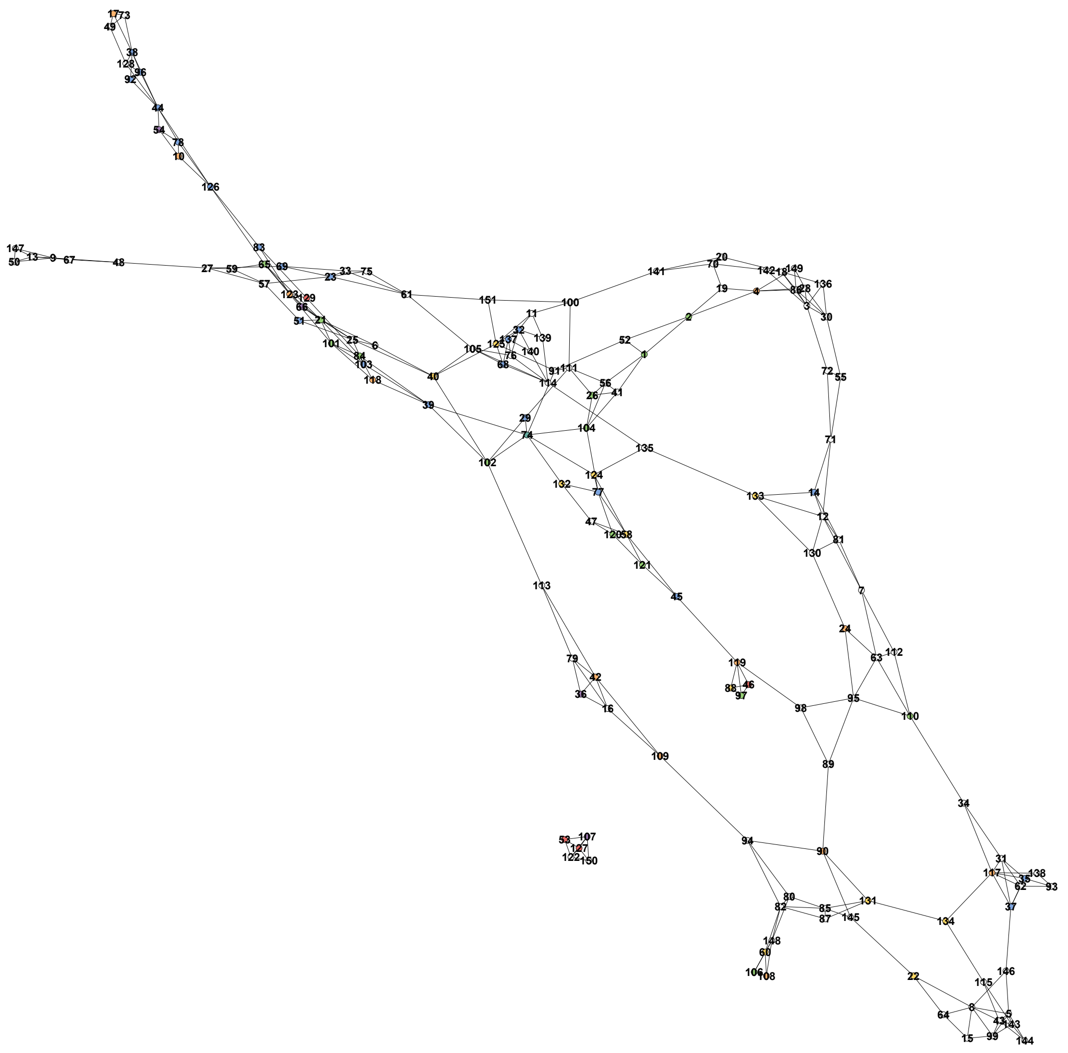

Figure 67: Graph k-NN Cityblock where k = 3

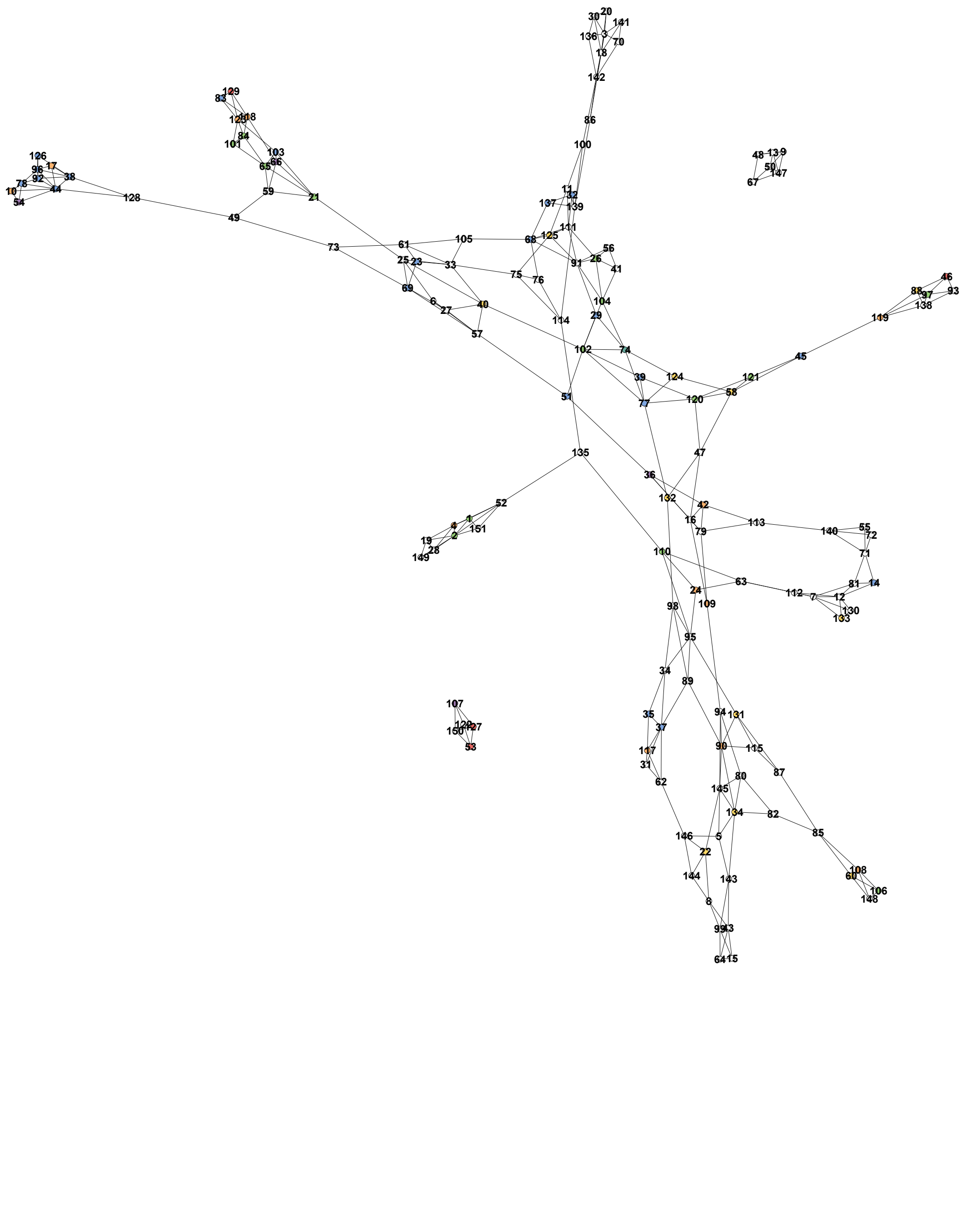

Figure 68: Graph k-NN Correlation where $k = 3$

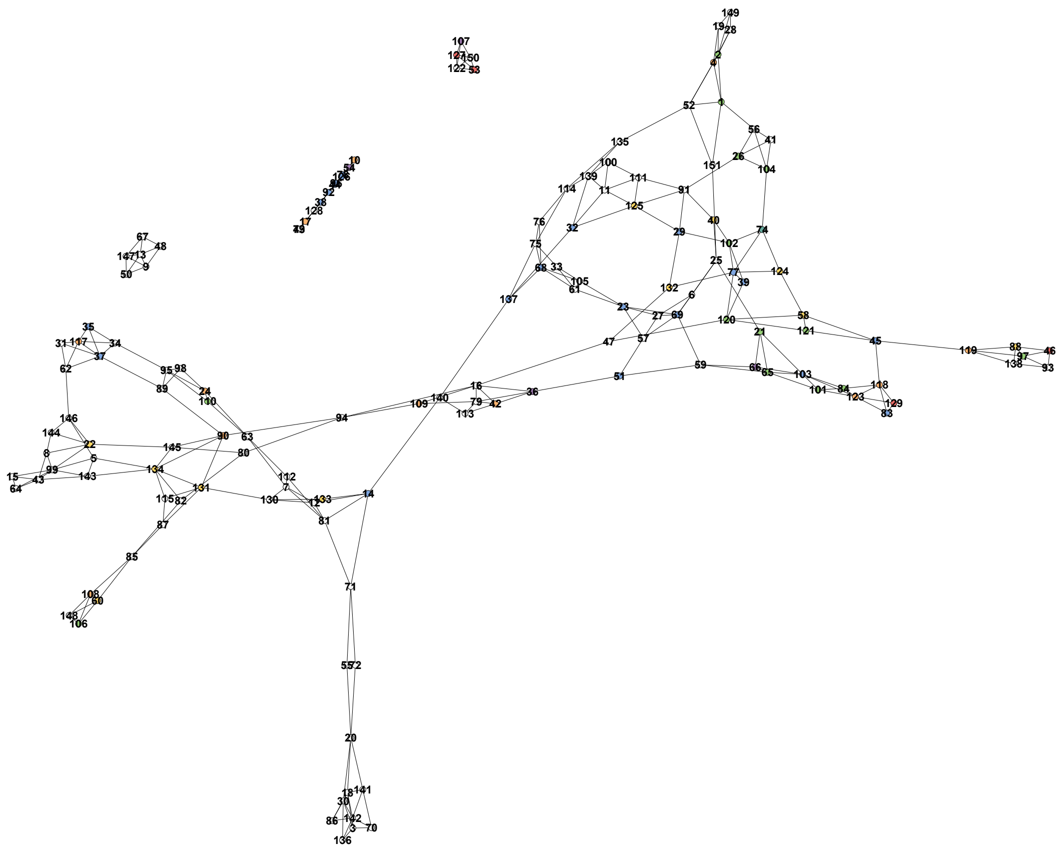

Figure 69: Graph k-NN Cosine where k = 3

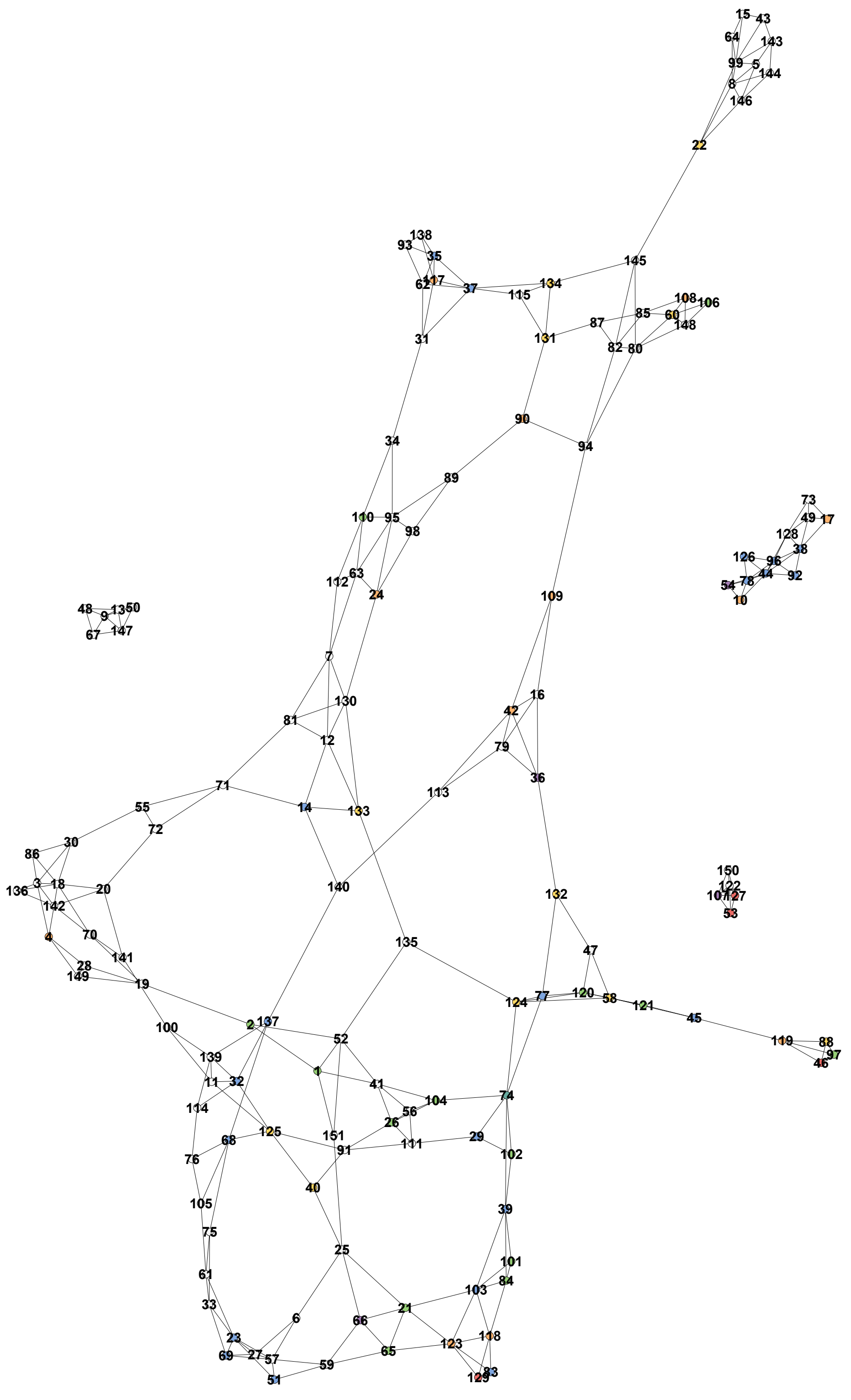

Figure 70: Graph k-NN Euclidean where $k = 3$

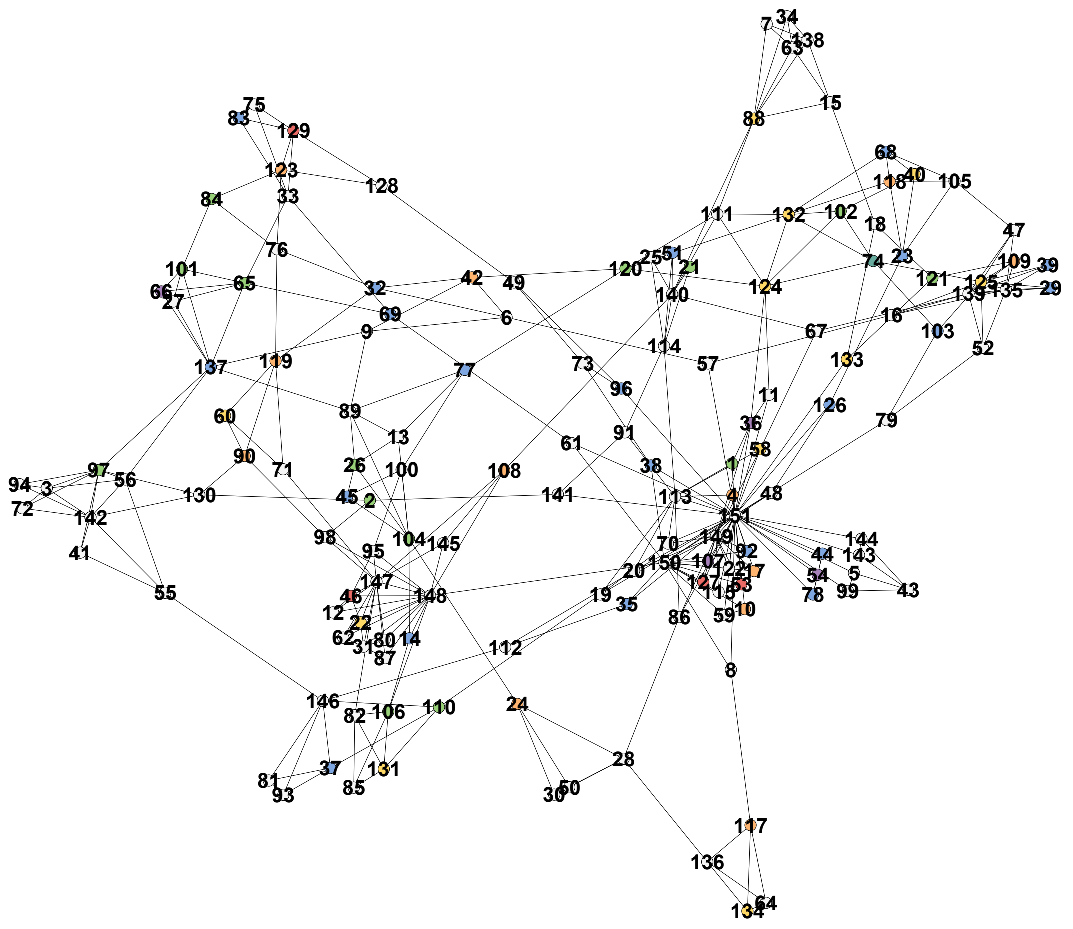

Figure 71: Graph k-NN Hamming where k = 3

E SQL table export

```python
#!/usr/bin/env python

import MySQLdb
import networkx as nx

'''
    'employee_position.csv' contains a list with employeelist eid
    and the employee position found in thesis Appendix A, e.g.:
    1,Director
    2,Director
    3,Employee
    4,Manager
    5,Employee
    ...
'''

def main():
    '''
    Job titles and their node colours. Some job titles have the same colour.
    They are "Director" = "Managing Director" and "Employee" = "N/A".
    '''
  colors = {
    'CEO':{'color':{'r':234, 'g':107, 'b':102, 'a':0}}, # Red
    'President':{'color':{'r':166, 'g':128, 'b':184, 'a':0}}, # Purple
    'Vice President':{'color':{'r':126, 'g':166, 'b':224, 'a':0}}, # Blue
    'Director':{'color':{'r':151, 'g':208, 'b':119, 'a':0}}, # Green
    'In-House Lawyer':{'color':{'r':103, 'g':171, 'b':159, 'a':0}}, # Blue-green
    'Manager':{'color':{'r':255, 'g':181, 'b':112, 'a':0}}, # Orange
    'Trader':{'color':{'r':255, 'g':217, 'b':102, 'a':0}}, # Yellow
    'Specialist':{'color':{'r':230, 'g':208, 'b':222, 'a':0}}, # Light purple
    'Analyst':{'color':{'r':212, 'g':225, 'b':245, 'a':0}}, # Light blue
    'Employee':{'color':{'r':245, 'g':245, 'b':245, 'a':0}}, # Gray
    'Managing Director':{'color':{'r':151, 'g':208, 'b':119, 'a':0}}, # Green
    'N/A':{'color':{'r':245, 'g':245, 'b':245, 'a':0}}, # Gray
  }
  positions = {}
  enron_email = {}
  undirected = nx.Graph()
  directed = nx.DiGraph()

  with open('employee_position.csv', 'r') as in_file:
        '''
        Generate a list over Enron job positions
        '''
    for line in in_file.readlines():
      line = line.strip().split(',')
      positions[int(line[0])] = line[1]
  try:
    connection = MySQLdb.connect('localhost', 'root', 'password', 'enron')
    cur = connection.cursor()
    query = 'SELECT eid, Email_id FROM employeelist ORDER BY eid'
    cur.execute(query)
    employees = cur.fetchall()
    for employee in employees:
      if not employee[0] in undirected.nodes():
        undirected.add_node(employee[0])
        undirected.node[employee[0]]['label'] = employee[1]
        undirected.node[employee[0]]['viz'] = colors[positions[employee[0]]]
      if not employee[0] in directed.nodes():
        directed.add_node(employee[0])
        directed.node[employee[0]]['label'] = employee[1]
        directed.node[employee[0]]['viz'] = colors[positions[employee[0]]]
      if not employee[1] in enron_email:
        enron_email[employee[1]] = employee[0]
    for n in undirected.nodes():
      query = 'SELECT rtype, rvalue FROM recipientinfo WHERE mid IN (SELECT mid FROM message WHERE sender LIKE \'%s\') AND rvalue IN (SELECT Email_id FROM employeelist)' % (undirected.node[n]['label'])
      cur.execute(query)
      recipients = cur.fetchall()
      for recipient in recipients:
        # recipient[0] = TO, CC, BCC
        # recipient[1] = recipient email address
        if not undirected.has_edge(n, enron_email[recipient[1]]):
          undirected.add_edge(n, enron_email[recipient[1]], weight = 0)
        if not directed.has_edge(n, enron_email[recipient[1]]):
          directed.add_edge(n, enron_email[recipient[1]], weight = 0)
        undirected[n][enron_email[recipient[1]]]['weight'] += 1
        directed[n][enron_email[recipient[1]]]['weight'] += 1
      print 'Completed Enron employee eid %s' % (n)
  except KeyboardInterrupt:
        '''
        Uncomment exit(0) to allow ctrl-c to stop exection and continues to
        write partial exported SQL information to files. This was used for
        testing.
        '''
    exit(0)
  except MySQLdb.Error as e:
    print 'Error %d: %s' % (e.args[0], e.args[1])
    exit(0)
  finally:
    if connection:
      connection.close()
  print 'Undirected (graph) # of vertices %s and edges %s' % (len(undirected.nodes()), len(undirected.edges()))
  print 'Directed (digraph) # of vertices %s and edges %s' % (len(directed.nodes()), len(directed.edges()))
  nx.write_gexf(undirected, 'undirected_colour.gexf', version = '1.2draft')
  nx.write_gexf(directed, 'directed_colour.gexf', version = '1.2draft')
  print 'Written graph and digraph to file'

if __name__ == '__main__':
  main()
```

Listing E.1: sqlexp.py

F Enron social network analysis

```
#!/usr/bin/env python

import networkx as nx

undir_dataset = {}
dir_dataset = {}
undir_file = 'undirected_colour.gexf'
dir_file = 'directed_colour.gexf'
length = 30

def write_dataset_to_file(graph):
  dataset = dir_dataset if graph.is_directed() else undir_dataset
  filename = dir_file if graph.is_directed() else undir_file
  filename = filename.replace('colour.gexf', 'features.csv')
  with open(filename, 'w') as infile:
    for actor in dataset:
      row = str(actor) + ','
      length = len(dataset[actor])
      for i in range(length):
        row += str(dataset[actor][i])
        if i < length - 1:
          row += ','
      infile.write('%s\n' % (row))

def add_tuple_to_dataset(dataset, tpl):
  for item in tpl:
    if item not in dataset:
      dataset[item] = []
    dataset[item].append(tpl[item])

def prepare_graph(graph):
  print 'Read %s with %s nodes and %s edges' % \
    (dir_file if graph.is_directed() else undir_file, \
    graph.number_of_nodes(), graph.number_of_edges())
  # Remove self-loops because they are uninteresting for SNA
  remove_selfloop_edges = graph.selfloop_edges()
  if remove_selfloop_edges:
    print 'Found %s self-loops, removing them!' % \
      (len(remove_selfloop_edges))
    graph.remove_edges_from(remove_selfloop_edges)
    print 'Graph have %s nodes and %s edges' % \
      (graph.number_of_nodes(), graph.number_of_edges())
  # Also remove isolates from the graph
  node_isolates = nx.isolates(graph)
  if node_isolates:
    print 'Graph is not connected, removing isolates: %s' % (node_isolates)
    graph.remove_nodes_from(node_isolates)
    print 'Graph have %s nodes and %s edges' % (graph.number_of_nodes(), graph.
    number_of_edges())
  return graph

def preserve_graph_colors(graph):
  dataset = dir_dataset if graph.is_directed() else undir_dataset
  for node in graph.nodes():
    n = graph.node[node]
    if node not in dataset:
      dataset[node] = []
    dataset[node].append(n['label'])
    dataset[node].append(n['viz']['color']['a'])
    dataset[node].append(n['viz']['color']['r'])
    dataset[node].append(n['viz']['color']['b'])
    dataset[node].append(n['viz']['color']['g'])

def social_network_analysis(graph):
  func = lambda x: x[1]
  dataset = dir_dataset if graph.is_directed() else undir_dataset
  if graph.is_directed():
    print 'Strongly connected: %s' % (nx.is_strongly_connected(graph))
    print 'Weakly connected: %s' % (nx.is_weakly_connected(graph))
    print 'Number of weakly components: %s' % (nx.number_weakly_connected_components(
    graph))
  if not graph.is_directed():
    print 'Diameter: %s' % (nx.diameter(graph))
    print 'Periphery: %s' % (nx.periphery(graph))
    print 'Radius: %s' % (nx.radius(graph))
  print 'Degree centrality: %s' % \
    sorted(nx.degree_centrality(graph).items(), key=func, reverse=True)[:length]
  add_tuple_to_dataset(dataset, nx.degree_centrality(graph))
  if graph.is_directed():
    print 'Indegree centrality: %s' % \
      sorted(nx.in_degree_centrality(graph).items(), key=func, reverse=True)[:length]
    add_tuple_to_dataset(dataset, nx.in_degree_centrality(graph))
    print 'Outdegree centrality: %s' % \
      sorted(nx.out_degree_centrality(graph).items(), key=func, reverse=True)[:length]
    add_tuple_to_dataset(dataset, nx.out_degree_centrality(graph))
  print 'Closeness centrality: %s' % \
    sorted(nx.closeness_centrality(graph).items(), key=func, reverse=True)[:length]
  add_tuple_to_dataset(dataset, nx.closeness_centrality(graph))
  print 'Betweenness centrality: %s' % \
    sorted(nx.betweenness_centrality(graph).items(), key=func, reverse=True)[:length]
  add_tuple_to_dataset(dataset, nx.betweenness_centrality(graph))
  print 'Eigenvector centrality: %s' % \
    sorted(nx.eigenvector_centrality_numpy(graph).items(), key=func, reverse=True)[:
    length]
  add_tuple_to_dataset(dataset, nx.eigenvector_centrality_numpy(graph))
  print 'Katz centrality: %s' % \
    sorted(nx.katz_centrality_numpy(graph).items(), key=func, reverse=True)[:length]
  add_tuple_to_dataset(dataset, nx.katz_centrality_numpy(graph))
  print 'Load centrality: %s' % \
    sorted(nx.load_centrality(graph).items(), key=func, reverse=True)[:length]
  add_tuple_to_dataset(dataset, nx.load_centrality(graph))
  if not graph.is_directed():
    print 'Current flow closeness centrality: %s' % \
      sorted(nx.current_flow_closeness_centrality(graph).items(), key=func, reverse=
    True)[:length]
    add_tuple_to_dataset(dataset, nx.current_flow_closeness_centrality(graph))
    print 'Current flow betweenness centrality: %s' % \
      sorted(nx.current_flow_betweenness_centrality(graph).items(), key=func, reverse=
    True)[:length]
    add_tuple_to_dataset(dataset, nx.current_flow_betweenness_centrality(graph))
    print 'Approximate current flow betweenness centrality: %s' % \
      sorted(nx.approximate_current_flow_betweenness_centrality(graph).items(), key=
    func, reverse=True)[:length]
    add_tuple_to_dataset(dataset, nx.approximate_current_flow_betweenness_centrality(
    graph))
    print 'Communicability betweenness centrality: %s' % \
      sorted(nx.communicability_betweenness_centrality(graph).items(), key=func,
    reverse=True)[:length]
    add_tuple_to_dataset(dataset, nx.communicability_betweenness_centrality(graph))

def main():
  graphs = []
  graphs.append(nx.read_gexf(undir_file))
  graphs.append(nx.read_gexf(dir_file))

  for graph in graphs:
    graph = prepare_graph(graph)
    preserve_graph_colors(graph)
    social_network_analysis(graph)
    write_dataset_to_file(graph)

if __name__ == '__main__':
  main()
```

Listing F.1: sna.py

G Graph construction

```
#!/usr/bin/env python

from scipy.spatial import distance
import networkx as nx
import numpy as np

undir_graph = nx.Graph()
dir_graph = nx.DiGraph()
neighbors = [1, 2, 3]
epsilon = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06]
removed_nodes = [0, 116]
weighted = False

def k_nearest_neighbors(k, dataset, graph, dist):
    matrix = []
    # Construct distances for the complete graph
    for i in range(152):
        matrix.append([])
        # Some distance measures did not like all null array/list
        if not i in removed_nodes:
            for j in range(152):
                if not j in removed_nodes:
                    matrix[i].append(dist(dataset[i], dataset[j]))
                else:
                    matrix[i].append(0.0)

    # Check the distance to find the closest k neighbors
    for i in range(152):
        if not i in removed_nodes:
            iknn = [] # Index
            knn = [] # Weight (calculated by distance measure)
            removed_nodes.append(i)
            for j in range(152):
                # Avoid self-loops and removed nodes from previous Python script
                if not j in removed_nodes:
                    iknn.append(j)
                    knn.append(matrix[i][j])
                    selection_sort(knn, iknn)
                    iknn = iknn[:k]
                    knn = knn[:k]
            removed_nodes.pop()
            for j in range(len(iknn)):
                if weighted:
                    graph.add_edge(i, iknn[j], weight = knn[j])
                else:
                    graph.add_edge(i, iknn[j])

    write_graph_to_file(graph, k, 'k_nearest_neighbors', dist)

def e_neighborhood(e, dataset, graph, dist):
    matrix = []
    # Construct distances for the complete graph
    for i in range(152):
        matrix.append([])
        for j in range(152):
            matrix[i].append(dist(dataset[i], dataset[j]))
    # Check the distance to find neighbors within e
    for i in range(152):
        # Skip isolates and invalid node indexes
        if not i in removed_nodes:
            removed_nodes.append(i)
            for j in range(152):
                # Avoid self-loops, isolates and invalid indexes
                if not j in removed_nodes:
                    d = dist(dataset[i], dataset[j])
                    if d < e:
                        if weighted:
                            graph.add_edge(i, j, weight = d)
                        else:
                            graph.add_edge(i, j)
            removed_nodes.pop()
    write_graph_to_file(graph, e, 'epsilon_neighbors', dist)

def write_graph_to_file(graph, iteration, construction, dist):
    fname = 'directed_' if graph.is_directed() else 'undirected_'
    w = '_weighted' if weighted else ''
    fname += '%s_%s_%s%s.gexf' % (iteration, construction, dist.__name__, w)
    print 'Generated graph: %s' % (fname)
    nx.write_gexf(graph, fname, version = "1.2draft")
    graph.remove_edges_from(graph.edges())

def selection_sort(alist, blist):
   for i in range(len(alist)-1,0,-1):
       pos_max=0
       for j in range(1, i+1):
           if alist[j] > alist[pos_max]:
               pos_max = j
       temp = alist[i]
       alist[i] = alist[pos_max]
       alist[pos_max] = temp
       temp = blist[i]
       blist[i] = blist[pos_max]
       blist[pos_max] = temp

def minmax(dataset, dist_measures):
    for dist in dist_measures:
        minimum = 100
        maximum = 0
        for i in range(len(dataset)):
            if i not in removed_nodes:
                removed_nodes.append(i)
                for j in range(len(dataset)):
                    if j not in removed_nodes:
                        d = dist(dataset[i], dataset[j])
                        if minimum > d:
                            minimum = d
                        if maximum < d:
                            maximum = d
                removed_nodes.pop()
        print dist.__name__, minimum, maximum

def validate(dataset, dist_measures):
    # Change '10' to evaluate another Enron employee
    eid = 10
```

```python
    minimum = 100
    maximum = 0
    for dist in dist_measures:
        matrix = []
        for i in range(152):
            matrix.append([])
            # Some distance measures did not like all null array/list
            if not i in removed_nodes:
                removed_nodes.append(i)
                for j in range(152):
                    if not j in removed_nodes:
                        matrix[i].append(dist(dataset[i], dataset[j]))
                    else:
                        matrix[i].append(0.0)
                removed_nodes.pop()
        # sorted(matrix[x])[3] where 3 is becuase two empty arrays pluss "self-loop" identical matrix
        print 'Distance: %s, min: %s, max: %s, array: %s' % (dist.__name__, sorted(matrix[eid])[3], sorted(matrix[eid])[-1], matrix[eid])

def main():
    # Scipy distance measures between two 1-dimensional arrays
    # http://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.euclidean.html
    distance_measures = [
        distance.braycurtis,
        distance.canberra,
        distance.chebyshev,
        distance.cityblock,
        distance.correlation,
        distance.cosine,
        distance.euclidean,
        distance.hamming,
    ]

    for filename, graph in (('undirected_features.csv', undir_graph), \
                            ('directed_features.csv', dir_graph)):
        with open(filename, 'r') as f:
            n_cols = 0
            # Extract node metadata such as label and color
            for line in f.readlines():
                actor = line.strip().split(',')
                n_cols = len(actor)
                eid = int(actor[0])
                # Add new node to graph
                graph.add_node(eid)
                graph.node[eid]['label'] = eid # or actor[1] for e-mail address
                graph.node[eid]['viz'] = {'color': {
                    'r':int(actor[3]),
                    'g':int(actor[5]),
                    'b':int(actor[4]),
                    'a':float(actor[2])
                }}
            # Extract node features
            dataset = np.zeros((152, n_cols - 6), dtype = np.float)
            f.seek(0)
            for line in f.readlines():
                features = line.strip().split(',')
                eid = int(features[0])
                for i in range(6, n_cols):
                    dataset[eid][i - 6] = float(features[i])

        # Graph construction algorithms
        for dist in distance_measures:
            for k in neighbors:
                k_nearest_neighbors(k, dataset, graph, dist)
            for e in epsilon:
                e_neighborhood(e, dataset, graph, dist)
        #minmax(dataset, distance_measures)

if __name__ == '__main__':
    main()
```

Listing G.1: neighbourhood_approaches.py