A novel data preprocessing solution for large scale digital forensics investigation on big data

Heng Zhang

Masteroppgave
Master i informasjonssikkerhet
30 ECTS
Avdeling for informatikk og medieteknikk
Høgskolen i Gjøvik, 2013

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

A novel data preprocessing solution for large scale digital forensics investigation on big data

Heng Zhang

2013/05/30

Abstract

As the rapid development of high-technology, more and more novel and interesting applications and systems emerge. For example, people are willing to share their life any time any where just by accessing their Facebook accounts. In the same time, the popularity of mobility offices and fault-tolerance working platforms are becoming more and more hot than ever. For example, Dropbox is an popular cloud storage services among the world in recently. In addition, Google collaboration platform is one of the most successful business application for global users to work together in any time, even if they are not in the same office geologically. It is not difficult to find that more similar examples regarding to this concern.

However, Nothing is prefect forever. Technology is a double-edged sword, especially in the information technology field. It pops up a lot of challenges. Consequently, digital forensics investigators pop up a significant question of how to implement large scale digital forensics investigation on big data effectively. It is impossible to handle those cases manually. However, some advanced techniques have been developed by research communities. For example, machine learning techniques are one of the most suitable candidate solutions to handle these big data cases. The significant merit for applying machine learning techniques is not only to introduce an automatic way of working, but also to process those complicated cases with higher precision than other means. Machine learning techniques consist of these stages, input gathering, data preprocessing, algorithm designing & deploying and output evaluation.

The data preprocessing is an inevitable step for achieving better performance from machine learning techniques. However, research societies pay a lot of effort on advanced machine learning algorithm development and performance optimization. The crucial step of data preprocessing seems to be regarded by the same significance. This is the motivation for us to conduct this piece of work in this field.

In this paper, we are going to address how to facilitate the implementation of large-scale digital forensics investigation on big data set with the help of our data preprocessing solution. The methodology introduced in this paper is a hybrid solution based the stochastic theory, Grubbs' criterion and the machine learning method, K Nearest Neighbour (KNN) algorithm. The complete technique contains two round of preprocessing work. While, the performance study on experiment results reflects a considerable achievement by our solution.

Sammendrag

Med den hurtige utviklingen av høyteknologi, fremtrer flere og flere nye og interessante programmer og systemer. For eksempel er folk villige til å dele sitt liv når som helst og hvor som helst gjennom sine Facebook-kontoer. Samtidig øker populariteten av mobile kontorlœsninger og feiltoleranse-arbeidsplattformer. Eksempelvis er Dropbox en nyere og populær skylagringstjeneste som er i bruk i de fleste land i verden. I tillegg er Google sin samarbeidsplattform en av de mest suksessfulle forretningsapplikasjonene som gjœr det mulig for brukere å jobbe sammen når som helst rundt om i verden, uten å måtte være på samme geologiske lokasjon. Det er ikke vanskelig å finne flere lignende eksempler som omfatter dette.

Uansett er ingen ting perfekt for alltid. Teknologi er et tveegget sverd, spesielt i feltet for informasjonsteknologi. Det dukker opp mange utfordringer. Derfor har digitale etterforskere stilt et betydelig spørsmål om hvordan å effektivt implementere storskala digital etterforskning av "big data". Det er umulig å behandle de etterforskningssakene manuelt. Allikevel har noen avanserte teknikker blitt utviklet av forskningssamfunn. For eksempel er maskinlæringsteknikker en av de beste kanditatløsningene for å håndtere etterforskningssaker av "big data". Den betydelige fordelen med å anvende maskinlæringsteknikker er ikke bare at the introduseres en automatisk måte å få ting til å fungere, men også å kunne behandle de kompliserte etterforskningssakene med høyere presisjon enn andre metoder. Maskinlæringsteknikker består av disse stadiene, innsamling av inndata, preprosessering av data, algoritmeutforming & -distribusjon og evaluering av utdata.

Databehandlingen er et uunngåelig steg for å oppnå bedre ytelse fra maskinlæringsteknikker. Allikevel bruker forskningssamfunn mye krefter på utvikling og ytelsesoptimalisering av avanserte maskinlæringsteknikker. Det avgjørende stadiet for preprosessering av data ser ut til å ha vært oversett lenge. Dette er motivasjonen vår for å gjennomføre dette arbeidet i dette uutforskede feltet.

I denne rapporten vil diskutere om hvordan å legge til rette for implementasjon av digital etterforskning av "big data" i storskala, ved å bruke vår løsning for preprosessering av data. Metodologien som blir introdusert i denne rapporten er en hybridløsning basert på teori om stokastisitet, Grubbs' kriterium og maskinlæringsmetoden, KNN-algoritmen. Den fullstendige teknikken inneholder to runder med preprosesseringsarbeid. Samtidig reflekterer ytelsesanalysen av eksperimentresultater i løsningen vår en betydelig prestasjon.

Preface

I would like to show my gratitude to Prof. Katrin Franke for encouraging me to write this paper for master degree theses and in particular for supervising my master thesis work during the fourth semester. Meanwhile, I also would like to appreciate the supervision from Jayson Mackie. He gives me so much valuable guidelines and information for delivering a high quality piece of work.

It is quite a pleasant experience to spend two years of full time master study in Høgskolen i Gjøvik with so many good friends and excel classmates. Here I want to show my thanks to Andrii Shalaginov. I can always learn a lot of stuff from him by knowledge sharing and specific techniques discussion.

Finally, I would like to show my deeply appreciation to my family. I can hardly make any achievement in my life without their love and support.

Heng Zhang, 2013/05/30

Contents

Acronyms

ASIC Application-Specific Integrated Circuit

CPU Central Processing Unit

FPGA Field-Programmable Gate Array

KNN K Nearest Neighbour

PC Personal Computer

SVM Support Vector Machine

XML Extensible Markup Language

DDoS Distributed Denial of Serice

SSD Solid State Drive

MB Megabyte

GB Gigabyte

TB Terabyte

EB Exabytes

List of Figures

List of Tables

1 Introduction

The purpose of this chapter is to outline an overview of the subjects and challenges in question, as well as justification and motivation of its importance. The chapter also proposes research questions to guide the thesis combined with a further discussion of the planned contributions. Meanwhile, the methodology adopted are presented before going forward to the theoretical foundation part. The structure of the report is listed at the end of this chapter.

1.1 Topic covered by this paper

A large number of novel systems and applications emerge in past decades by rapid technology development. One of the most significant phenomenons is more and more bigger data set scenarios appeared than before. It results in a large number of problems in a wide range. For instance, one typical challenge is the large digital forensics techniques are highly demanded to meet the requirements for big data set digital crime investigation. It can be formally formulated as large scale digital forensics investigations on big data set cases.

The topic for this research activities is mainly involved in seeking an effective solution to develop a new method for mitigating such a problem. In our paper, we are specially concentrated to develop a novel data preprocessing method, which has been widely recognized as a very important and indispensable sector for machine learning based digital forensics investigation methodologies.

The difficulties might raise from every phase of digital forensics processing procedure, which consists of different stages as evidence collection, storage, analysis, searching and visualization. Every stage is possible to pop up various kinds of difficulties. It make investigation process more challenging than our expectation.

In this paper we are mainly focusing on the data preprocessing phase. To be specific, that is the preparation work for detecting suspicious or irrelevant data contained in origin evidence collections. After this phase, preprocessed data set would be greatly benefit for following forensics procedures for better accuracy, efficiency and performance.

To achieve our goal, some important techniques and theories are introduced into this master thesis project. For example, python programming competence is demanded for developing a prototype of data preprocessing application. Meanwhile, the statistic analysis theory as machine learning algorithms are also applied and so on.

1.2 Keywords

Information Security, Digital Forensics, Big Data, Machine learning, Data preprocessing

1.3 Problem description

As a novel challenge in this information era, the data processing requirements from everywhere keep on growing with a tremendous scale. What is worse, this challenge becomes more and more severely as time goes on. For our own interested field, a large number of large scale digital forensics investigation cases on big data suffered a lot from this challenge. In front of the big data with various kinds of information, it is not easy to find all valuable evidence for digital forensics investigators.

Some solutions have already been addressed to solve these new challenges. For instance, one solution is to upgrade hardware profiles, like faster multi-core Central Processing Unit (CPU) , specific optimized Application-Specific Integrated Circuit (ASIC), customized Field-Programmable Gate Array (FPGA), etc. However, the growth of hardware processing capabilities still can not completely cover the increasing computational requirement. Due to the theoretical and practical constrains consist of material science research progress, manufacture engineering level, cost control and so on.

On the other side, some software solutions are proposed to handle these problems. Consequently, more novel software applications developed to mitigate this challenge in different levels. The problem here turns to how to improve the efficiency and effectiveness to meet the requirement by all possible resorts.

Recently, it is very popular to introduce machine learning methodologies into the practice of digital forensics investigation. A complete machine learning work flow contains several different phases. For this paper, we put special emphasize on its initial stage, which is the preprocessing stage. Different input for a dedicated machine learning algorithm will produce different output. Analogously, different level of quality in input end will lead to different level of quality in the output end. It is a quite interesting and significant topic worth for further work. This is the exactly research question in this paper.

1.4 Justification motivation and benefits

The importance for solving this problem is to improve the performance for better machine learning based digital forensics investigation solutions. Furthermore, it will cut the economy and time cost for many business enterprises and organizations. What is more, It is also a contribution of knowledge for whole research society.

1.5 Research questions

The main purpose of this work is to develop a useful preprocessing solution to deal with raw digital evidence collections, which will be handled in machine learning based forensics tools by digital forensics investigators. In order to offer a clear view of our work, the following research questions have been well defined and explicitly addressed

in following chapters of this paper.

- **What should be preprocessed by implementing the preprocessing solution ?**
  The objective evidence for large scale digital forensics investigation need to be defined, which will be handled by the preprocessing solution.
- **How to determine the outliers among the raw data set ?**
  This is the core part to achieve the goal of improving the input quality by data preprocessing phase.
- **How to guarantee the preprocessing solution theoretical sound ?**
  The reliability of this solution should be assessed by a solid theory foundation in a scientific way.
- **How to evaluate the performance of the preprocessing techniques ?**
  In order to prove the performance improvement by applying the data preprocessing solution, explicit quantitative analysis work need to be presented.

1.6 Methodology

In order to address these research questions appropriately, it is very necessary to present a clear view of relevant theories and techniques. Furthermore, they are required to link and organize in a proper way.

First of all, systematically literature study is a crucial step to obtain an overview about a specific research question. A comprehensive survey paper of existing theories progress and practice techniques development is so benefit to provide a solid foundation to carry on our own research activities for going further. For example, the survey work from Hodge & Austin [1] made a extensive investigation in the research field of outlier detection. It offered a good starting point for conduct our own study in data preprocessing field.

By literature study, we find that existing methodologies related to data preprocessing attempt to deal with this challenge in different ways. The experiment results derived from their works illustrate different level of performance improvement. All those papers inspire us to work on this paper in a creative way.

Statistics is a very important and useful tool to study the characteristics from a large scale random sample set. Combining this similarity with the big data background in large scale digital forensics investigations, it is quite suitable to deal with the evidence information collection job from raw data set in digital evidence collections. In this paper, we have applied the Grubbs test criterion to conduct the stochastic-based outlier detection as the initial phase of our preprocessing solution.

Machine learning is another very helpful tool to solve this challenge. The advantage of machine learning techniques is able to process the evidence extraction work in an intelligent way. More higher precision can be obtained than other approaches. In this paper, we select the K-Nearest Neighbour algorithm to conduct the preprocessing work

as the second phase of our solution.

Mathematics theories are strikingly presented to convince our solution in a scientific sound manner. The mathematical explanations are illustrated in relevant chapters as the theoretical foundation for each technique applied in our work.

1.7 Contributions

Since this master thesis is dedicated to the field of data preprocessing. The expected output of our work is to develop a novel data preprocessing solution, which is able to improve the performance for large scale digital forensics forensics investigations.

Our work make a contribution to handle the outlying data for digital forensics practice. Sometimes it could be defined as *outliers*. Our preprocessing solution is based on the idea of combination with stochastic theory and supervised machine learning method.

In addition, the research work may provide some new inspiration for other researchers' work.

1.8 Thesis outline

This piece of thesis is divided into several chapters. We applied a top-down approach for presenting a clear view of the all work. The outlines of our work listed in below.

1.8.1 Structure description

- **Chapter 2 Theoretical Foundation**
  This chapter is going to present the relevant theoretical background and the state of art, which are the prerequisites to get a clear view of the topic. Afterwards, we presented the framework of digital forensics science and the large scale investigation challenge.
- **Chapter 3 Data Preprocessing**
  This chapter is going to present topic of data preprocessing. The problem of data problems existing in raw input has been intensively discussed in the beginning. Then we began the detailed discussion on the methodologies of our own solution from the perspective of stochastic and machine learning respectively.

- **Chapter 3 Empirical Implementation**
  This chapter is going to present the experiment framework, which consist of experiment environment construction, test data set preparation, outlier detection and experiment results collection.

- **Chapter 4 Performance Study** This chapter is going to present the comprehensive quantitative analysis. It presented the valuable information from the experiment results. Furthermore, the performance of our proposed solution has been verified.

- **Chapter 5 Conclusion** This chapter is going to draw the conclusion based on all the work in our paper.

- **Chapter 6 Future Work** This chapter is going to present some proposals for future works with our solution.

2 Theoretical Foundation

We intend to organize this chapter in the following way. First of all, background of our work will be proposed. Afterwards we will illustrate the state of art related to our work, which is oriented to provide a clear view for our topic. Then, we are going to provide a concise framework of digital forensics science. In addition, the novel challenge of large scale digital forensics investigation is discussed in details.

2.1 Background

Currently, many systems and applications are relying on the digital flow for fulfilling their own functionality or roles. However, problems also raised from the same place. The scale of data sets keeps increasing in fast speed. It is stimulating people's requirements for deploying high-performance hardware as well as the high-capability software for handling more transactions. For example, it was a long time interval for computer users migrated from an 80386 computers to Intel Pentium CPU computers. But more shorter time interval for computer users migrated from Intel Pentium-series computers to Intel Core i7-series computers. Another example, the 64-bit version operation systems outperform the 32-bit versions, which is able to process more data in the same time than 32-bit version operation systems.

However, hacker societies are also becoming more and more sophisticated. The digital crimes committed by malicious attackers are taking places more frequently than before. For various kinds of purposes, attackers are more interested to choose those high Internet traffic systems as their targets. Since they can take more advantages of such attacks rather than penetrating a common user's Personal Computer (PC). This is one of the reasons accounting for large scale digital crimes.

Consequently, digital forensics investigators have to handle a large number of requests for implementing large scale forensics investigations on different digital crimes. The intuitive reaction to this challenge is to apply machine learning methodologies for handling those cases. Particularly, it is suitable to mitigate the human incompetence for dealing with a lot of evidences collected from digital crime scenes. Since the manual processing speed by digital forensics investigators can hardly complete with a well-designed computer solution.

In this piece of work, we paid a weighted attention to develop a new data preprocessing technique, which is able to handle large scale digital forensics investigations on big data.

Data preprocessing is really an interesting topic and worth of working on it. Because it can puts huge influence on final results with machine learning algorithms. The topic of

data preprocessing really covers a quite wide range. A lot of research activities concentrated on the area of developing outlier detection techniques, which are the key method to implement data preprocessing tasks.

2.2 The state of art

Some efforts made by the research societies to investigate this challenge in different depth and some of them are applicable to dedicated circumstance. For example, some papers pay attention to present an overview within this field. Meanwhile, some other researchers attempted to solve the outlier detection problems by applying cross-discipline knowledge. Additionally, some other researchers contribute to this challenge by endeavouring and optimizing existing solutions for better results.

Research communities already spend many efforts to develop various advanced solutions to manage this challenge. It is necessary to outline their contributions. Taking the retrospective action is aiming to help readers to obtain preliminary knowledge about this field.

For our topic of data preprocessing, some other terminologies are also adopted as data cleansing, data purification, outlier detection and so on. But core task is to detect the outliers and improve the quality of given data set in order to obtain a more tidy set of data for future process. Fortunately, a large number of forerunners have already made significant progress in this topic.

2.2.1 Survey works

Some researchers done survey type of work, which are quite helpful to offer the overview in data preprocessing field. For example, the work from Hodge & Austin presented research works in this field comprehensively[1]. In their paper, it offers an comprehensive outlook for the activities from research society. Several different kinds of methodologies are represented in terms of different properties of raw data sets. For instance, statistical methodology can be applied to do a mathematical analysis for data sets without any knowledge in advance. However, there are also several different specific techniques in stochastic field. One classic technique is based on proximity-based approaches. Another widely adopted method is parametric-based methods, which is quite convenient for evaluating target data set in a quite high speed, even for extremely large scale data set. The reason accounting for such performance is the inherent characteristics of the model. The complexity of this approach is directly related to the model rather than the scale of data. The performance is intimately depending on the given parameters. While, Müller and Freytag also made the similar work on presenting the framework of data cleansing [2]. Comparison among existing data preprocessing approaches has been conducted in their work. Meanwhile, they also offer some quality criterion as the guidelines of data preprocessing. A series of similar papers can be found, like [3], [4], [**?**] and so on.

2.2.2 Statistic solutions

Statistic approaches can be found as the initial stage in this field, one of the most significant milestone is the work of *Procedures for Detecting Outlying Observations in Samples* by Franke E Grubbs in 1969[**?**]. He successful applied the stochastic methodology to deal with the outlier detection problem in random sample sets. The performance of his method shown a convincing result with high satisfaction. Afterwards, a lot of publications are coming up related to stochastic theory. In addition, Barnett has also made great efforts in the same field like the work of *Outlier in statistical data* [6]. Many other research works applied the statistical methods to detect the outlying data in the big random data set, such as [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17] and so on. Main idea of these methodologies based on investigating stochastic properties from the data distribution of all observed random objects.

2.2.3 Machine learning solutions

Machine learning is a more sophisticated approaches comparing to other methods. It is able to self-study from the data set after assigning appropriate algorithms. With the abilities of representation and generalization, machine learning algorithms can process data in a more intelligent way. This is a quite desirable feature for people. Comparing to other techniques, it can conduct the data processing task with higher accuracy in fast speed, without human intervention.

A large number of machine learning methodologies have been applied in order to fulfil the purpose of outlier detection. The kernel-based solution applied to detect the outliers on the different density level, like the work from [29], [30], etc. While, the Support Vector Machine (SVM) is also introduced to deal with the outlier problem described in the papers like [31], [32], etc. Some researchers also tried to solve this problem by applying neural networks algorithm like [33] [34], [35], etc. Moreover, several hybrid systems have been developed by combining different types of methods together, like [36], [37], etc.

KNN is one of the most famous and effective measure to mitigate this challenge. It has been voted as one of top ten significant algorithms by researchers in machine learning discipline[18]. The principle for this algorithm is to measure the distance similarity between other data elements. The clustering behaviour depends on the class of neighbouring elements.

A lot of researchers made a lot of improvements based on the naive *kNN* algorithm. For example, those improvements can be found in the following literatures, such as [19], [11], [20], [21], [22], [23], [24], [25], [26], [27],[28] and so on. Those approaches tried to make use of the similarity characteristic from different angles.

2.3 Large-scale digital forensics investigation

Digital forensics is aiming to implement a complete working procedures related to the incidence details as much as possible. It covers a broad scope of what has happened in

the crime scene, when dose the crime occur, Who are the most likely suspects, how to extract and preserve these information in a scientific sound manner. In order to present reliable evidence in court, it requires thoroughly work and high convincing methodology during the forensics investigation period.

We will take a quick review on the principles of digital forensics discipline. It is a crucial step for further discussion and analysis activities in following chapters.

2.3.1 Preliminary basics

Ontology

It is a scientific sound manner to describe some specific domain by ontology [38]. Regarding to digital forensics discipline, All knowledge related to digital forensics field will be represented and built an overview in the way of digital forensics ontology. The purpose for conducting a digital forensics investigation is trying to find out objects, evidences and other entities occurred or existed in the digital crime scenes as well as the relationship among them. Some researchers have already spend efforts in this area. For example, the work from Brinson et al. illustrates the cyber forensics ontology 1 as followed[39].

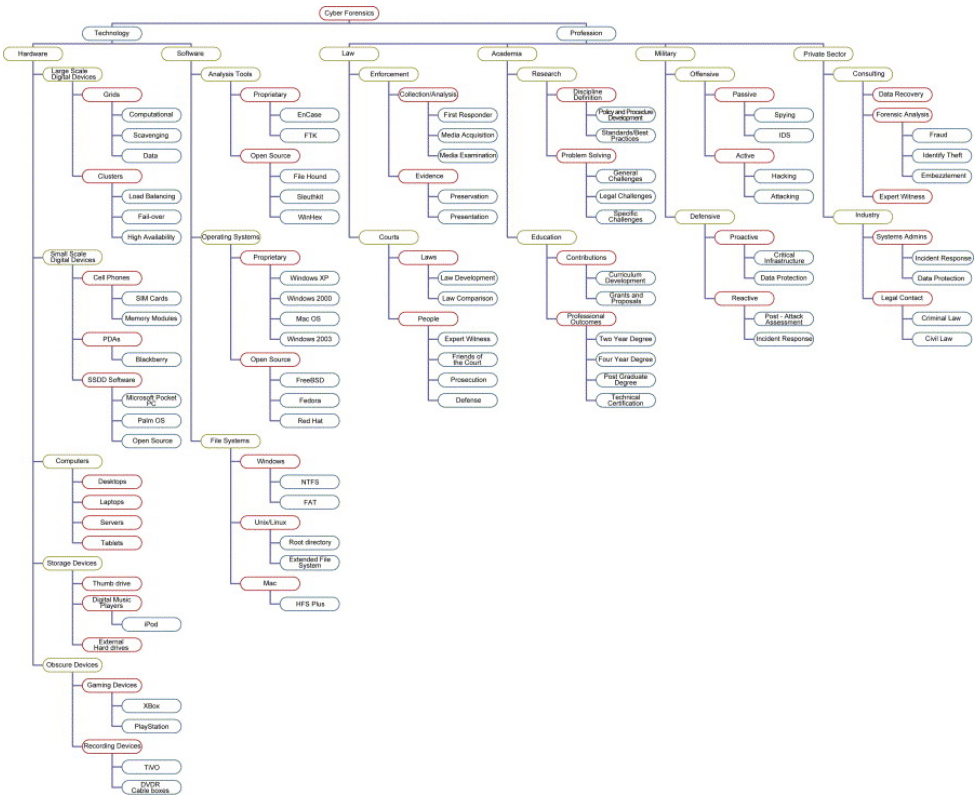

Figure 1: Cyber Forensics Ontology

However, the main motivation for us to explain digital forensics ontology is to develop a data preprocessing solution in a forensics sound manner. It will significantly leverage the efficiency and accuracy in most circumstances. For example, a machine learning based digital forensics investigation system could be developed for working

automatically with properly designed ontology form, such as by the Extensible Markup Language (XML) machine readable file. A classic paper from the research work by Klein et al., which addressed this methodology comprehensively 1 [40]. This is the popular methodology applying in current digital forensics practices.

Digital evidence

Nowadays, thousands of electronic devices or information systems show up in daily life, greatly benefit of our life in a more enjoyable and convenient way. However, every single action in this digital world would probably produce some kind of traces residing in the information systems, let alone the massive scale Distributed Denial of Serice (DDoS) [41] attacks launched by malicious hackers. That means the more interactions occurred with information systems, the more records or traces would be preserved in the systems. From the perspective of digital forensics, all these traceable records are defined as ***digital evidence***.

However, digital evidence is really a complex conception, which can refer to various kinds of aspects. Generally speaking, digital evidence refers to any digital data or information, which is capable to proof a committed crime case or substantial legal relationships among the perpetrators, digital crime cases and victims.

Properties of digital evidence

The key regulation to govern the digital evidence about lawsuits within digital crimes in the court is ***Daubert standard*** [42], which is widely applied in the judicial practice around the world since officially adopted by the Department of Justice of United States.

Daubert Standard

The Daubert Standard is an important regulation in juridical practice, which enables the expert witnesses' testimony of digital evidences admissible during the trial in the court. Daubert criterion [43] is elaborated as followed.

> A witness who is qualified as an expert by knowledge, skill, experience, training, or education may testify in the form of an opinion or otherwise if:
>
> (a) the expert's scientific, technical, or other specialized knowledge will help the trier of fact to understand the evidence or to determine a fact in issue;
>
> (b) the testimony is based on sufficient facts or data;
>
> (c) the testimony is the product of reliable principles and methods; and
>
> (d) the expert has reliably applied the principles and methods to the facts of the case.

Consequently, several crucial prerequisites should be guaranteed for those digital evidences. We are going to elaborate them in the following paragraph.

- ***Admissibility***

  This is the predominating concern for any digital investigation project involved in the lawsuit. All evidences collecting and processing methodologies have to comply with this rule. Otherwise, no matter what has been done by the digital forensics investiga-

tors are totally in vain without the admissible approval by the court.

- ***Authenticity***

  The property of authenticity is aiming to offer the genuine based for making a convincable decision by the jury or judge according to relevant regulations. It is necessary to ensure the quality of forensics work on the evidences before presenting in the court.

- ***Associativity***

  All the evidences presented in the court should be directly or indirectly related to the defendant and his or her behaviours to commit crime cases. The link among the suspects and collected evidences by forensics investigators should be sufficient to support a proper final decision of the juristic trial. Since it is the essential foundation for admissibility.

- ***Integrity***

  Among the investigation process, the original status of evidences should be preserved properly. Any tampering actions or changes even without purpose can introduce some extent of uncertainty for the lawsuit. Based on this consideration, all the evidences collected from the crime scenes should make an image or backup files for further investigation.

- ***Reliability***

  This means that procedures for gathering evidences and analysing invisible details should be fully accountable. Any ambiguous factor residing in the evidence chain have to wipe out in advance. To be concisely, the digital evidence submitted into the court should be a bundle of essence of trustful proofs.

2.3.2 Principles of digital forensics

A recognized definition of digital forensics can be found in the paper from Kruse and Heiser as followed :

> "*. . . forensics is the preservation, identification, extraction, documentation and interpretation of computer media for evidentiary and / or root cause analysis . . .* " [44]

Meanwhile, another definition is given in the technical report from DFRWS[45] (Digital Forensic Research WorkShop), which is an international community for both academics and practitioners related to the field of digital forensics. The definition is quite realistic associating with the industry practice.

> "*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.* [46]

From their perspectives, it is a science of dealing with every phase of investigation activities for the lawsuit against those digital crimes. We can see the technical investigation

mainly gets involved with the *targets* and the *implementation methodologies*. Sometimes, the term of *targets* could be regarded as scenes of digital crimes.

Although evidence-oriented forensics work is closely involving with the phases like preservation, identification, extraction and so on. The core principles for digital forensics work are Chain of Custody (**CoC**) as well as Order of Volatility (**OOV**), which guarantees the quality of the final work more reliability.

1. ***Chain of Custody***
   Chain of Custody is a dedicated terminology in the legal context. It is mainly referring to document the actions conducting in a complete forensics process. Whenever the digital forensics investigators conduct the standard behaviours like discovery, preservation, possession for the findings of evidence. In this circumstance, digital forensics investigators are fully responsible for the custody of every phase in the investigation chain.

   The biggest concern for the evidence collected by digital forensics process is tampering operation on original digital evidence. Since the unique property of digital crime scene is transient and fragile. In consequent, some actions should be taken in order to ensure the integrity of evidence. In case the operations among investigation process will contaminate or purge the origin evidence unintentionally. Hashing the image copy of origin evidence are widely applied in practice as an integrity check methodology.

2. ***Order of Volatility***
   As the processors in computer systems become more and more faster than before. The status of information varied more and more rapidly in the systems. For example, the operation speed of reading and writing a floppy disk is quite slow. It changed tremendously when the operation occurred in an hard disk. Furthermore, the operation speed on new storage devices, like Solid State Drive (SSD), shows even more faster than all past storage devices. This is really a big challenge for forensics investigation practice.

   Order of Volatility is a significant principle for handling such a problem, which is oriented to the lifetime of different data which are existing in different layers and locations in the computer system. The evidence should be collected as much as possible for supporting further legitimate actions. We have to consider working in a reasonable way with different level of priorities, which are especially depending on the volatility of evidence. For instance, the evidence existing in CPU cache is more volatility than the evidence existing in hard disk. There is the table 1 shown the transient properties of evidence existing in various locations of a computer system [47] .
   From the table 1, evidence from CPU cache is more volatility than hard disk or optical disk. Correspondingly, the former one should do necessary real time forensics action and the later one could perform a post-mortem investigating task later.

Registers, peripheral memory, caches, etc.	nanoseconds
Main Memory	nanoseconds
Network state	milliseconds
Running processes	seconds
Disk	minutes
Floppies, backup media, etc.	years
CD-ROMs, printouts, etc.	tens of years

Table 1: Table of Volatility

2.3.3 Process of digital forensics

Digital forensics investigation is a series of serious scientific activities, which is closely related to gathering evidence in the digital crime scene. we are going to illustrate from the perspective of practice. This is quite benefit of illuminating our topic for those interested readers.

It is a crucial step to sketch the standard process for digital forensics. Many researchers have already made some significant contributions in this field. For example, we can find the relevant contents from the research publications, which addresses this question from different angles. [48], [49], [50], [51], [52] and so on.

Identification	Preservation	Collection	Examination	Analysis	Presentation	Decision
Event/Crime Detection	Case Management	Preservation	Preservation	Preservation	Documentation	
Resolve Signture	Imaging Technologies	Approved Methods	Traceability	Traceability	Expert Testimony	
Profile Detection	Chain of Custody	Approved Software	Validation Techniques	Statistical	Clarification	
Anomalous Detection	Time Synch.	Approved Hardware	Filtering Techniques	Protocols	Mission Impact Statement	
Complaints		Legal Authority	Pattern Matching	Data Mining	Recommended Countermeasure	
System Monitoring		Lossless Compression	Hidden Data Discovery	Timeline	Statistical Interpretation	
Audit Analysis		Sampling	Hidden Data Extraction	Link		
Etc.		Data Reduction		Spacial		
		Recovery Techniques				

Figure 2: Digital Forensics Investigation Process

By checking table2, we can get a full idea of digital forensics. So much information has been conveyed by the units with different tasks. However, the main concerns for practice can be found in the X-axis. If we regarding the X-axis in the first row as a vector, it consists of seven components as identification, presentation, collection, examination,

analysis, presentation and decision. Almost every component contains several necessary tasks. We will depict it concisely as followed.

1. ***Identification***

   In this phase, the main task is to identify the incidence in a comprehensive way. It gets start from the system alerts and some other similar monitoring modules or components. For most circumstances, the auditing behaviours implemented as a routine. The crime scene forensics work are oriented to detect the relevant profiles, anomalies and so on. In the same time, the complaints should be also figured out.

2. ***Preservation***

   The main task for preservation phase is to manage discovered evidence among the entire investigation process. It is a necessary procedure for implementing forensics investigation actions on the backup or image files derived from the reliable and approved techniques. According to principle of Chain of Custody, the origin evidence should be preserved in intact status. All kinds of details need to associate with the time synchronization for further documentation precessing. The case management work should be done also in this phase.

3. ***Collection***

   It is an important sector for constructing a foundation for analysis work in future stage. When forensics investigators conduct their behaviours, the principles we mentioned before should be clearly bear in mind. Not only the integrity of evidences should be well managed appropriately, but also the order of working procedure should be taken into account to the volatility of digital evidences existed in different locations. However, a significant awareness should be bear in mind when accessing some type of confidential documents the approvals should grant by relevant law enforcements agencies. All practices and operations by forensics investigators must conform to legitimate acts and regulations rigorously. In order to facilitate future work, the data reduction task is an optional choice to the investigators. Meanwhile, the recovery techniques should be also ready to use in case some special requirements in latter stage.

4. ***Examination***

   For the job in this phase, forensics investigators should pay more attention to discover more details and information from various kinds of media and evidences. They should trace back all the behaviours happened in the past with an inverse order of time line. Meanwhile, the hidden clues should be pay enough effort to examine carefully. Since a sophisticated hacker will probably tamper or hide the traces in victims' system in order to cover his traces. When found the hidden evidence, the extraction procedure need to implement carefully. Sometimes, machine learning techniques are quite useful to extract evidence, especially among massive irrelevant information.

5. ***Analysis***

   Analysis is a crucial work for interpreting the collected evidence correctly. But it is a really abstract concept and can contain a lot of aspects around this issue. However, several typical methodologies have been pointed out. The primary one is analysing the traceability of the findings. If indeed, maybe more details can be derived in some way. Another one is doing statistical analysis with a bundle of evidences. The mathematical features can reveal some invisible but important information from the evidence piles. A special case in the statistic methodology is spacial analysis. If some type of mode shown up spatially among the statistical space. It might be more important than those density ones. Due to more valuable information can be conveyed by them. Furthermore, it is quite helpful to detect different types of protocols in this stage. Every protocol has its own features and characteristics. They are quite beneficial of the analysis work. Machine learning techniques are widely applied in this phase. It greatly mitigates the capability gaps between human beings and computer systems in the aspect of speed and accuracy, especially for large scale cases and time-critical cases. The associativity between every independent evidence should be taken into account, which is a logic line for reconstruction the digital crime scene. Another crucial aspect is the time line information, which is curving all the activities with the time dimension.

6. ***Presentation***

   Presentation work is very important to summarize your work properly. The first priority should be deliver a well written documentation. Then all the evidences should be illustrated convincing testimonies. However, influencing objects should be stated in proper way. Furthermore, the recommended counter measures also need to be proposed in the presentation. In addition, it is quite helpful to show the interpretation of the statistical work in the final report.

7. ***Decision***

   The last phase for a complete digital forensics investigation is to make an objective conclusion based on all conducted activities with the digital evidences. However, the main role for this phase is not the digital forensics investigators. That is the responsibility for the judge and jurors in the court. Based on the court debate and digital forensics report etc., they can make an official decision on the suspects by corresponding legitimate regulations.

2.3.4 Large scale challenges

As the technology rapidly develops, it deeply changes a lot of things in every aspect of people's life. One of the challenge is the large scale digital crimes, which never existed in the past days. What is worse, this shows a trend of increasing popular in future. In this section, we will be going to address this novel challenge comprehensively.

Big data challenge

Digital forensics investigation might leave such an obscure impression in people's mind. That is working closely with a floppy disk or hard disk from the victims' computers or information systems, which is too pedantic to leave the public an impression that investigators only doing such kind of routine jobs every day. However, time flies and things changed. The capacity for the storage devices increasing from Megabyte (MB) to Gigabyte (GB), then heading to Terabyte (TB) size just in quite a short time. What is worse, so far we can see that there appears a trend of continuously increasing storage capacity requirement and without an end in the near future. However, that is not all the story. From the another perspective, the various kinds of information generated by different kinds of devices or systems add more complexities for understanding.

Big data

In past decades, the information technologies have been developed with quite fast paces. A large number of novel inventions have been created. People can conveniently get more faster network access speed as well as the more higher processor frequencies. Meanwhile, the most interesting phenomenon is the expenditure incredibly going down than before. But it is going to generate more imperfect, complicated and unstructured form of massive data set.

By an intuitive point of view, big data could be regarded as some extremely large set of data collection, which is too large in scale or too complicated in logic structure to overwhelm the feasibility of capturing, processing, storing, interpreting and visualizing by existing database management systems or traditional business intelligence applications.

It sounds like such a pleasant offer for us that we can get better experience if people do not need to pay more money for it. Just in this reason, it is stimulating more and more bigger information generated directly among thousands of different channels. For example, more broader bandwidth turns the on-line streaming and business services into reality. This can be proofed by the blooming popularity as Youtube, Flicker, so on. In the same time, the mobility terminals become to be more and more powerful, like PDA, tablet PC, intelligence cellphones and so on. Furthermore, there are many different kinds of popular applications running in people's computers, like Skype, BitTorrent, Dropbox and so on. Correspondingly, many people severely addict to use social networking websites like Facebook,twitter, Linkedin, and soon.

In recent year, a lot of high popularity large scale computation services have brought to the public. Several famous examples for implementing large scale computation for big data are listed out in below.

- Amazon Elastic Cloud,
- Google App Engine,
- Microsoft Azure,
- IBM Big Data Platform,

- VCE's Vblock,
- Falcon Credit Card Fraud Detection System,
- NASA Center for Climate Simulation,
- SETI@HOME / Rosetta@HOME / RNA World by BONIC Project,[**?**]

For example, Amazon Elastic Cloud or Google cloud services both are offered cloud storage service for the public with different sizes, which is mainly depending on the various requirements from customers. Google drive, as an online storage service from Google cloud services, offers at least 5 GB capacity for free. Hundreds of millions of people signed up Google account around the world. How can we imagine the storage space Google needs ? The operations from Google users as production, submission, presentation, communication, collaboration, modification and deletion take place in every second. All these data changes are processed by Google server clusters in time. In addition, this is not all of the challenges. In the cloud environment, even any trivial operation occurred has to be synchronized with the remote Google servers located somewhere. This mechanism leads to extra data traffic for the origin data size. As a result, the massive data flow running inside the Google data center is an astronomically scale. The online business websites like eBay have to process millions of online orders from the customers around the world simultaneously without any error. Meanwhile, online shopping transactions are not a single autonomic system. It has to communicate with the different bank systems for payment as well as the logistic shipment system and stock system, etc. That is a quite complicated system. Any error occurred in any phase will lead to complain or failure for some certain deal case. It is also high critical challenge for handling such a big data scenario. Furthermore, social networking services, like Facebook, Twitter, Flicker and so on can produce massive data flow in every second by the large user numbers. Moreover, large scale distributed computing platforms, like BONIC project consisting of tasks like SETI@HOME[53], RNA World[54],etc. is running with a global scale and processing all the data packages collaboratively with other clients within the same grid simultaneously. In consequent, we can easily find that how big volume of data running in the system.

It is not difficult to imagine how large amount of data volume generated from the various channels we mentioned above. Some interesting data have been revealed by Cisco as followed, according to *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017* [55].

- Global mobile data traffic grew 70 percent in 2012.
- Last year's mobile data traffic was nearly twelve times the size of the entire global Internet in 2000.
- Mobile video traffic exceeded 50 percent for the first time in 2012.
- In 2012, a fourth-generation (4G) connection generated 19 times more traffic on average than a non-4G connection.

In addition, Cisco has also make a prediction figure 3 of internet traffic during 2012 to 2017. It is also a solid evidence to support the fact that big data is a real fact. From that report, we can find that the traffic increases in a remarkable manner. The characteristic of this trend is non-linear relationship. To be exactly, we can say that it will

approximately increase exponentially in near future. Since the volume of data traffic in 2012 is 0.9 Exabytes per month. When it will be up to 11.2 Exabytes for the same time in 2017. What we have to mentioned her is one Exabytes equals to two to the sixtieth power bytes. (1Exabytes (EB) = 2^{16} Byte)

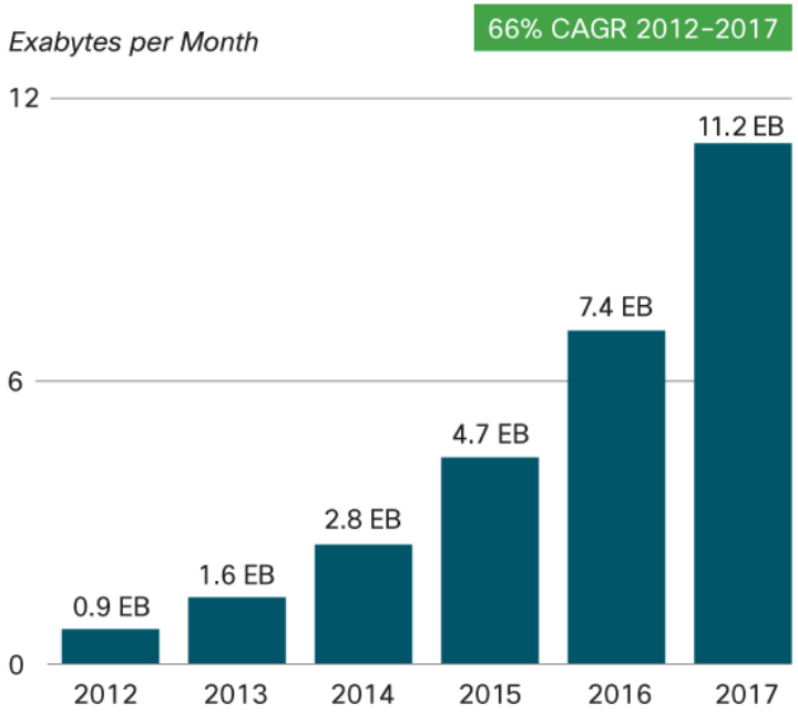

Figure 3: Traffic Prediction From Cisco

In front of the extremely large data, it seems that this is far beyond the processing capabilities of the naive approaches used by now. When asked a question of launching an digital forensics investigation on such a large scale crime scene, it is a really urgent problem for people to develop new solutions to handle it.This is also a very challenging question in the scope of digital forensics.

We can not ignore the complexity of a question are not always rigorously proportional to the scale in a linear way. For some circumstances, non-linear relationship takes place more than linear one. When large scale scenario applied into the digital crime cases, the problems probably turn to be a very difficult problem. That is a real challenge changed the way of working for digital forensics investigators.

Beside the extremely large scale of the data set, many different kinds of hard problems also occurred relating to the fact of big data. We will go to outline the challenges related to the digital forensics concerns, which has practical significance for developing the correspondingly digital forensics techniques.

Challenges

- **Privacy**
  First priority of the security concern is the privacy, which is the crucial sensitive information for individuals. If we describe this concept in a technical way, it could be regarded as

  > right of individuals to control or influence what information related to them may be disclosed [56].

  . Here we can see that individuals not only conclude the real individual person but also the social units, such as business companies, public organizations or government institutions, etc.

  Nowadays, many websites resort to collect all-round information from dedicated group of people or objects with purpose to make benefit from it. That is why always people have to click various message boxes like "*OK*" or "*Accept*" or "*Agree*" during surfing the web or accessing some kinds of content. The most common circumstances occurred in ignoring ways,such as blocking other program, freezing the screen, continuously popping up, and so on. People can only give consent to click that button, then all the unpleasant stuff vanished immediately. However, it is grant the right to collect some types of information including obvious privacy data for the program or website owners or designer. By this means, some special purposes can be meet by the malicious parties relying on those data set, such as health information, gender information, religion information, political attitude, and so on.

  An simple application regarding to get benefit from this scenario, the political election candidates can conduct an intensively investigation on the collected big data from his constituency to make an attractive election promises, which is an intelligent approach to redirect the votes to his own side. As a matter of fact, Barack Obama 2012 campaign has successfully applied this big data analysis approach to overturn the traditional dominance of TV campaign advertising [57]. However, even for regular website or so-called *big websites* like Facebook or Google, people have filled so much information on their servers. Based on those information, Google or Facebook can market their high precision advertisement successfully by the big data set. The foundation for ***precison*** is thoroughly analysing customers' personal privacy information for deriving specific marketing patterns with high confidence level. In the other hand, such huge size of data collected by the service providers will probably portray individual's life with quite a small granularity. It means that all your life will be monitored by the data holders or relevant government departments with interpreting big data set.

  This raises the issue of protecting confidentiality properly in big data set scenarios. Regarding to the scope of digital forensics, the digital forensics investigators should try to solve the problem like finding out the evidence of malwares related to violate privacy information on big data set as well as other technical challenges.

- **Access**
  It is a quite important step to access the information, which needs to be investigated

by digital forensics investigators. In the old days, it is quite convenient to achieve this goal. All the relevant evidences can be physically collected in the scene of crime. But the situation has totally changed in the big data scenario, especially for cloud environments.

For example, if some malicious hackers launched some certain attack by attacking programs or scripts residing in Google App Engine. All the attacking procedures are implemented according to the scripts saved in that Google AppEngine account. The first problem is where should the forensics investigators find that physical devices, like hard disk, data CD, etc. Since all the attacking behaviours taken place in the cloud environment. The data evidences would reside in the suspects' account or Google servers. To access attacker's account, the login confidential should be known in advance. Otherwise, it is impossible to pass the authentication mechanism by Google servers. Maybe investigators can collect the evidence in Google data center in some special cases, like CIA, FBI or NSA law enforcement agencies and so on. But for most circumstances, it is impossible to get the data from service providers. To say the least, even the evidences can be retrieved by some resorts there is still impossible to locate the real attackers in real person. Since the information for signing up that account could be totally subtly fabricated.

- **Collection**
  The novel phenomenon with the big data background, all data generated, processed, stored would be likely to scatter here and there. Especially, the most popularity concepts like working in cloud or grid computation advocate for this selling point with great efforts. The idea behind them are collaboration in the same time without considerations of geological restrictions, time zone differences, data processing capacity, etc.

  A simple example is the Bonic project from UC Berkeley. The participant for some certain sub-project in Bonic would be located in any city, any country, even any continent. The capacity of computation on client's computers varied quite differently, depending on different hardware configurations and online time. The servers divide the whole project into a large amount of small task packages. Every client can fetch its own share of work packages whenever its time slot available. After processed the fetched package, the corresponding result will submit to the server. In this case, if we have to trace a dedicated data in some contain package, it seems nearly impossible to do that. At first, the whole information all stored in the database located in the super data center. That is similar as finding a needle in a haystack. Secondly, the data is likely to be assigned into some specific data package for some certain task for remote clients. Afterwards, the processed results will be returned back to servers. During the whole process, it is impossible to estimate the location of a dedicated data package. Furthermore, it is impossible to estimate the processing time on a specific computer. It is also impossible to trace the return path of that dedicated data package in the whole collaboration network, and so on. If we intend to collect the target information or evidence contained in some dedicated data package, every phase in this case can introduce different level of complexity for performing digital forensics investiga-

tion. For a complete process, that will probably be a very substantial complexity in total.

We have to consider the physical restrains both from hardware aspects and software aspects. We can see that if we intend to collect evidence from just one place. It seems quite easier than the distributed network environment as we mentioned above. But the challenge is also with high complexities from the capacity or constrain of somewhere. For instance, even for a single web server, we have to conduct the live digital forensics investigation for online fraudulent incidence. It pops up a difficult question as how to locate the target information among thousands of threads of service request from the remote clients. If that web site is a very popular one, there would probably be millions of connections existed in the same time. Without efficient solutions, the valuable evidence would be likely missed by the massive user requests from remote. In addition, even all the information was stored in the hard disk in that web server. One extreme case is how can the digital forensics investigators find the valuable evidence from the Terabyte-level storage media within a short period. It is always some kind of constrain on the time table in the working environment. We should count in all relevant evidences collection time in total. That means if the court defined a specific time buffer for working on forensics investigation procedures, the collection time for every single evidence should be quite a short period. There are many cases like this.

- **Interpretation**
  Besides all above considerations, we have to face the challenges as the logical complexities in the data structure or mysteriously relationship for each element in a clear way. In some circumstances, it has been categorized as the problem of data visualization. Without decoded the inside meaning in a correct way, the task of visualizing the big data set seems to be an incredible thing. No matter how reluctant we are, this is a substantial challenge we have to face to.

  In a brief case, we can take an example for describing this problem, which can show a picture to the readers. Nowadays, more computation is related to the simulation of important phenomenon models related to people's daily life, such as the climate forecast simulation for meteorology agencies can facilitate people's personal arrangement and business plan, the stock market prediction for the bankers in Wall street can estimate the potential risks and possible trends in future to make big money from investment or transactions, the crustal changes simulation of the earth for geology scientists' research and so on. Even for the industrial design and manufacture circumstance, simulation technology is also a very significant approach to collect the necessary data with high value. Sometimes, it is nearly impossible to construct such kinds of prototypes for experiments in physical environment, based on the concerns of economy or time. People need to understand the relationship and every details for high accuracy simulation output. In such cases, the computation capacity is not the only vital factor to impact the final consequence.

  The more complex phenomenon is modelled, the more complicated factors are needed

to take in to account. All types of patterns and variables should be considered in a complete way. However, this is impossible to realize all aspects into practice of simulating computation on super-computers. For solving this problem, researchers have developed many approaches to facilitate this task without decreasing the accuracy of the simulation results. One solution is properly selecting some more priority factors or sets of variable with higher weight, which could be possible to obtain a better approximation results than the naive methods. If we put this scenario into digital practice practice analogically, we have to face such a case to find the digital evidence with a broad range of different types of information and structured data packages with a big data set background.

As we mentioned above, we can see such a picture that is when traditional digital forensics practices encounter the big data scenario, the difficulty for conducting such a forensics investigation will increase with a non-linear relationship with high possibility. Sometimes, it is not difficult to come across those cases with exponential growth.

Realistic requirements

The information security industry is a significant and meaningful business in current world. It offers the security protection and defence related to run a secure business. From the news reports from various mass media around the world, people can see that as soon as some hacker successfully attacked some certain business companies, their stock prices will be decreased immediately in the stock market. If the successful intrusion incidences occurred in military systems or government institutions, the loss of confidential profiles would probably be an unaffordable thing. That is the realistic demanding requirements to motivate digital forensics science going forward.

As a novel challenge for digital forensics technology, practice on large-scale crime scenarios pops up a quite tough situation in front of us, both for the industry stakeholders and the academy society. Although, it is rooted from the tradition digital forensics practice. It is still quite different than the tradition forensics methodologies.

First of all, we are going to discuss an interesting case with the digital forensics practice with the big data set background in the follow paragraphs, which could give some inspirations for our later discussion.

There is a large number of various attacking patterns and methods are popular within the hacking communities. However, more and more newly invented approaches are being developed by those sophisticated hackers or hacking groups every day. It asks the research communities and digital forensics investigators to address such challenges rapidly, in order to catch up with the increasing urgent requirements from the market.

As the continuous development of network technology, everyone's live is more dependant on the Internet. It turns to be one of the most important things in our daily life. People use email services to share information without concerning about the delivery

time in the way, use Skype to make conversations without concerning about the geography distance between the speakers. That is a quite awesome thing and very convenient for us. But a lot of new attacking techniques has been exquisitely designed for the VoIP environment recently.

For example, people sometimes prefer to share information online via instant messengers, like Skype or MSN. When users received some files or web links sending reminder from someone in their contacts list. For most cases, the remote party, who has totally no security awareness in mind, will probably click the confirmation button or the suspicious web link. If they were the normal files or web links, here is no problem to do that. But if the requests are sent by the malicious attackers, there will be a very serious security problems for the target machine. In other words, we can called them *victim systems*. One typical approach is the web links are subtly fabricated by the attackers. It looks like the common picture files, with the file name ending by *.jpg*. People get used to share the funny pictures among friends without considering the potential risks. As soon as the click on that web link, victim systems get trapped by the malicious attackers.

The most common cases is the phishing attack, which has forged a website with high similarity like the genuine one, especially for displaying the web pages from a real bank. Then people are prompted that the password should be reset immediately for security concerns. Then the phishing log on interface of that forgery bank web site are presented in victim's screen. A large number of innocent victims are naively input their user IDs and real passwords. When they found the failure of login, they will probably try it again and again. They totally have no idea what they have done by themselves. For remote attackers, they are quite glad to see such kind of behaviours from those benighted. When the victims try their log-on confidential again and again, it will display by plain text on the screen of attacking computer, which tells the malicious attackers that this is definitely my password and it should work why I still can not log on my bank account.

However, this is the most naive way of attacking. No big deal for some experienced computer users. They will keep an eye on such cases by having a glance at the address bar. It is a quite effective way of avoiding such type of attack by checking the URL there. The more common cases is introducing some well-designed virus to achieve their purposes. For example, one of the Skype based virus called *W32/Ramex.A* [58]. This is some type of worm virus in Skype network with high infectiousness. It takes the advantage of the Skype API, which is an emergency vulnerability by Skype. When people clicked the web link, the "*.jpeg*" image file will actually turn to be a virus file. It calls for the windows dialogue message box as "*Run/Save*" and insists in asking for the perform an action either to run or save a virus file by the name of "*.scr*", which is an executable file format in windows operation system. If the permission has been granted, the victim systems have substantially infected virus of *W32/Ramex.A*. Then, the victim system will automatically spread out this virus by sending the same malicious request to all the people in the contacts list by Skype. The propaganda speed of this virus can be reached up to a very big number.

The problem for those infected computers is not only spreading the virus itself to

other computers in the whole Skype network, it could be also taken down those victim systems as the zombies controlled by the remote hackers. They can perform any type of operations just if they want. By this approach, it is easily for malicious attackers to build a large scale private attacking network, which has the high capability of computational resource and enormous network traffic. It is quite a big risk for those hackers to manipulate such kind of network. It could leverage the final power of their attacks by the rate of couples of times. All Skype users over the world reach up to hundreds of millions. This is also popping up a urgent challenge as how to carry on the digital forensics methodology in such a large scare cases.

For another example, the BitTorrent is a very convenient software for sharing the large scale file around the world with a relativity high speed, depending on the number of peers. The more active bit torrent clients running in the network, the higher speed can yield for the participants. However, it has been widely abused to spreading illegal files, copyright protection films and music albums, and so on. This is a novel urgent requirement for digital forensics industry from the market. Since the financial loss caused by pirate film copies from Hollywood goes up to billions dollars per year. That is why the priority for implementing large scale digital forensics investigation is so high. Without the reliable evidence in hand, Hollywood can not win a lawsuit in court.

The problem for carrying on a successful digital forensics investigation on BitTorrent networks is quite a big challenge. According to the BitTorrent protocol, all the transmission are encrypted from end to end. That offers high security for the sharing process. But for the third party, like the forensics investigators or law enforcement inspectors, it is not easy to detect, which type of information is sharing in transmission. Furthermore, the seeding scheme of bit torrent application turns every peer in the sharing network to be the host and client in the same time. It tremendously expands the spectrum of investigation for digital forensics practitioners. As a result, much more efforts need to be spend to fulfil a thoroughly investigation than the common single computer evidence collection in the traditional scenario. Meanwhile, distributed design framework makes it is quite difficult to locate the target files and the real illegal publishers. Without a nature person in the indictment, it is meaningless to raise a lawsuit in court, and so on.

By now, large scale digital forensics methodology is quite important, urgent, necessary and demanding in current time. Due to the big data background, the difficulties have been amplified by many times for carrying on large scale digital forensics investigation. Most traditional standalone computer forensics investigation approaches can hardly meet this big data challenge. New techniques and innovative approaches should be developed, both for efficiency consideration and effective concerns. This is the main motivation for we to work in this paper.

3 Data Preprocessing

Nowadays, machine learning methodology is becoming increasing popular among different kinds of scenarios, from business unities to personal life, etc. However, the importance of machine learning methodologies can not be overemphasized in this field. Many mature machine learning algorithms are available by now, like SVM, K-Means, Decision tree, Artificial neural networks and so on. The importance of data preprocessing procedure can not be overemphasized for any machine learning application as well. The quality for learning input put significant influence on final learning results.

Data preprocessing is an inevitable phase for a successful machine learning application. As a prerequisite, this procedure is able to improve the quality of the input and facilitate the subsequent procedures in a complete machine learning work flow. The saying of "Garbage in, garbage out" intuitively reflects this situation for many machine learning practices.

The main task for data preprocessing is to represent the data in a proper way depending on users' own purposes with some feasible approaches. It is a good example comparing to human being's information processing way with high similarity, pointed by Bobrow & Norman's work citeBobrow as followed.

> "... Consider the human information processing system. Sensory data arrive through the sense organs to be processed. Low level computational structures perform the first stages of analysis and then the results are passed to other processing structures. ..."

Appropriately processed data set can eliminate the unexpected elements and undesirable influence. Data preprocessing process is a time-consuming task. The measures adapted into this process are aiming to overcome those problems encountered in every individual case.

3.1 Definition

Even though the importance for data preprocessing task is commonly recognized and mentioned by all kinds of communities and stakeholders. There is still no uniform definition given out clearly. Here we are going to describe it in a mathematics way based on the work by Famili et al. [59] as followed.

Considering a mathematics transformation F implemented on the domain X, which can establish a mapping relationship to domain Y as followed.

For finite sets X and Y, it meets such a relationship like,

$$\forall x_i \in X, \exists X_{ip} \rightarrow Y_{iq} \tag{3.1}$$

Where,

1. Domain X : raw input data set.
2. Domain Y : consequence data set processed artificially.
3. Transformation F : algorithms for establishing the mapping relationship.

According to our purpose, we should try to find a proper transformation F for solving the challenge with a better performance than non-optimization solutions. Consequently, It is necessary to interpret this mathematics expression a little bit further.

The raw input data set X consists of massive information, both valuable parts for us and mess parts for us in the same time. However, the processed data set Y should try to filtered out the noise and irrelevant information contained in the origin data set X as much as possible. In order to express this difference before and after the transformation process, we have introduced the footnote p for X_{ip} and footnote q for Y_{iq}. Specifically, the footnote i denotes the serial number of the elements in every data set. Meanwhile, the footnote p denotes the number of elements existing in raw input. Furthermore, the footnote q denotes the number of elements existing in processed data set.

For clearly understanding, it is also worth to point out the properties of this transformation in detail as followed.

1. $p > q$, due to the noisy and irrelevant elements have been eliminated from the data set.
2. In ideal circumstances, the valuable knowledge volume of X should equal to those of Y. If not, the most precisely approximation should be achieved in Y.
3. The F should strictly work in the defined domain and can not introduce new uncertainties.

As a result, our mission could rewrite as how to find a proper transformation F to meet all the three requirements in the same time. If it is impossible for all situations, the candidate transformation should be one of the most accurate optimization model than all the rest ones.

3.2 Problem with data

Regarding to the tasks like data preprocessing, data cleaning or data purifying, the kernel problem is handling the outliers in a proper way in order to get a relatively satisfactory set of elements. In some other publications, outliers also be called by "*Anomalies*". But the readers should be aware of that there is no essential discrepancies between these synonyms. In order to follow the same style, we will prefer to use *outliers* in consecutive chapters.

As the priority concern, it is quite important to have a clear view in mind that what is

outlier. Due to in different application contexts, the terminology of *outlier* should probably refer to quite different type of specific data elements or clusters in that raw data set. It is really complicated to address this question. But we will try to present an elaborated map of it in a structured way as possible as we can.

To get a well organized category, we have to take into account with the properties of the object itself. For our paper, we develop our work mainly based on the previous work from Famili et al.[59] and Müller et al.[2]. The consideration is the scale of target information. Another perspective is the nature of information structure. It is a natural way of thinking according to the principles as from naive to complicated, from less to mass. This is the way that we are going to organize relevant content.

- **Problems of over-fit**

  redundancy

  A common problem for data analysis activities are desired to excavate only the valuable information from a big mass of data set, which is quite meaningful for some specific purposes or applications. The principle of guiding this selection task is the high relevance with those key data elements. From the work of One successful example for applying this methodology is to extract relevant data from some kinds of expert systems data bases by online learning[60]. By this approach, the expert systems are able to converge the searching area from a wide range into a related narrow spectrum gradually. In the same time, the accuracy of the answer can be improved as well in most cases. Redundant information existed in the whole data base is the priority concern for designing a suitable preprocessing mechanism.

  In addition, the task of reducing the redundancy on a data set, especially on the big data set, can significantly improve the efficiency for the analysis work in later phases. Supposing that if the redundancy rate of some certain big data set is only about 8 percent and the total size of the raw data set is only 100 Gigabyte, the workload will be reduce 8 Gigabyte for subsequent procedures. If the problem suffered by the curse-of-dimensionality, it is quite important to implement a carefully data preprocessing task to remove the redundant high dimensional irrelevant elements in raw data set.

  Noise

  The concept of *noise* in this context is not related to the redundant data elements as we mentioned above. Since the redundant data can even contain some kind of meaningful information comparing with the pure noisy data. However, the noisy data can nearly contain any useful information for the later analysis process.

  There are many potential sources for causing noisy data into the raw data set. For example, the error occurred in the data collection phase can introduce severe data quality problem. Meanwhile, due to some reasons, information loss in some individual data element could happen partially or even entirely during the transmission phase. In consequent, it will result in the noisy data among some contain records. In some circumstances, the record will turn to be meaningless if the integrity has been

violated. For this concern, that data record will definitely turn to be noise in the whole data set.

Noisy data will put different levels of negative influence on the performance of later process applications, which is closely depending on the degree of the noisy. It is obviously that the completely noisy-free data will prevail over the complete noisy data for the same data analysis procedure in the subsequent phase.

Excess

Many extreme big data scenario can cause continuous outliers generation. Due to the hardware or software processing capacities constrains, when input speed overwhelming these processing capabilities, there will lost a lot of information in that circumstance. In data loss scenario, outliers is not the anomaly phenomenon. Inversely, it is difficult to find the normal elements in right positions of that sequent series.

For example, considering such a case takes place in the real time telecommunication systems. When it come across emergency incidence, like terrorism attack as 911 in Unite States, all the people are trying to make call in the same time. Due to the hardware and software constrains, there is no enough telephony exchange capacity available for meeting their requirements. Furthermore, if some hackers intended to make use of this chance to launch telephone system attacks, the system is too vulnerable to sustain those malicious attacks. In addition, the system might not be capable to log all relevant information about intrusion activities by full occupancy of processor and memory resources. In such circumstances, all devices are busy and the response of the entire system are bluntness. A lot of information will be bypassed, including the valuable log files for monitoring system activities.

- **Problemss of under-fit**

Insufficiency

The most typical class of outlier situation is insufficient data set. If no other problems with data elements, it is still can be regarded as outlier for data analysis tasks. For scientifically sound, the experiment of research activities should be reliable in any condition. One of the prerequisites is any conclusion derived from the data analysis process should be on a substantial scale samples. That means too small samples space is also a type of potential risks. Even very few outliers will directly put remarkably influence on that small group of data.

Incomplete

A important type of data problem is prone to occurred in compound data units, which contains several various components like attributes. Those attributes express value of different properties of specific aspects. Missing some attributes of data set can likely lead to very serious problem of the quality of data analysis work or performance for application operations based on that incomplete data input.

As a example, supposing that a stochastic research project is launched to derive some kind of conclusion with importance. If the outliers of missing attributes are counted as the meaningful input as other normal data elements, the conclusion will be probably be mislead in a wrong way than the right results. The root reason is the statistic population of that research problem can not provide enough normal sample points to represent all the characteristics as it should be. Another example is quite common in machine learning algorithm implementations. A highly significant machine learning algorithm is the decision tree[61] [62]. it is quite vulnerable to such a case. The right classification consequences can only be derived by working all relevant attributes by each individual data element. In addition, another attribute-sensitive example can also be found in machine learning field. As an initial procedure for common machine learning algorithm is spending some efforts on the raw data set in the input end. Here we are going to take data set from UCI[63] for example, which is one of the most popular machine learning test set among academy communities and industrial researchers. The raw test data set from UCI is a collection of data records. Every record takes one single line in data file. However, it looks like a mess at first glance. Without splitting it into several meaningful segments in a proper way, it is impossible to conduct machine learning algorithm in later stage. Based on this consideration, the incomplete record (in other word, Outlier) will turn to be impediments or at least introducing some extent of complexity for the whole machine learning process.

Missing

The outlier of missing some types of attributes in data record can occurred problems as mentioned above. However, even if it is intact, it is still possible to cause some problem. To be specific, all attributes are there but some value of attributes are missing. Maybe in some circumstances, people will consider to pick out that entire data record. However, this is the ideal case. In some cases, it is not possible to do that. Since there is very important attributes contain priority value in the same data entry. Otherwise, the performance or result will form bias.

- **Problems of non-fit**

  Granularity

  Couples of sources can produce information for subsequent phase as input data. Consequently, it is regarding to the different level of sources. Every different level of source will generate different type of data either by structure or by granularity. For example, the binary data is the small granularity information and the logic layer information is the more complicated information in granularity. When all these information mixed together unintentionally, it is an outlier for the corresponding process.

  Source

  Nowadays, many companies and organizations are trying to replace the paper documents with electrical files. That means all the information are turning to be digital information. Different purposes, standards, requirements, the information probably

vary from here to there in that enterprise. One certain type of data format requires an dedicate application for analysis. When the multi source situation taken place in that mono format analysis application. It would be result in very serious problem. As a result, the carefully attention should be paid to this kind of outlier in this multi source case.

textbfCompatibility

Compatibility is always a big concern for processing different types of data in a specific application. It is quite common for data collected from different channels. This type of problem results from different way of expressing the information from various places. When the presenting way is going to various forms, the analysis applications are also need to solve the incompatibility problem as first priority.

3.3 Process of data cleaning

From the perspective of data preprocessing practice, it can be regarded as a set of well designed operations on target data set in order to remove the inherent outliers. The output of data preprocessing procedure should obtain a collection of sub set of the origin input data, which contains better quality and more accurate representation capability for whole data set, It is generated by an automatic method rather than manual handling. Since this job of work is more trivial from step to step, people won't have such kind of patience to take it. What is more, sometimes the object data set is quite complicated, both for logical consideration and for scale consideration. It is far away from the scope of expenditure people can afford.

A complete data preprocessing working loop consist of several significant steps as followed: [2]

1. Reformat the data with value concern
2. Enforce the integrity constraint
3. Derive missing property
4. Purge contradiction
5. Merge duplicates
6. Detect outliers

From that list, we can easily find that the outlier detection is the last step in a complete working process. For our own interest, we just put emphasis on this step, which we are going to apply some novel methodologies to improve the performance of final output of detecting outliers than other methods. But it is worth to be bear in mind that the goal for us is to get as pure sub set of data as possible by the outlier detection procedure, in order to facilitate their works. Since it is just a preparation work for subsequent handling process, like data analysis or machine learning applications, etc. That means it is not appropriate to expect that all the anomalies could be detected from this piece of work.

Figure 4: WIP

3.3.1 Outlier overview

Data preprocessing encompasses a large number of specific techniques to detect the outliers in an effective way. First of all, the definition is quite interesting to be highlighted here, which is a principal guideline for instructing the entire working process. Without a clear view on a target, it would probably can not meet the requirements of research purpose.

By the observation with outlying concern, the data elements deviated from other membership elements with a remarkably level in the same data population. From the work of John [64] , we can also regard that some sample points are situated in one class, which actually should belong to another class. By this type of vertical behaviours is rarely to be observed in general. In addition, the definition from Yu's work is also meaningful to be mentioned here [65]. From their perspective, the outliers inside some pile of data set could be regarded as the noisy samples is lying out of a specific set of clusters, which has been defined already. In the same time, outliers could be referred to those sample points, which are right outside of that defined cluster but not the same as the noise. Furthermore, according to the consideration of Barnett [6], outliers should be the inconsistent sample points with the rest of other sample points from the same population in observations.

In the beginning, we are intend to provide a graph, which is an intuitively way to illustrate the concept of outliers essentially. In the following part, we can see the illustration of outliers inside a scatter plot along with other normal samples. 5

In this figure, we can find several sample points away from the circled region. To be specific, there are five outliers in this graph, which as been marked as point **X, Y, Z, V, W**. The samples residing inside is the normal value, which contain very valuable information. And the five distant samples should be removed from the original data set. Actually, this graph is from a survey work from Blake [66]. It looks very easy to identify these outliers by naked eyes of human being. But the real situation is quite different than expectation. Since the computer can not see all the elements at the first glance. It is im-

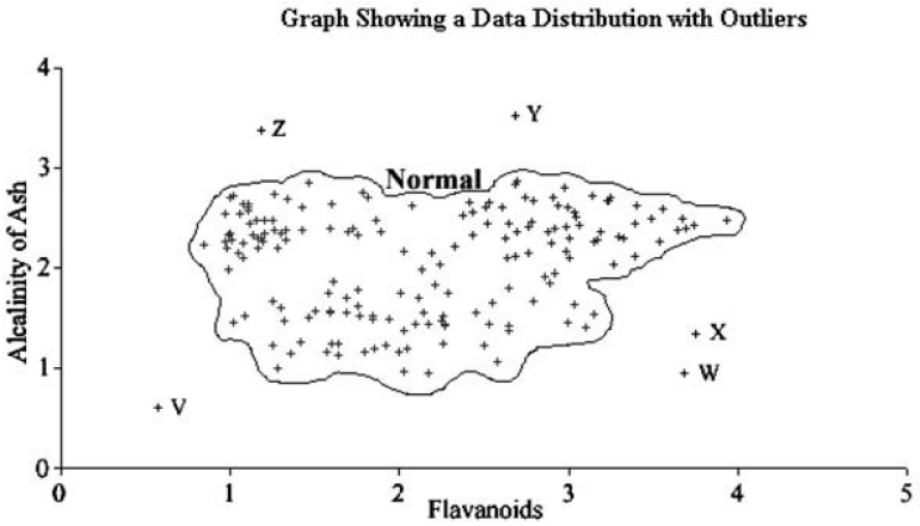

Figure 5: Demonstration for outliers inside a data set

possible to find these five outliers without any effort. People need to development some kinds of methodologies to make it work.

Outlier detection applications or methodologies should be developed in the specific application scenarios as well as some tailored work for better adaptation. There is not universal solutions to detect outliers in various deployment environments.

From the previous works, we can find that there are several typical classes of outliers detection methodologies by the perspectives of technical considerations. They are listed out as followed. [1]

- **Without knowledge**
  This is the biggest challenge for people to conquer in practice. There is entirely no knowledge about the data set, which has to be processed immediately. If we consider this type of problems in the machine learning way, it should be regarded as the instance of unsupervised learning method analogously.

  In front of this challenge, a pile of data set just present there and without any knowledge can be derived before handle it. As a promising candidate solution, it is better to try some approaches from the consideration of statistics scope. In order to apply statistical methodology, the target data set should be keeping in static and can not vary during the whole analysis period. In addition, the data set should be possessed in hand before handling it. It assumed that the outliers are separated from the normal sample set due to some uncertain reasons. As a special case, the main cluster, like the area circled by line, would split into more sub areas.Then the outliers detection

methods for the sub set turn to more challenging than only one-class data cluster circumstances.

However, it is quite interesting that a large number of data collected from the same source, it could be possible to detect the outliers by comparing the later comers with existing ones.

From the work of Rousseeuw [67], we can find two main solutions to mitigate such a challenge. The diagnostic methodology is widely applied into practice to deal with such kind of problem. It used to mark out the potential outliers in a data set. After marking these suspicious data elements, the consequential steps will skip the processing work on them. This is an iterative way of working and the outlying set will be reduced by the diagnostically detection gradually. In the end, all suspicious outliers will be purged from the raw data set. While, an approach called accommodation can be the alternative. It takes the outliers as parts of the generated statistic model with a robust machine learning methods, which is used to realize the classifying function. To be specific, this classification approach can curve the boundary of normal data samples, which contains most of the normal data inside. Meanwhile, it is no vulnerable in front of the interference from outliers. However, it can't overemphasized on the robustness of this methodology here. Because the less robust algorithms will deviate the desirable set of generated normal data elements. It is can only be applied into such situations as just quite few outliers existed in set with majority normal data. For example, from the work from Torr [68], we can find that an inexpensive computation solution has been developed by implementing the least squares methodology.

- **Partial knowledge**
  It is possible to come across such a situation, which the majority is normal data samples and only a very few outliers. It is can be rewritten as a problem of semi supervised machine learning problem to be solved. Due to the normal data set has been labelled by a sufficient size. Then the core problem to be solved is how to detect the outliers among the whole data samples. The prerequisites for such a case is that the normal sample data set should be labelled by manually intervention in advance .

  When the detection process launched, it will learn from the model in an incrementally way by the new arrivals. The purpose of this behaviour is to tune the fitness accuracy for curving a more explicit boundary for normal dataset[69] [70]. It can recognize the new arrivals by the location whether in the premises of normal data set or not. If indeed, that new data could be categorized as the normal. If not, that new data could be detected as the outlier by the learning algorithm. However the problem here is the boundary curved by the learning algorithm seems to be quite strict for real practice. It only works on the situation that the sample data completely outside the scope or absolutely inside the premises. The compensation improvement should be introduced to mitigate such a problem. Fortunately, the algorithm without rigorous constrain on the boundary could be fix this type of problem. In other words, we can regard it as a slack detection on the generated boundary with appropriate estimation.

Another problem for this type of circumstance, it should be no outliers used for training the modelling algorithms. By the well trained detection model, new sample data will be correctly detected as long as located outside the premises of normal set. If the normal data set shifted by some uncertain reasons, it should be retrained to generate an updated detection model for more accurate outputs.

- **Full knowledge**
  In this circumstance, it is always shown up with a set of pure normal sample data with label. If we consider this scenario into the geometric space, we can see that all the points scattering out of the dedicated normal area should be labelled as outliers. Analogously, it could be regarded as the supervised learning model within the machine learning field.

  This methodology is suitable to be applied into the online classification cases. The classifier can compare the online sample data against the supervised learning model, if contradicted, the label could be attached on that suspicious outliers. Otherwise, the new sample data should be accepted as the normal one.

  To construct a robust learning model, the training set should contain enough data across the whole distribution of the data source, which can be able to reflect the inner essence of its characteristic. However, it is worth to emphasize that correctness can be guaranteed by the known distribution. But some of the new sample data could be not so correct, which derived from other area out of the distribution. Except that the representative capability of the learning model is fault-sustainable, the learning output should be suspected under this situation.

3.4 Stochastic criterion

As a challenging problem for the whole research societies and industrial vendors, it is quite usual to come up against a big mass of data, which is very important to deal with in a carefully way. Due to it contains very valuable information in that set. In some ideal circumstance, all the data collected in hand is totally perfect for later use. But in other circumstance, it is not so optimistic. Data set in hand is more or less imperfect for further processing phases in general cases. They are the outliers what we have mentioned before.

Although, there is a large number of various kinds of outliers in real world, such as literal, digital, lexical, logical structure, and so on. But for our research project, we are just going to concentrate on the digital outliers detection problem. It seems to be not a complicated but a more straightforward task. It is still take a crucial place in the data preprocessing field. Since most of the application scenarios have widely introduced the digital form to represent some kinds of meanings or operations. For example, the file system permission expression in all kinds of Unix-like operation systems can be written as a combination of "**R**","**W**" and "**X**", which refers to *Readable*, *Writeable* and *Executable* respectively [71]. However, it is only used for facilitating understanding of common computer users. As another alternative, the file permission expression can also be written in the digital serial consisting of **1**,**2**and **4**.

By proper sum up these three basic elements, it is capable to express any value from 1 to 7. The addition computation is according to the proper combination of different permissions. Meanwhile, it is not the most difficult thing for the computer users. It is more complicated to understand the file system permission issue clearly on directories control configuration [72]. This is just one example for demonstrating the use of digital inside the computer system. We can not difficult find more cases like this. The purpose for us to pick up this example is to show that is even common digital value outlier detection seems naive but it is really quite necessary and useful in many realistic scenarios. Based on this consideration, we spend a lot of efforts on it.

We can find a lot of examples in daily life that a lot of work should be processed by some kind of data in hand. Those collected data is usually containing some kinds of problems with their quality. For example, many experimental results have been collected from some kind of experiments. In consequent, there is a large amount of data related to that specific experiment. The gross results will represent some type of variance rather than the uniform way. As a result, the people should make a decision how to filter out those improper elements with reasons. In other words, those outliers should be rejected by some kind of scientific theory support. For the rest results should be accepted also based on the same sound theory support.

According to the background of data collected from experiments and all of results with digital value, it is naturally to apply the mathematics methodology to investigate this type of problem. Let us consider this case in the statistical way, we can regard all the results as the set of observations on all the experiment process. It is a population formed by all the samples. The outliers can be regarded as the observation on anomaly cases in target experiment.

The main purpose of statistics is focusing on the core tasks of collecting, organizing, analysing, interpreting and presenting a group of data in a mathematically way [73]. From the perspective of statistics, the outliers is a set of elements which deviate from the other elements in gross data set.

For interests of us, there are two alternatives to be considered in front of examining the raw data set [**?**].

...

- An outlying observation may be merely an extreme manifestation of the random variability inherent in the data. If this is true, the values should be retained and processed in the same manner as the other observations in the sample.
- On the other hand, an outlying observation may be the result of gross deviation from prescribed experimental procedure or an error in calculating or recording the numerical value.

...

Back to this piece of work for us, it is to find out some outliers from that massive data set by stochastic methods. The expected output should screen out some obvious outliers

by applying stochastic methodology. Since our outlier detection methodology encompasses multi rounds, rather than those solutions settling a matter in one step. Maybe this sounds like a little bit naive approach. Since the object of our research is aiming at the big data set in the digital forensics application scenarios. Even merely 1% percent improvement on the detection performance should be a great progress for the entire process. Just taking the enormous cardinal into account, this 1% improvement can exempt a quite big computation overhead for consecutive working procedures. If it is able to elevate more than that percentage, more overhead can be exempted from both more expensive computation requirements and bigger time budget reservations.

Many different patterns inside the test data set. raw data set always contains the valid data elements, irrelevant elements, redundant elements, exceptional data elements and so on.

The primary concern for preprocessing the raw data set is to filter the noisy data. Noisy data refers to the meaningless data of some certain data set. In some circumstance, this term is used for corrupted data. For the perspective of software functionality, it is not so difficult to find that those noisy data elements can not be interpreted or understood by machine learning software suites. In addition, There is a very simple but quite intuitive sample to illustrate this concept.

Training samples from the test data set population always contains a large number of valid the data elements for the supervising learning. However, all this work based on the ideal case only occurred in laboratory environment. To be specific, the data set has been constructed in the ideal way. Most of the noisy data have been cleaned with this data set. The discrepancy between different groups are quite apparently and the noisy data nearly can not be found in the training data set. Due to this reason, the learning results from the supervising algorithms became very efficiency and quite satisfactory.

For the above concern, it is not difficult to be aware of that noisy data will take up unnecessary extra storage space than its required capacity. Furthermore, it might put adversely affect on the learning results by machine learning techniques.

We can find a lot of reasons accounting for noisy data.T he hardware failure, sampling error, programming bugs, blind answers of questionnaire, inappropriate operation mode, unstable experiment environment and so on. All these factors will introduce the impediments for conducting machine learning experiment.

Many different methods to deal with this challenge. For example, an typical mitigation is to do the analysis job in statistical way. It can make full use of the information, which gathered from all the collected historical data to pick out the data noise. It will facilitate the subsequent work quite a lot.

However, the naive cleaning methods would not be possible to meet the requirement at the expected level. We try to reconsider this challenge from another angle. Due to the core problem here is how to preprocess the raw data set. To be specific, the key problem

can be rewritten as how to clean the raw data set and filter the inherent noisy data effectively.

Based on above discussions, it is strongly recommended to put our work on a sound mathematics base. The theory of errors is quite benefit for the processing work. Meanwhile, there is a large number of research works conducting among the statistics academy community. It makes more mathematically sound results than going into a blind way.

When some suspicious data generated or collected from all kinds of experiments in the laboratory. According to the conventional terminology, these suspicious data will be called as **Outlier** or **Exceptional Data**. Since they are not the kind of expected experimental output. Naturally, a subsequent challenge here is a decision of retaining or purging such kind of unreasonable data. That is the right field covered by the discipline of theory of errors.

However, several famous statistical test for outliers as followed. They are effectively and applicable in different kind of data test circumstances. The method of dealing with the exceptional data deployed in our research project will be one option in this list.

List of test for outliers

- Grubbs' test
- Dixon's *Q*-test
- PauTa criterion
- Chauvenet's criterion
- F-test
- *t*-test

Here we are going to give a brief introduction to each option for better understanding of those statistic methodologies.

In the beginning, the *t*-test is used to test statistics, which meets the Student's *t* distribution if the null hypothesis supported by the real situation. Generally speaking, this methodology is widely applied into the sample population conforming to normal distribution with the known scaling term. Furthermore, if the scaling term is unknown, it could be replaced by the estimation from the raw data set. It quite facilitates the statistical work in some cases. However, for the perspective of data mining or machine learning field, the test data set is totally random and chaos. Due to the expensively computation work, the *t*-test seems not be a good choice for this project.

Furthermore, the F-test is a test analysis method on the F-distribution, which supports the null hypothesis. However, it has a very obvious drawbacks for our project here, which are the requirements of both normal distribution with the population and same standard deviation with each population strictly. It is quite difficult to meet both requirements in

the same time.

What is more, Chauvenet's criterion is used to assess the data elements in the raw data set whether an outlier or a valid one by a set of empirical observations. It computes the mean value and standard deviation for the population in the beginning phase. The next step is adopting the normal distribution for determining the probability how much a data element would be suspicious as an outlier. The base for this evaluation work is the difference between the suspect data and mean value of the population.

Moreover, PauTa criterion is well known as the nearly all the data sample will distribute within the scope of 3 times of standard deviation around the mean value (or mathematics expectation of that normal distribution population). This is valid for all the cleaning task. But the output of this method is not so efficiency. In other words, there is still a large number of noisy sample inside the population, even the samples out of the scope has been purged. Another problem for this method is the population should be assumed to follow the ideal normal distribution in advance.

In addition, Dixon's *Q*-test is a very famous methodology to deal with the errors. It is a type of significance test in statistics field. The interesting point is the test can not be used more than once in testing population. It computes the *Q* value by the increasing ordered samples by dividing the value of *gap* by the value of *range*. If the individual *Q* value is greater than the reference value, which is corresponding to the confidence level and samples, the single sample is rejected as an outlier. It is commonly deployed to the case of measuring a set of experiment data and purging the outliers under such a circumstance.

Last but not least, the Grubbs' test for outliers detection is highly recommended by the statistics societies. It is quite common to be used by processing the outliers from a univariate data population, which is in assumption of conforming a Gaussian distribution in advance. The outliers can be purged from the raw input data set iteratively. That means the procedure will not stop until no outlier left in the population by this rule. Notably, the size of population should not be too small, in which case most of data elements would be recognized as outliers with high possibilities. Fortunately, it is not a big problem in our case. Due to we have to deal with the big data scenes, which happens to avoid that drawback.

3.4.1 Grubbs' test

An outlier can be regarded as a far-deviated element from the normal date collection. In many cases, It is quite important to adopt a more cautious perspective to review those outlying elements. From the statistics perspective, it should follow such a procedure in below to screen the outlier observations among a random population.

- Reject an experiment observations
- Correct experiment observations (Statistically)
- Reject suspicious data and conduct more observations

- Handle the suspects with theory of truncated sample

In order to examine a set of outliers among a group of statistic observations, it is inevitable to introduce some kind of value to make a judgement on the testing hypothesis. It is a threshold of determining one doubtful observation either to discard or to accept, which is based on the random sampling theory. The rejection decision will be made when the observation results exceeded that threshold, otherwise the acception decision will be made.

The critical value represents the criterion of the random samples from a series of statistic observations, which conforms the same distribution, population or source. It can be interpreted as the small possibility for the sample criterion to exceed that threshold. This can be called as "Significant level" in Grubbs' test, which can be regarded as the potential risk of rejecting a normal samples as the outliers. The lower significant level can achieve better final results, while the higher significant level (for example, greater than 5%) is not recommended in practice.

The outlier criteria is based on the assumption of Gaussian distribution based population, which is underlying all the statistical work for detecting outliers. It means that it should not be misuse this approach without confirming the distribution model in advance. Otherwise, the results will be invalid for further statistical analysis.

Definition

For a set of observations from the random samples, they can be denoted in an increasing order as $X_1 \leq X_2 \leq X_3 \leq X_4 \leq X_5 \ldots \leq X_i \leq \ldots$. Assumed that the size of the observation set contains N elements. Then, we can express our statistic population as $S = \{X_1, X_2, X_3, ...X_n\}$. For the most suspicious samples among these observations, it should be either the smallest one or the biggest one, which located in the end of this sequential series.

The criterion can be expressed as followed 3.2, which is only valid to examine the right side outliers, which means the most biggest sample observation value.

$$G_i = \frac{x_i - \overline{X}}{S} \tag{3.2}$$

If the left side outlier detection need to be implemented, the corresponding criterion adopts another formula 3.3.

$$G_i = \frac{\overline{X} - x_i}{S} \tag{3.3}$$

Where,
$\overline{x}$ 3.4, denotes the mean value of the observation population.

$$\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{3.4}$$

$\overline{S}$ 3.5, denotes the statistic estimate of standard deviation in the observation population.

$$S = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2} \tag{3.5}$$

We have to point out that the critical value is 5 % level of significance in our later experiment, which denoted by P there.

Demonstration

Here it is an intuitive example to illustrate how the Grubbs' test procedure finding the outliers in the a Guassian-based random data set.

Supposing that there are several value has been collected from the observation from the experiment process, like the following list 2.

Number	1	2	3	4	5	6	7	8	9	10
Value	7.7	8.2	6.0	5.4	4.7	10.1	14.0	6.5	7.3	9.0

Table 2: Experimental data from random measurement

However, For implementing Grubbs' test, it needs to reorder the data series by an increasing sequence for the subsequent work. Correspondingly, we have to rearrange the elements in list from the smallest to the biggest as the following table 3.

Number	1	2	3	4	5	6	7	8	9	10
Value	4.7	5.4	6.0	6.5	7.3	7.7	8.2	9.0	10.1	14.0

Table 3: Sequential data array

Then, we have to compute the mean value $\overline{x}$ by formula **??** for all the elements in this as well as the computation of standard deviation by formula 3.5. By the computation with the value $n = 10$, we can get the mean value $\overline{x}$ equals to 7.89, while the value of standard deviation equals to 2.704. So far it is a right time to calculate the deviation value for end elements. For instance, the deviation between the minimum sample and the mean value is 3.19. It derives from the difference between 7.89 and 4.7 as the leftmost element in the list. Meanwhile, we can also get the deviation between the maximum sample and the mean value equals to 6.11. Similarly, this can be derived from the difference between 7.89 and 14.0 as the rightmost sample in the population.

The following steps intend to determine the suspicious samples by the above computation. Comparing the leftmost discrepancy and rightmost discrepancy, it is not difficult to find that the value of 6.11 is greater than the value of 3.19. This could be interpreted as the rightmost element in this list is more suspicious among all the samples in the origin data set.

Right now, it needs to compute the criteria value for the rightmost element in this list by the formula.3.2 We can obtain the value of i equals to 10 due to its position of 10 in

that list. And this value equals to the value of 2.260 by dividing the deviation between the maximum sample and the mean value of 6.11 with the standard deviation value of 2.704. Then it needs to look up the Grubbs' table to determine whether this value be accepted or rejected according to the confidence interval. If the computed value is greater than the corresponding value shown in the Grubbs' table. It would be an outlier with high possibility.

Here is part of the Grubbs' table shown as followed.6 In this case, we assume the test level with the value of 0.05. Consequently, the confidence possibility equals to the value of 0.95 (1-0.05=0.95). Naturally, we can look up to the value equals to 2.176 from the given Grubbs' table. By comparing the value of computing result of 2.260 with the table value of 2.176, we can reject the suspicious element by the comparison. As a result, we can remove it out and the same iteration can be performed on all the rest elements in the updated list. The iteration procedure can be customized by setting the proper parameters for automatic processing by computer program.

Grubbs Table --- Criterion Gp(n)

n \ P	0.95	0.99	n \ P	0.95	0.99
3	1.135	1.155	17	2.475	2.785
4	1.463	1.492	18	2.504	2.821
5	1.672	1.749	19	2.532	2.854
6	1.822	1.944	20	2.557	2.884
7	1.938	2.097	21	2.580	2.912
8	2.032	2.231	22	2.603	2.939
9	2.110	2.323	23	2.624	2.963
10	2.176	2.410	24	2.644	2.987
11	2.234	2.485	25	2.663	3.009
12	2.285	2.550	30	2.745	3.103
13	2.331	2.607	35	2.811	3.178
14	2.371	2.659	40	2.866	3.240
15	2.409	2.705	45	2.914	3.292

Figure 6: Grubbs' Table

3.5 Machine learning

Among all the other most popular machine learning algorithms, such as C.45 algorithm[61], K-means algorithm[75], Support vector machines[76], Apriori algorithm[77], EM algorithm[78], PageRank[79],AdaBoost[80], Naive Bayes[81],CART[82]. K-nearest neighbour classification algorithm (*k*NN)[83] [84] is one of the implementation-friendly approach relatively. It is quite suitable for implementing the second round of preprocessing task immediately after executing Grubbs' detection.

The advantages of K-nearest neighbour algorithm are quite apparently. It is able to offer more satisfactory generalizing capability [85]. In addition, for the large scale data scenarios, the error rate of KNN algorithm can be able to achieve not more than twice times, comparing to Bayes classification algorithm. Even that Bayes classifiers have been optimized for specific cases appropriately.[86] This a quite good merit as an implementation-friendly solution with a satisfactory quality of results.

The following considerations for us to choose a proper machine learning algorithm to

implement the clustering task in data preprocessing phase.

- It should fulfil the task of clustering data elements.
- It should require not much knowledge as prerequisites.
- It should be no expensive computational requirements or complexity relatively.
- It should achieve relatively satisfactory performance on output.

Counter-example

If we just consider the problem of clustering target sample elements properly, it seems that K-Means is also a good candidate solution. Due to it is unsupervised learning approach. Meanwhile, it is also not difficult to implement, the same on distant-metric principle. Based on these advantages, it seems better than our selection as K-Nearest Neighbour algorithm. The decision made by us is based on the following considerations.

The main pitfall for K-Means algorithm is the predefined value of **K**. Let us put this decision behaviour into the preprocessing scene. In front of the massive data set, we have not so much knowledge about its nature. What we know is that it contains the normal data samples, which we are desiring to make full use of. Referring to the outliers, we do not have so much knowledge about it, neither the number of outliers nor the distribution of the outlying subset. According to this algorithm, **K** is the number of seeds in raw input data set, which is selected in a random way. Without a proper **K** value, the clustering would go into a wrong way. The results are likely to be quite different than expectations. That is why it is the crucial challenge for us to select an appropriate value for a specific **K** in advance. Although, there is a relatively pleasant solution for mitigating such an embarrassment by ISODATA [87] algorithm. From the work of Ball and Hall, we can find that a relatively appropriate value of **K** could be obtained by automatic merging and splitting procedures.

The second consideration is the selection of initialization points for launching the K-Means algorithm. It is so important that the final clustering results will vary with extravagantly degree depending on different initialization points. In some circumstance, it would be total different from one to another. As a result, it is also a crucial challenge to select an appropriate set of initialization seeds for satisfactory output. However, K-Means++ algorithm [88] [89] is capable to conquer this problem. From the work of Ostrovsky et al. and Vassilvitskii et al., we can see that the cluster central points could be yielded by minimizing the variance between intra classes. It is still a good approximating solution so far. Even though the precise selection for arbitrary data set is still a NP-hard problem [90]. What is more, K-Means algorithm itself is vulnerable to outliers sensitively.

3.5.1 K-Nearest Neighbour

Comparing to other machine learning algorithms, K-Nearest Neighbour requires nothing more to work normally, except the distances between each other in the data set, which is quite the same as K-Means algorithm. However, the distance variable is not difficult to get in any case. The final clustering results are intimately depending on the distances. Sometimes, the distance-based approaches are also referred as a type of "Lazy" method-

ology when comparing with other algorithms.

There are several kinds of machine learning algorithms, which work on the distance parameters in the data set. In contrast to other more sophisticated machine learning algorithms, The distance-based approaches seem to be more straightforward conceptually. In order to work, it requires to retrieve the distance value, store in somewhere, then classify the data elements properly in target data set. Even for this lazy methodology, it could be more complicated to represent the distance in a more advanced and complicated way. For example, the utility of "neighbour" elements related to more complicated structures in specific algorithm.

As a sophisticated approach, K-Nearest Neighbour algorithm is intending to find a set of *k* elements in the training data set, which are the most closest to a dedicated sample data in the test data set. The voting mechanism for labelling a testing data is depending on the predominance of a special class in the neighbourhood. It requires three conditions to implement this approach to cluster the test data correctly. In the following paragraphs, we are going to discuss it in detail.

Given a data set S for training in later phases. Mathematically, we have $s_i\,(m, n) \in S$, while m denotes the training sample and n denotes the label of sample m. Meanwhile, a test sample should also be provided. Let us write in the form of $t = (m', n')$, while m' denotes the test data and the n' denotes the class label of the test data. Afterwards, the distances between sample, denoting by p, and all training sample in set S will be measured (denoting by D_p) by this algorithm in order to classify it correctly by its K nearest neighbours.

Definition

According to classic methodology, Euclidean distance has been widely applied to measure distance in K-Nearest Neighbour algorithm. Mathematically, Euclidean distance can used to depict the distance between elements in vector space.

Given a multi-dimensional vector space with cardinal of n. Then it could be denoted by R^n. Now, any point in this space is corresponding to a dedicated vector of $\overline{V}$ 3.6

$$\overline{V} = \langle a_1\,(x)\,, a_2\,(x)\,, a_3\,(x)\,, \ldots, a_n\,(x)\rangle \tag{3.6}$$

Here the vector attributes like $a_i, i \in [(1, n) \cap Z^+]$. Then the distance between 3.7 two points can be expressed as followed.

$$^{d_{ij}=}\sqrt{\sum_{k=1}^{n}\left(a_k\,(x_i) - a_k\,(x_j)\right)^2} \tag{3.7}$$

Since the labelling mechanism coincides to the majority in its K nearest neighbours. As a result, the determination representation 3.8 can be written as followed. [91]

$$n' = argmax_l \sum_{(m_i, n_i)\in Dp} \delta\,(l, n_i) \tag{3.8}$$

For δ 3.9, it denotes that,

$$\delta(u, v) = \begin{cases} 1, u = v \\ 0, u \neq v \end{cases} \tag{3.9}$$

Here we have, i denotes for the i-th element. l denotes for the label of different class. n_i denotes for the i-th nearest neighbour's class label. Function δ is used to return the positive vote if the conditions founded. Otherwise, it will return the negative vote for the class label to be attached.

Right now, the K-Nearest Neighbour algorithm consists of the following key factors to run, if from the perspective of global view [18].

- **Input**
  It consists of the training set, test data,
- **Measurement**
  It performs the computational work to measure distance between test data and all training samples.
- **Output**
  The decision of class label to be attached on that test data by formula 3.8

With the above clustering process, we are able to attach correct class labels to the test data. For example, the red dot in the following graph 7 is the test data to be labelled. While in the training data set, we have two types of different classes. One class is denoted by blue stars, while another one is denoted by green squares. If we choose K with 3, then the red dot should turn to green square. However, if we choose K with 7, then the red dot should turn to blue star.

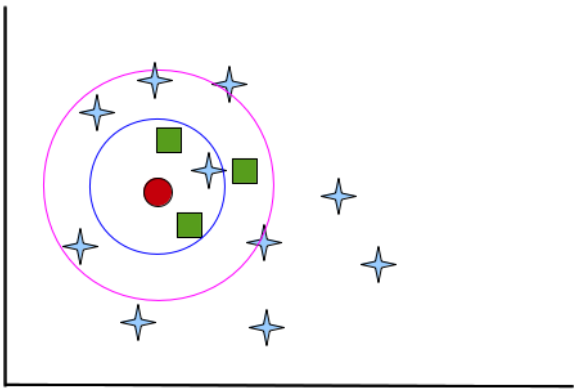

Figure 7: K-Nearest Neighbour Model

Discussions

However, the flaws should be mentioned as well as the merits talked above. The drawback for distance-based clustering mechanism is the cost of computation burden. However, this flaw can be overridden in some circumstance. For instance, if no other machine learning algorithms can work properly without extra information from the data set. That

means even the price for implementation such an algorithm, it is still worth to do it rather than nothing to do. It is also a kind of requirement calling for further improvement by the entire research communities.

Another remarkably pitfall is to select the appropriate value for K. It is closely related to the sensitive level of those K-elements data groups. If the K set with a small value, it would probably be noise-intolerant, which will easily give rise to the false positive problem. In contrast, if the value of K is set so big, it will probably give rise to the problem of false positive. In that circumstance, the group of majority elements' class label will-predominate the classifier to label real outliers correctly. We will discuss this problem further in the following part.

What is more, the choice of metrical distance is also a crucial consideration for K-Nearest Neighbour algorithm. It should be carefully selected according to specific circumstance. For example, different kinds of distances take different approximation ways 8 to the centroid point. That means when selecting different distance should take their different approximation way into consideration as well. The principle of selecting an appropriate distance for accurate K-Nearest Neighbour algorithm is that even a tiny difference on distance can also soundly indicate the high possibility of same class.

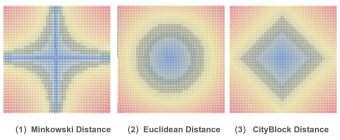

Figure 8: Approximation Way of Different Distance

However, even for computing a specific type of distance, there would be also likely to have some problems. For example, for computing the Euclidean distance by formula 3.7. If the order of magnitudes some attributes are overwhelming the others, the influence of those be overwhelmed attributes will be eroded. That is, if $a_i \gg a_j$, then $(a_i - b_i)^2 \gg (a_j - b_j)^2$. So for the Euclidean distance formula 3.7, the i-th attribute is predominating over all the other attributes. Coincidentally, the distance between vectors should take the less predominating attributes, such as a_j , as the priority consideration for K-Nearest Neighbour classification.

Improvement approach

The primitive K-Nearest Neighbour classification suffered some problems. The accuracy is severely relying on the raw distance value, which is a vulnerable in some circumstance. Some improvements can offer a better performance than the primitive KNN. The main idea is to scale each distance value with weight respectively. It is called as the weighted K-Nearest Neighbour algorithm.

There are many publications about the topic of adding some kind of weight to primitive K-Nearest Neighbour algorithm. For example, Weighted K-Nearest Neighbour algorithm surpasses high dimensional application application scenarios, when only quite few

data is available to use. [92] [93]. While, the recently research interests in the academy communities are concentrating on the independent weighting scheme without considering the dedicated distributions.[94] In addition, the kernel functions are very popular among the research societies as well [95] [96] [97]. Furthermore, some weighting mechanisms are aiming to the distance between nearest neighbouring elements and query samples [98] [99].

Every different class sample puts influence on its neighbouring data elements. However, the influence can be regarded as a uniform distribution. If we are weighting all the influence of every individual data element, it could be expressed as followed. 3.10

$$\forall s_i \in S_1, \exists F(s_i) \equiv 1 \tag{3.10}$$

Where, the function F is the influence index or "*Weight*" for each data in the training data set.

Kernel function

For the perspective of history, the kernel function theory has a long history. Many talented mathematicians and researchers made a large number of fruitful achievements in this field. Especially, it significantly leveraged the development of machine learning discipline. The most famous example to illustrate this is the Support Vector Machine methodology. The kernel function for SVM algorithm remarkably facilitates the processing capability with high dimensional problems in machine learning field.

The main principle of kernel function approach is to conquer the curse of dimension challenge. In some circumstance, the kernel function approach is also called as kernel trick in machine learning field. According to the machine learning practice, object data set can be split linearly in higher dimensional feature space, which can not make it in the origin dimension feature space. However, it will lead to several problems in the higher feature space by applying this technique to execute machine learning directly. For example, the representation of mapping function and its parameters can not be confirmed. While, the cardinal of feature space also can not be determined. In addition, the biggest problem is the curse of dimension, which is totally a disaster for computation occurred in higher feature space. With the help of kernel function approach, these challenges can be mitigated to a notable extent.

Definition

For $x, z \in X, X \subset R^n$, non-linear function $\phi(\cdot)$ turns the mapping transformation from space X to feature space F. The kernel function given out as followed.

$$K(x, z) = \langle \phi(x), \phi(z) \rangle \tag{3.11}$$

Where it holds : $F \subset R^m, n << m$ and $\langle \cdot \rangle$ is the inner product.

From above formula 3.11, we can see a subtle connection established between the lower space and the higher space, which successfully evades the curse of dimension problem. From the perspective of machine learning, it laid the solid theoretical foundations

for resolving the complexities of classification or regression problems in high dimension feature space.

Characteristics

The kernel function approach is so useful that it can be applied in a large amount of domains. We can find so many example everywhere. It is mainly due to its unique merits as followed.

- No knowledge required about the non-linear transformation function $\phi(\cdot)$ in details, such as the representation, parameters, and so on.
- The computation work load has significantly reduced in high dimension scenarios. The most ingenious thing is the origin cardinal has taken no influence itself.
- The variance of kernel functions will result in the change of the mapping relationship between the input space and the higher dimension feature space, which will radically change the performance in the end.
- It can compound other different algorithms to develop more novel approaches with better performance.
- The Mercer theorem [100] governs the selection of an appropriate kernel function. Any function meets all the conditions of Mercer theorem can be used as kernel function.

In order to weight each distance appropriately, it is necessary to transform the distance result into the index of similarity. The input for this approach is the distances measured by computing Euclidean distance before. Then, we need to the find an appropriate kernel function to accomplish the goal for transformation, which is from distance dimensionality to probability dimensionality.

Mathematical representation

It is significant to give out some criterion about the desirable kernel function. According to the work from of Hechenbichler et.al,[97] , we can see that :

- $$\forall d_{ii} \in D, \exists K(d_i) \geq 0 \tag{3.12}$$
- $$\forall d_i \in D, K(d_i) \leq K(0) \tag{3.13}$$
- $$\forall d_i, d_j \in D \cap d_i < d_j, \exists K(d_i) > K(d_j) \tag{3.14}$$

With above constrains, we can see that the kernel function for transformation should be always voting a non-negative value per distance. It can guarantee the meaningful for every weight. While, the maximum vale can be yielded only at the same point, which is soundly support the transformation principle of *More Closer, More Similar*. What is more, the kernel function is monotonously descending in a rigorously way. By the guarantee of rigorous monotonously descendant, the ranking mechanism for selecting the most closest K nearest neighbours can work efficiently.

Based on the above consideration, we can find several different types of mathematics kernel functions as followed.

- Inverse

$$K(d) = \frac{1}{d} \tag{3.15}$$

- Gaussian

$$K(d) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2} \cdot d^2} \tag{3.16}$$

- Cosine

$$K(d) = \frac{\pi}{4} \cdot \cos\left(\frac{\pi}{2} d\right) \cdot 1_{\{|d| \leq 1\}} \tag{3.17}$$

- Epanechnikov

$$K(d) = \frac{3}{4} \cdot \left(1 - d^2\right) \cdot 1_{\{|d| \leq 1\}} \tag{3.18}$$

Where, the function of $1_{\{|d| \leq 1\}}$ is the indicator function for the valid domain of d. If the value loactes in the scope, it equals to 1. Otherwise, it equals to 0.

Although, there are many other types of kernel functions. We just list out the most relevant ones for intuitive demonstration. For instance, the cosine kernel is quite useful in text classification. It outperforms all the other kernel functions on that case. In order to conform the purpose of preprocessing, we should choose the most easier implemented approach as well as lower computation expenditure.

Consideration

After a carefully consideration, we decide to deploy the naive distance-based K-nearest neighbour algorithm rather than utilizing kernel function to perform the transformation task. From the above mathematics representations 3.15 , 3.16, 3.17, 3.18, we can aware that they are all involving with the complex mathematical computation. The time overhead may seem not a very big deal for common cases. However, taking our own scenario into consideration, the sum of time overhead in computing these kernel functions in big data set will be a considerable value. Multiplication operation takes more time than pure addition operation inside the computer system, let alone the advanced function computation like $\cos$ or e^x computation.The big(O) of computing these kernel functions are much bigger than those basic operations. If the volume of elements to be computed is quite a big number, the deeply influence of time overhead can not be ignored. Ultimately, they will turn to be a heavy burden for fast computing within a given time buffer.

Another consideration for making this decision is based on the mission of our research. All our work is aiming to generate a relatively smaller-size and more-valuable data set by applying preprocessing techniques. This product of our preprocess operation should remove the noise and other valueless information from the raw source, without causing the valuable-data loss . That means all the valuable information should be preserved as much as possible. It is really a dilemma to make a proper trade-off between purging and preserving. Kernel function is one of the key factors to achieve better performance. But it can be implemented in the later stage. For example, it can be executed in the machine learning based applications for automatic evidence analysis activities, which

the input is prepared by our methodology.

In summary, we intend to hold a conservative altitude to process the raw data set. Two main concerns in our mind are relatively inexpensive computation overhead and less evidence loss exceptionally.

4 Empirical Implementation

The chapter intends to illustrate the details of implementing data preprocessing work for large scale digital forensics investigation. The main goal for our methodology is to detect the inherent outliers from the raw big data set without causing exceptionally evidence loss. While, the proof of concept experiments are executed in order to verify our proposed approach into practice.

4.1 Experimental environmental setup

For implementing our data preprocessing work, we have to rely on substantial physical resources and corresponding software facilities.

4.1.1 Framework

The framework for our data preprocessing methodology is mainly consists of two significant phases. As the picture 9 shown, The first round is implementing the stochastic approach to detect the outliers by Grubbs' criterion. The immediate processing stage is to detect the remaining outliers by applying K-NN machine learning algorithm. We can regard that the first round of detection by Grubbs' test is aiming to filter out those obviously anomaly data. Since when the anomaly data varied far away from the centroid point, we can use standard deviation σ to reflect such cases. While, the K-nearest neighbour algorithm is good at classifying the testing data with the distance-base measurement on most closet training elements. So the design of this working framework is reasonable and suitable for handling our problem.

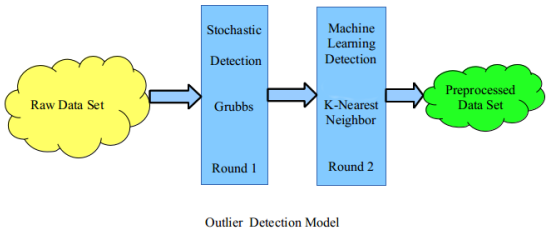

Figure 9: Outlier Dection Model

4.1.2 Hardware configuration

The data preprocessing task for large scale digital forensics investigation on big data set is really a tough problem. The first challenge is a high computation capacity demanding task. However, most common personal computer can not be able to meet this requirement completely. In order to demonstrate a representative example, we did not choose the most powerful physical devices in our experiment. Or we can almost say that this

hardware configuration for experiment can be easily surpassed by most recent type of model computer products. Meanwhile, comparing to Microsoft Windows™ operation systems, Linux systems take lower hardware resource consumption by itself. What is more, we select the python language to develop our prototype, which is one of the most popular programming languages among the research society and especially predominated in scientific calculation.

The hardware configuration of our experimental environment has been listed in the following part:

- OS : Ubuntu 12.04 LTS, Desktop
- Version : Linux-x86_64 bit
- CPU : Intel Core™2 Duo Processor T5750 (2M Cache, 2.00 GHz, 667 MHz FSB)
- Memory : 2048 MB
- Hard drive : 40 GB
- Programming Language : Python 2.7.3
- Programming IDE : ActiveState Komodo IDE, version 8.0.2

4.1.3 Data set

First of all, we have to mention the application scenario we have to handle. Primarily, the input of our example is a big data set conceptually. However, in order to demonstrate a prototype of our methodology, It is appropriate to scale our problem to a relatively smaller size. We going to take a group of 100 samples as the input for our experiment. The reason for us to go in this way is based on the following considerations.

Initially, Let's assume that we already have a Petabyte-size data set in hand. It is impossible to handle such a big mass of data by most computers. Then, a reasonable solution for handling this case is to properly divide this raw input into many smaller scale packages. Afterwards, these numerous packages can be processed by one computer consecutively or processed by clusters of computer concurrently. However, this approach has already deployed widely in our life. One of the most famous example is the concurrent processing service running in Google search engine.[101][102][103]. Similarly, it is also reasonable for us to do that.

Meanwhile, It is appropriate to construct a set of data with a considerable size in order to maintain the nature of origin data source in that big data. Since too small data collection may be deficient representation of the inner characteristic of origin data set. Then, we consider that 100 sample set is enough to illustrate the prototype of our method. The more sophisticated improvement can be done as the future work for those interested readers.

In this piece of work, we are going to adopt random data set, which is approximating to Guassian distribution with a given scope. This is based on the consideration of statistics principle. According to central limit theorem, (in abbreviation, CLT)a sufficiently large number of independent random variables, each with a well-defined mean and variance,

will be approximately normally distributed. Variant forms are also founded if only the mean converged to Gaussian distribution. [104]. This offered a theoretically foundation. Meanwhile, in realistic world, data is always generated by some certain type of source, which is likely to comply with some type of distribution. While the Gaussian distribution is the most common distribution for expressing the mathematical characteristic of those data collections. This also another significant consideration for us to make such a decision.

The base test data with 100 samples are generated by python programming language. It is a random sequence, which is comply to the Gaussian distribution. As the scatter plot shown in the second quadrant of that picture10 , this distribution holds the mean value $\mu = 0.2$ and the standard deviation $\sigma = 0.1$.

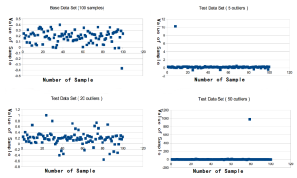

Figure 10: Scatter plot of Grubbs' outlier detection test sets

4.2 Experiment execution

The experiment execution is focusing on the verification of the availability of our method, whether it works or not. Meanwhile, we have implemented several trials with different parameters in order to illustrate the validity of our approach, which is also in the purpose of proofing its inner properties from our methodology.

The proof of concept work consists of two steps consecutively, which is strictly follow the working process of the framework. The first round is to detect the inherent outliers by Grubbs' test. Afterwards, the output will be inspected and handed over to the input side of K-nearest neighbour learning algorithm. During in the experiment period, all relevant results will be well collected.

4.2.1 Grubbs' Detection Implementation

In terms of the Grubbs' test formula, it contains two closely-related key variables inside. The first one is the value of N, which is standing of the testing group size, which is also the total number of elements in one test sequence. In addition, another key factor is the value of P, which is standing for the significance level, sometimes it is also denoted by α in some publications. According to the Grubbs' critical value table 14, we can see that the smaller value of P, the higher value of critical value. It could be explained that the higher significance level we set, the more critical to the value of Gi. $P = 0.05$ is an appropriate setting of value. In our research work, we are going to examine the various setting of N and check different level of influence on the each corresponding outputs.

Data preparation

We decide to use the 100-elements data set for testing the Grubbs' detection performance. As illustrated in the following table 4, We have launched nine Grubbs' detection experiments with different parameter setting. The "X" symbol means the implementation will be done in our experiment.

Grubbs' detection test data set (100 elements in total)			
P=0.05	N=100	N=50	N=20
50-Outlier	X	X	X
20-outlier	X	X	X
5-outlier	X	X	X

Table 4: Grubbs' detection test data set

However, we have done some manipulation work on the base data set, in order to introduce more randomized properties on the test data set for each group of detection experiments. As the scatter plot shown 10, these data sets for each group of detection experiments contain more diversity than others. For the plot in the first quadrant of this picture, the outlier value of 979.20 makes the sharp contrast to plots in other quadrants. Meanwhile, the plot in the third quadrant of this picture gives a more intuitive illustration on its nature distribution. For the first and fourth quadrants of this picture intend to demonstrate the the existence of extreme outlying elements in that test data set.

It is necessary to explain selecting value of samples in our experiment, which is mainly in the range of $[0, 1]$. According to digital forensics practice, normal accessing mode in file system will finally illustrate some kind of fixed pattern after a relative long period and for some locations nearly never been touch by common users [47]. The actions occurred in computer systems convey a lot of information to the digital forensics investigators, such as file accessing times, login frequency, and so on. Normalizing these features by a proper reference value, we can derive an expected interval. Normal operations should be fallen into that domain, which represents by the normalized value. As a result, the malicious behaviours could be detected as outliers in this circumstance.

Grubbs' Outlier Detection

The task of outlier detection in the first round executed by the python program in an automatic way. We assign the input test data set for that detection program in the beginning stage. Then that program will detect the outliers by iterations in terms of the principle of Grubbs' test. The test and run-time environment based on ActiveState Komodo IDE in Ubuntu. There is an illustration 11 for the ongoing outlier detection operations by Grubbs' criterion as followed.

The Grubbs' detection program is developed in python programming language. Here is the a part of code shown in below, which is responsible for making a final decision of the suspicious outlier detected by predefined Grubbs' critical value. The subroutine criteria receives the passing parameter and returns the comparison results on Gi and Gp for making a judgement.

```
#####################################################################
```

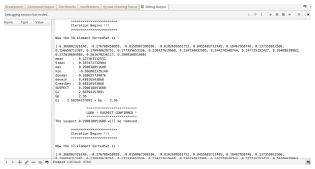

Figure 11: Execution of Grubbs' detection application

```
# g  -> derived vaule of Gi from current data set
# Gp -> predefined value of P and N.

        def criteria(g,Gp):
            if g>Gp:
                print 'Gi : ',g,'>','Gp : ',Gp
                print '''
                *************************
                LOOK ! SUSPECT CONFIRMED !
                *************************'''
                if cmp(minimum,g)==0:
                    print 'The suspect', minimum ,' will be removed.'
                    del SortedSet[0]
                else:
                    print 'The suspect', maximum, 'will be removed.'
                    del SortedSet[-1]
                    return True
            else:
                print 'Gi : ',g,'<','Gp : ',Gp
                print '''
                *************************
                Congratulations ! Seems no suspect...
                *************************'''
                return False
#########################################################################
```

Test Results

The complete detection results of each group can be found in the appendix part. We provide a brief explanation on those tables 12, which its data is derived from the experiments.

In those test reports, all the highlighted cells with different colours denote outliers by the previous manual manipulation. While, the bold figures denotes for the detection output by executing Grubbs' detection application. For example, those bold figures also with highlighted background mean correct detections by applying Grubbs' detection approach. However, for those bold figures without highlighted background mean false detection result by applying Grubbs' detection approach. They are false positive error generated by this approach. The detailed performance analysis of experiment results will be conducted in the following chapter.

4.2.2 KNN Detection Implementation

According to the working framework of our methodology, the second round outlier detection is based on the machine learning techniques. In our experiment, the K-nearest

neighbour algorithm is applied to implement further outlier processing. This approach is aiming to remedy the deficiency of Grubbs' detection methodology. The expected output of executing this detection procedure should improvement the final precision of detection results.

Data preparation

The input data for implementing the K-nearest neighbour algorithm is from the output of first round detection by Grubbs' detection program. Based on the optimization consideration, we will choose the outperforming output from first round. Then K-nearest neighbour algorithm will continue the outlier detection processing task.

In the other hand , we will go to implement several rounds of K-nearest neighbour detection with different value of K, in order to depict their different influence on the final detection results.

Moreover, We will manually attach corresponding labels to a subset of samples from the same test data set, in order to make a training set for implementing K-nearest neighbour detection algorithm. Due to K-nearest neighbour is a supervised learning approach and it must work with the correct training set. As the table shown 5, the training data set used for K-nearest neighbour detection contains 20 labelled samples, which consist of 11 outliers and 9 normal elements. The expected normal samples randomly distributed and approximately converged to a Gaussian distribution with $\mu = 0.2, \sigma = 0.1$. While the class label "1" denotes the data belonging to the normal data class and label "0" denotes the data belonging to the outlier class. In the execution stage of K-NN detection, all the test data will be labelled in the same way when they checked by this type of machine learning algorithm.

In the same time, the testing set consists of 80 elements in total. It is also a subset from the same data set, which the complement is the training set we discussed before. This type of setting can guarantee the learning process running correctly.

KNN Outlier Detection

This round of outlier detection by applying K-nearest neighbour algorithm is the supplementary step for the initial Grubbs' detection. Mathematically, The whole data set consists of two subsets, which strictly holds the relationship of mutual complementary set for each other. It is the prerequisite for all the supervised machine learning approaches, including the K-nearest neighbour approach.

We develop the program to realize K-nearest neighbour algorithm to execute the detection task in an automatic way. The training set with 20 manually labelled samples has been defined in the source code. The testing set is a collection of 42 outliers and 38 normal data. However, when they used as one of the inputs for our K-NN detection program, these 80 elements have no label attached. During the whole detection experiments, we have verified several different value of K in this algorithm in order to yield the average performance for K-NN detection approach.

Training set for KNN Detection (11 outliers, 9 normal elements)		
Element number	Sample value	Class label
1	0.246636021	1
2	-0.160113788	0
3	0.210831019	1
4	-3.090334241	0
5	0.30912883	1
6	5.19187195	0
7	-0.185123008	0
8	0.736757547	0
9	0.032929701	1
10	0.457869706	0
11	-0.236629796	0
12	0.209487151	1
13	0.063040367	1
14	1.038088677	0
15	0.217809562	1
16	0.829951166	0
17	0.360353945	1
18	-0.344817988	0
19	1.32130244	0
20	0.052086917	1

Table 5: Training set for KNN detection algorithm (20 samples)

The core code segment of K-NN detection algorithm is illustrated as followed. We adopt the python programming language to develop the source code. The subroutine of KNN is aiming to retrieve the label for the testing data for the top K nearest neighbour. While the decision subroutine is targeted to labelled that testing data in terms of the classification result. The most crucial step is the voting mechanism in the final stage, which the majority label of a class will predominate another class and be attached to the testing sample.

```
#########################################################################
#Derive label for test data by KNN algorithm, assign value 7 to K.

def KNN(testpoint, st, K=7):
    dis=metric(testpoint, st)
    DIS=heapq.nsmallest(K,dis)

    print 'Current top K-neareast neighbours are: \n', DIS

    label=[]
    def labellist(DIS):
        for i in range(K):
            j=DIS[i][1]
            label.append(st[j]['Label'])
        return label

    KNNlabel=labellist(DIS)
```

```
    print 'The label of K-nearest neighbours are : ', KNNlabel

    def decision(label):
        if(label.count(1) > label.count(0)):
            print 'Classification result : Label 1'
            return '1'
        else:
            print 'Classification result : Label 0'
            return '0'
    return decision(label)

#####################################################################
```

Test Results

The complete K-NN detection results of each value of K can be found in the appendix part. We just illustrate one sample of the detection result here and provide a brief explanation on those tables 13, which its data is derived from the experiments.

Here we taking the K=5 as example, the left column is the classification results generated by our KNN detection algorithm. The highlighting cells in yellow stand for the outliers, which should be detected and labelled as "0" correspondingly. The middle column is used to rectify the wrong attached labels, which are also highlighted by the magenta bar. It used to highlight the wrong classification elements in the testing set. For the complete test reports of our KNN detection experiments can be found in the appendix part of this paper.

Grubbs Test Report (n=50, p=0.05)		
50-outlier Set	**20-outlier Set**	**5-outlier Set**
-4.146156263	-0.545568371	-0.488924399
-4.130945409	-0.398256648	-0.470278733
-3.090334241	-0.354078839	-0.366802326
-2.078057979	-0.346423253	-0.276788458
-2.044163061	-0.310063628	-0.015096731
-0.366802326	-0.264096399	-0.01026995
-0.354078839	-0.236263867	-0.005653937
-0.344817988	-0.215134841	0.002724068
-0.337359013	-0.128255665	0.006574268
-0.328255665	-0.015096731	0.032929701
-0.314324743	-0.01026995	0.038088677
-0.244278549	-0.005653937	0.039874753
-0.238924479	0.006574268	0.040523837
-0.237096041	0.032929701	0.044163061
-0.236629796	0.038088677	0.045568371
-0.229880069	0.039874753	0.050820797
-0.22042762	0.044163061	0.052086917
-0.191582856	0.050820797	0.063040367
-0.190018852	0.052086917	0.077591673
-0.185123008	0.063040367	0.078057979
-0.17707927	0.077591673	0.085123008
-0.167704529	0.078057979	0.090334241
-0.160113788	0.085123008	0.104679587
-0.146423253	0.090334241	0.10707927
-0.015096731	0.104679587	**0.114324743**
-0.01026995	0.10707927	0.126603189

Figure 12: An example of Grubbs' outlier detection test report

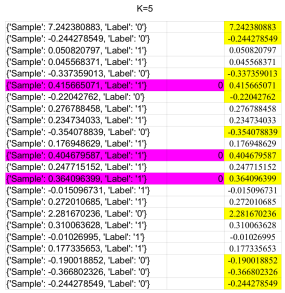

Figure 13: An example of KNN outlier detection test report (K=5)

5 Performance Study

In this chapter, the comprehensive examination will be conducted on the test results. The purpose is to addressing the research questions in our paper with rigorous quantitatively analysis. In the end of this chapter, a concise summary will be presented.

5.1 Discussions on Grubbs' detection

In the initial stage of our detection prototype, we have conducted several rounds of Grubbs' detection experiments. They mainly get involved in the variance in the value of N and the number of total outliers. Now, we are going to examine them individually with quantitatively means.

Before going forward to the detailed discussion on each table, we intend to do a quick explanation on the terms in the tables.

- **Detection Counts** :
  The number of total detection occurrence.
- **Correct detection counts** :
  The number of successful outlier detection occurrence.
- **Test Detection Accuracy** :
  The accuracy of comparing the number of correctly outlier detection occurrence with the total detection times.
- **Overall Detection Accuracy** :
  The accuracy of comparing the number of correctly detected outliers with the number of all outliers in the raw data set.
- **Test Detection Rate** :
  The total number of detection occurrence comparing with the total number of elements in the raw data set.

5.1.1 Variance in N

Case 1 : N=100

The Grubbs' detection report of $N = 100$ is shown in 6 . This table reflects the relevant index by the experiments with different numbers of outliers when keeping $N = 100, P = 0.05$ as static parameters.

We can find that the occurrence number of Grubbs' detection with this profile is almost in the same level. The minimum detection rate is 50%, while the maximum detection rate is 65%. The higher detection rate means the more expensive computational overhead, which is the crucial concern for big data background.

From another perspective, we can see that the overall detection accuracy is consider-

Performance Report of Grubbs' Detection (N=100, P=0.05, 100 samples)			
	50-Outlier	20-outlier	5-outlier
Detection Counts	65	50	63
Correct detection counts	27	10	2
Test Detection Accuracy	41.54%	20%	3%
Overall Detection Accuracy	54%	50%	40%
Test Detection Rate	65%	50%	63%

Table 6: Performance Report of Grubbs' Detection (N=100, P=0.05)

able acceptable for us. The maximum rate is 54%, which means almost more than half of the inherent outliers will be detected in the raw data set. The medium accuracy is 50%, which is still an acceptable result for the data preprocessing work. However, the minimum accuracy is only 40%, which is not so benign result.

If we taking the total detection work load and overall detection precision into consideration simultaneously, the 50-outlier testing set outperformed the other two test groups with 20-outlier and 5-outlier respectively. The only imperfection is the high ratio of detection occurrence, even the other two groups are also approximately at the same level.

Case 2 : N=50

The Grubbs' detection report of $N = 50$ is shown in 7 . This table reflects the relevant index by the experiments with different numbers of outliers when keeping $N = 100, P = 0.05$ as static parameters.

Performance Report of Grubbs' Detection (N=50, P=0.05, 100 samples)			
	50-Outlier	20-outlier	5-outlier
Detection Counts	53	1	66
Correct detection counts	29	0	2
Test Detection Accuracy	54.7%	0%	2%
Overall Detection Accuracy	58%	0%	40%
Test Detection Rate	53%	1%	66%

Table 7: Performance Report of Grubbs' Detection (N=50, P=0.05, 100 samples)

We can find that the occurrence number of Grubbs' detection with this profile is quite different for 20-outlier test set, comparing with other two groups. The minimum detection occurrence number is 1, while the maximum value is 66. This is a dramatic change among each other groups.

From another perspective, we can see that the overall detection accuracy is also acceptable for us. The maximum rate is 58%. It can be said that more than half of the outlying data will be detected in the raw data set. While the minimum accuracy is only 1%, which is still an acceptable result for the data preprocessing work. However, the minimum accuracy is only 1%. It is hardly to say that is an good result.

As the above case (N=100), the 50-outlier testing set also outperformed the other two

test groups, considering with both detection work load and overall detection precision in the same time. The only defect is also the relative high frequency of detection occurrence.

Case 3 : N=20

The Grubbs' detection report of $N = 50$ is shown in 8 . This table reflects the relevant index by the experiments with different numbers of outliers when keeping $N = 100, P = 0.05$ as static parameters.

Performance Report of Grubbs' Detection (N=20, P=0.05, 100 samples)			
	50-Outlier	20-outlier	5-outlier
Detection Counts	27	19	27
Correct detection counts	12	5	2
Test Detection Accuracy	44.4%	26.3%	7.4%
Overall Detection Accuracy	24%	25%	40%
Test Detection Rate	27%	19%	27%

Table 8: Performance Report of Grubbs' Detection (N=20, P=0.05, 100 samples)

We can find that the occurrence number of Grubbs' detection with this profile is different among different test sets. The minimum detection occurrence number is 19, while the maximum number of detection occurrence is 27. In summary , we can say that the work load of detection is almost in the same scale. But one interesting phenomenon is the total number of detection occurrence has declined significantly, compared to the above two sets of experiment with different value of K.

From another perspective, we can see that the overall detection accuracy varies from 40% to 20%. It can be said that at least around one fourth of the outlying data will be detected in the raw data set. This is more or less OK for initial stage of data preprocessing work.

Comparing with above cases ($N = 100\&N = 50$), the 5-outlier testing set outperformed the other two test groups, from the perspective of detection work load and overall detection precision simultaneously. It is a good solution for detecting lower ratio of outliers in raw data set, with the price of 27% detection coverage rate and 40% detection accuracy.

5.1.2 Cross check for N

In order to investigate the performance of each setting of parameter pairs as (N,P), we have launched the cross check sector as followed.

Case 1 : Outliers = 50

The Grubbs' detection reports reorganized by the total number of outliers as $Outlier = 50$ shown in 9 . This table reflects the relevant index by the experiments with different parameter pairs of (N, P) when the total number of outliers setting as static parameters.

Performance Report of Grubbs' Detection (50 outliers, 100 samples)			
	N=100, P=0.05	N=50, P=0.05	N=20, P=0.05
Detection Counts	65	53	27
Correct detection counts	27	29	12
Test Detection Accuracy	41.5%	54.7%	44.4%
Overall Detection Accuracy	54%	58%	24%
Test Detection Rate	65%	53%	27%

Table 9: Performance Report of Grubbs' Detection (50 outliers, 100 samples, P=0.05)

From the table 9, we can see that if we want to derive up to 50% of the overall detection accuracy, the price for that is the relative high work load spent on the detection overhead. The computation requirement is not less can 53%.

Case 2 : Outliers = 20

The Grubbs' detection reports reorganized by the total number of outliers as $\mathrm{Outlier} = 20$ shown in 10 . This table reflects the relevant index by the experiments with different parameter pairs of (N, P) when the total number of outliers setting as static parameters.

Performance Report of Grubbs' Detection (20 outliers, 100 samples)			
	N=100, P=0.05	N=50, P=0.05	N=20, P=0.05
Detection Counts	50	1	19
Correct detection counts	10	0	5
Test Detection Accuracy	20%	0	26.3%
Overall Detection Accuracy	50%	0	25%
Test Detection Rate	50%	0	19%

Table 10: Performance Report of Grubbs' Detection (20 outliers, 100 samples, P=0.05)

From the table 10, we can also find that if we want to derive up to 50% of the overall detection accuracy, the price for that is the relative high work load spent on the detection overhead. While the parameter pair of (N=50, P=0.05) shown a strange value as only 1 detection taken place and no outlier detected. That might be due to the coincidence of statistics properties residing in those divided data packages from the raw data set.

Case 3 : Outliers = 5

The Grubbs' detection reports reorganized by the total number of outliers as $\mathrm{Outlier} = 5$ shown in 11 . This table reflects the relevant index by the experiments with different parameter pairs of (N, P) when the total number of outliers setting as static parameters.

From the table 11, we can see that the overall detection accuracy is the right same as 40%. While the computational price is ranging from 27% to 63%. Relatively, It can say that the parameter pair of (N=20, P=0.05) outperformed than other two pairs. It is able to achieve the overall accuracy of 40% with the computational price of 27%, which is an ideal solution to processing lower percentage inherent (like this experiment, 5%) outliers data set.

Performance Report of Grubbs' Detection (5 outliers, 100 samples)			
	N=100, P=0.05	N=50, P=0.05	N=20, P=0.05
Detection Counts	63	66	27
Correct detection counts	2	2	2
Test Detection Accuracy	3.17%	3.03%	7.4%
Overall Detection Accuracy	40%	40%	40%
Test Detection Rate	63%	66%	27%

Table 11: Performance Report of Grubbs' Detection (5 outliers, 100 samples, P=0.05)

5.1.3 Quick review

According to the above quantitative analysis, we can say that the parameter pair of (N=100, P=0.05) outperformed than other two pairs in higher percentage inherent outlier data set. The main pitfall is the parallel higher work load for executing detection computation task. When the raw data set turns to be an extremely large scale, the computation price will probably be unaffordable for that scenario.

In the other hand, the parameter pair of (N=20, P=0.05) outperformed than others in the same lower percentage inherent (like this experiment, 5%) outliers data set scenario. It is able to achieve the satisfactory detection accuracy with a relative lower computation price.

5.2 Discussions on KNN detection

In terms of our detection prototype, KNN detection method is the second stage, which is immediately following the Grubbs' detection procedure. In this stage, the computation work is focusing on distance measurement and label voting in order to make a classification decision on a dedicated testing data. In order to investigate the performance of each different K, we have launched several rounds of experiments for verification. In addition, the training set consists of 42 outliers and 38 normal data elements. Now, we are going to examine them individually with quantitatively analysis.

Before starting to discuss the details in each table, we have to do a brief explanation on the terms in the following tables.

- **Origin Number** :
  The origin number of each type of data in the input before executing the KNN detection procedure.
- **Detection Results** :
  The number of detected elements by KNN classification application.
- **Detection Accuracy** :
  The accuracy of comparing the number of correctly detected elements with the total number of its corresponding class.

5.2.1 Variance in K

Case 1: K=3

The K-nearest neighbour detection report with K =3 is shown as followed 12. The table reflects the performance of this method with classifying 80 data elements in the testing set. We choose the top 3 closest neighbours' label as the base of voting the testing data a corresponding label or class.

Performance Report of KNN detection (K=3)		
	Outlying element	Normal element
Origin Number	42	38
Detection Result	37	43
Detection Accuracy	88.1%	113.2%*

Table 12: Performance Report of KNN detection (K=3)

From the table 12, we can see that 37 outliers have been successfully detected by KNN algorithm. It can reach up to 88.1% classification accuracy, which is a quite satisfactory output for us. However, there are still some outliers can not be detected from the testing data set. As the figure of 113.2%* shown, the labelled normal data set still has some outliers. The reason for this value resulted from the bias of some training data element. This bias impacted the classification procedure by biased distance measurement.

Case 2: K=5

The K-nearest neighbour detection report with K =5 is shown as followed 13. The table reflects the performance of this method with classifying 80 data elements in the testing set. We choose the top 3 closest neighbours' label as the base of voting the testing data a corresponding label or class.

Performance Report of KNN detection (K=5)		
	Outlying element	Normal element
Origin Number	42	38
Detection Result	36	44
Detection Accuracy	85.7%	115.8%*

Table 13: Performance Report of KNN detection (K=5)

From the table 13, we can see that 36 outliers have been successfully detected by KNN algorithm. It can reach up to 85.7% classification accuracy, which is a quite satisfactory output for us. Even through it is slightly imperfect when comparing with the case of K = 3. However, outlier still can not be entirely removed from the labelled normal set.

Case 3: K=7

The K-nearest neighbour detection report with K =7 is shown as followed 14. The table reflects the performance of this method with classifying 80 data elements in the testing set. We choose the top 7 closest neighbours' label as the base of voting the testing data a corresponding label or class.

Performance Report of KNN detection (K=7)		
	Outlying element	Normal element
Origin Number	42	38
Detection Result	36	44
Detection Accuracy	85.7%	115.8%*

Table 14: Performance Report of KNN detection (K=7)

From the table 14, we can see that 36 outliers have been successfully detected by KNN algorithm as well. The classification accuracy is the same as the previous case of $K = 5$, which is also a quite satisfactory output for us. The classification results keep stable as the case of $K = 5$.

5.2.2 Quick review

According to the case analysis with quantitative methods, we can say that the K-nearest neighbour performs quite well in the outlier detection work. Comparing the case of $K = 3$ and $K = 5\&7$, we can find that the classification result varied on the value of K. If the value of K is smaller, the closest training samples put more influence on the decision of classification. If the value of K fallen in some range, the performance of KNN algorithm will approximate to a stable standard.

5.3 Limitations & weaknesses

From the global view, we can see that this proposed solution is considerably suitable for preprocessing big data set for implementing large scale digital forensics evidence analysis. It can significant facilitate the forensics work for analysis evidence in big data set. However, this prototype of our solution has its own target application scenario.

5.3.1 Concerns of Grubbs'

For the initial detection stage of Grubbs' detection, we can see such facts related to various big data set. When the setting of parameter pair is $(N = 20, P = 0.05)$, it can achieve a satisfactory performance for the lower inherent outliers cases. For example, if the big data set is up to 1 Terabyte, 40% outliers can be correctly detection from the raw evidence package. Meanwhile, the computational overhead will be only 27% than other profiles, which chosen bigger N for processing the same size of data set. The other setting of parameter pairs for detecting outlier in big data set require relatively higher computation resource. That is a challenge and sometimes can not be meet. In other words, the selection of proper parameters for implementing an efficient detection task is a crucial concern for digital forensics investigators.

Furthermore, the absolute computational expense is still not so cheap, even with the optimized parameter setting for Grubbs' algorithm. Even if we can apply the divide and conquer method to process the big data set by computer clusters. It will still have to consume a lot of computational resource. Unfortunately, people have so fewer solutions in front of big data challenge currently.

5.3.2 Concerns of KNN

The second round of detection by KNN algorithm performed quite well for correctly detection. But the price of yielding higher accuracy than the Grubbs' algorithm is the higher computation requirement. In contrast to Grubbs' detection method, only the sorting operation is a time-consuming task. Actually, there are many advanced sorting algorithm that can mitigate this problem to some extent. While, for KNN algorithm, it involved many procedures as distance computing, sorting, voting and so on. It is a considerable huge overhead for the computer systems.

Another concern is the selection of K value. Different value of K will lead to different classification accuracy in the end. This is one of the pitfall for KNN algorithm. In order to achieve a satisfactory output, a carefully selection of K should be made in advance.

Then, the classification precision is closely related to the size of training set. If there was not enough training samples, the final classification result can not be guaranteed. Some kind of bias will be introduced into the output.

For our case, it is simply only 1-dimensional data set. However, for those higher dimensional circumstance, the degree of cardinal for each dimensional should also be put into consideration. It could result in bias when computing the distance in higher dimensional vector space.

6 Conclusion

In this master thesis, a challenging research question, which is closely related to the challenge of large scale digital forensics investigation on big data sets has been presented We have carried on a systematic theoretical study on the topic of digital forensics discipline accompanying with the big data challenges. Then we have proposed a novel data preprocessing solution for handling such a challenge. Afterwards, we have implemented a proof of concept experiment, in order to verify the performance of our work. All the data collected from these experiments have been carefully analysed in the performance study chapter.

The outcome of our work illustrates that our novel solution is capable dealing with the large scale digital forensics investigation on big data sets, which contain low percentage of inherent outliers. Our solution is designed in a double-round architecture. By the first round of Grubbs' detection, a considerable portion of outliers can be detected from the raw big data set by pure stochastic processing. Then the remaining of first preprocessing process will be processed by KNN detection algorithm. After completing those two rounds of detection work, the raw big data set becomes smaller but more meaningful data set for further digital forensics investigation than before.

Our research illustrated that by subtly selecting the appropriate parameters for each round of the detection process, a satisfactory output can be achieved in the large scale digital forensics investigations on big data set cases. Different value binding to parameters will lead to different amount of work load and varied quality in the final output. The final quality of our work can only guaranteed by the highly connected cooperation of both detection processes. In the worst cases, our solution can still work but require a higher computational expense. For some extremely large scale cases may be not easy to afford such a price.

7 Future Work

The data preprocessing is an extremely important step for constructing a solid foundation for digital evidence analysis and interpretation, especially encountering the large scale digital forensics investigation on big data set scenarios. So far there is not so many solutions for this novel challenge. In order to mitigate such a challenge, we have developed a brand new solution by combining Grubbs' detection with K-nearest neighbour machine learning for further detection. Even though our solution makes a strong progress in this task, there is still some further work that could be done on this solution in order to achieve better performance.

According to the topic of this paper, outlier detection in large scale digital forensics scenarios on big data by applying Grubbs' test and KNN machine learning methods. In term of this concern, the future work should be done in the following directions.

First of all, the input data for data preprocessing could be normalized by some kind of standards or formats, which is to govern the operations on data collection from the derived evidences and so on. Based on the uniform platform, it is easier to discover and eliminate the exceptions in the very beginning. Furthermore, Due to the stochastic approaches has been applied as the initial phase in our paper.

For example, more sophisticated methods might be developed to find the appropriate parameter pairs for different kinds of big data set.In terms of the principle of digital forensics, any valuable evidence should not be neglected with careless operations. Consequently, purpose of data preprocessing would better to employ a relatively conservative way.

Moreover, the optimization work is also keenly required for machine learning preprocessing in the second round. The main goal of data preprocessing is to eliminate the inherent outliers. But the KNN machine learning algorithm itself is an outlier-prone technique. It would be a quite interesting question for the academic communities. Last but not least, more subtle combinations of outlier detection methodologies can be developed to yield better preprocessing performance.

Bibliography

[1] Hodge, V. J. & Austin, J. 2004. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22, 85–126. doi:http://dx.doi.org/10.1023/B:AIRE.0000045502.10941.a9.

[2] Müller, H. & Freytag, J.-C. 2005. Problems, Methods, and Challenges in Comprehensive Data Cleansing.

[3] CHANDOLA, V. & KUMAR, V. Outlier Detection : A Survey.

[4] Agyemang, M., Barker, K., & Alhajj, R. 2006. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis*, 10, 521–538.

[5] Grubbs, F. E. 1969. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11, 1–21. doi:10.1080/00401706.1969.10490657.

[6] Barnett, V. & Lewis, T. 1994. Outliers in statistical data.

[7] Chikkagoudar, M. S. & Kunchur, S. H. 1983. Distributions of test statistics for multiple outliers in exponential samples. *Communications in Statistics-theory and Methods*, 12, 2127–2142. doi:10.1080/03610928308828596.

[8] Ueno, M. & Nagaoka, K. Learning Log Database and Data Mining system for e-Learning OnLine Statistical Outlier Detection of irregular learning processes.

[9] Zerbet, A. & Nikulin, M. 2003. A New Statistic for Detecting Outliers in Exponential Case. *Communications in Statistics-theory and Methods*, 32, 573–583. doi:10.1081/STA-120018552.

[10] Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., & Kanamori, T. 2011. Statistical Outlier Detection Using Direct Density Ratio Estimation. *Knowledge and Information Systems*, 26, 309–336. doi:10.1007/s10115-010-0283-2.

[11] Zhang, Y., Hamm, N. A. S., Meratnia, N., Stein, A., van de Voort, M., & Havinga, P. J. M. 2012. Statistics-based outlier detection for wireless sensor networks. *International Journal of Geographical Information Science*, ahead-of-p, 1–20. doi:10.1080/13658816.2012.654493.

[12] Lalitha, S. & Joshi, P. C. 1986. PERFORMANCE OF STUDENTIZED RANGE STATISTIC FOR TWO OUTLIERS IN A LINEAR MODEL. *Statistica Neerlandica*, 40, 157–168. doi:10.1111/j.1467-9574.1986.tb01512.x.

[13] Kalamangalam, G. P. 2008. A statistical outlier. *British Medical Journal*, 337. doi:10.1136/bmj.a2305.

[14] Martinez, R., Cebrian, M., Rodríguez, F. D. B., & Camacho, D. 2007. Contextual Information Retrieval based on Algorithmic Information Theory and Statistical Outlier Detection. *Computing Research Repository*, abs/0711.4.

[15] Lepsoy, S., Francini, G., Cordara, G., & de Gusmao, P. P. B. 2011. Statistical modelling of outliers for fast visual search. In *International Conference on Multimedia Computing and Systems/International Conference on Multimedia and Expo*, 1–6. doi:10.1109/ICME.2011.6012184.

[16] Chen, F., Lu, C.-T., & Boedihardjo, A. P. 2010. GLS-SOD: a generalized local statistical approach for spatial outlier detection. In *Knowledge Discovery and Data Mining*, 1069–1078. doi:10.1145/1835804.1835939.

[17] Black, M. J. & Rangarajan, A. 1994. The outlier process: unifying line processes and robust statistics. In *Computer Vision and Pattern Recognition*, 15–22. doi:10.1109/CVPR.1994.323805.

[18] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., & Steinberg, D. 2007. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14, 1–37. doi:10.1007/s10115-007-0114-2.

[19] Yu, C., Ooi, B. C., lee Tan, K., & Jagadish, H. V. 2001. Indexing the Distance: An Efficient Method to KNN Processing. In *Very Large Data Bases*, 421–430.

[20] ling Zhang, M. & hua Zhou, Z. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40, 2038–2048. doi:10.1016/j.patcog.2006.12.019.

[21] Tan, S. 2006. An effective refinement strategy for KNN text classifier. *Expert Systems With Applications*, 30, 290–298. doi:10.1016/j.eswa.2005.07.019.

[22] Guo, G., Wang, H., Bell, D. A., Bi, Y., & Greer, K. 2003. *KNN Model-Based Approach in Classification*.

[23] Xia, C., Lu, H., Ooi, B. C., & Hu, J. 2004. Gorder: An Efficient Method for KNN Join Processing. In *Very Large Data Bases*, 756–767.

[24] Shen, H.-B., Yang, J., & Chou, K.-C. 2006. Fuzzy KNN for predicting membrane protein types from pseudo-amino acid composition. *Journal of Theoretical Biology*, 240, 9–13. doi:10.1016/j.jtbi.2005.08.016.

[25] Jiang, Y. & hua Zhou, Z. 2004. *Editing Training Data for kNN Classifiers with Neural Network Ensemble*. doi:10.1007/978-3-540-28647-9_60.

[26] Soucy, P. & Mineau, G. W. 2001. A Simple KNN Algorithm for Text Categorization. In *IEEE International Conference on Data Mining*, 647–648. doi:10.1109/ICDM.2001.989592.

[27] Gedik, B., Singh, A., & Liu, L. 2004. Energy efficient exact kNN search in wireless broadcast environments. In *Workshop on Advances in Geographic Information Systems*, 137–146. doi:10.1145/1032222.1032244.

[28] Ramaswamy, S., Rastogi, R., & Shim, K. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. *Sigmod Record*, 29, 427–438. doi:10.1145/342009.335437.

[29] Bishop, C. M. 1994. Novelty Detection and Neural Network Validation. *Iee Proceedings-vision Image and Signal Processing*, 141. doi:10.1049/ip-vis:19941330.

[30] Roberts, S. J. & Tarassenko, L. 1994. A Probabilistic Resource Allocating Network for Novelty Detection. *Neural Computation*, 6, 270–284. doi:10.1162/neco.1994.6.2.270.

[31] j. Tax, D. M. & Ypma, A. 1999. Support Vector Data Description Applied to Machine Vibration Analysis.

[32] Decoste, D. & Levine, M. B. 2000. Automated Event Detection in Space Instruments: A Case Study Using IPEX-2 Data and Support Vector Machines.

[33] Lauer, M. 2001. *A Mixture Approach to Novelty Detection Using Training Data with Outliers*. doi:10.1007/3-540-44795-4_26.

[34] SOHN, H., WORDEN, K., & FARRAR, C. 2001. NOVELTY DETECTION USING AUTO-ASSOCIATIVE NEURAL NETWORK.

[35] Barreto, G. D. A. & Aguayo, L. 2009. *Time Series Clustering for Anomaly Detection Using Competitive Neural Networks*. doi:10.1007/978-3-642-02397-2_4.

[36] Smyth, P. 1994. Markov monitoring with unknown states. *IEEE Journal on Selected Areas in Communications*, 12, 1600–1612. doi:10.1109/49.339929.

[37] Hollmén, J. & Tresp, V. 1998. Call-Based Fraud Detection in Mobile Communication Networks Using a Hierarchical Regime-Switching Model. In *Neural Information Processing Systems*, 889–895.

[38] Gruber, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-computer Studies / International Journal of Man-machine Studies*, 43, 907–928. doi:10.1006/ijhc.1995.1081.

[39] Brinson, A., Robinson, A., & Rogers, M. 2006. A cyber forensics ontology: Creating a new approach to studying cyber forensics. *Digital Investigation*, 3, 37–43. doi:10.1016/j.diin.2006.06.008.

[40] Klein, M., Fensel, D., & Harmelen, F. V. 2001. The relation between ontologies and XML schemas. *Electronic Transactions on Artificial Intelligence*.

[41] Mirkovic, J. & Reiher, P. L. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *Computer Communication Review*, 34, 39–53. doi:10.1145/997150.997156.

[42] PATRICK CARMICHAEL, E. A. 1997. Federal rules of evidence. http://www.justice.gov/osg/briefs/1997/3mer/1ami/97-1709.mer.ami.pdf. Department of Justice of the United States of America.

[43] Department of Justice, U. 2011. Rule 702. testimony by expert witnesses. http://www.law.cornell.edu/rules/fre/rule_702. Legal Information Institure, Cornell Univserity.

[44] Kruse, W. G. & Heiser, J. G. 2002. Computer forensics: incident response essentials.

[45] Digital forensic research workshop. http://www.dfrws.org. (Digital Forensic Research WorkShop).

[46] Palmer, G. & Corporation, M. A Road Map for Digital Forensic Research. Technical report, November 2001. URL: `http://isis.poly.edu/kulesh/forensics/docs/DFRWS_RM_Final.pdf`.

[47] Farmer, D. & Venema, W. 2004. *Forensic Discovery*. Addison Wesley Professional.

[48] III, G. G. R. & Roussev, V. 2006. Next-generation digital forensics. *Communications of The ACM*, 49, 76–80. `doi:10.1145/1113034.1113074`.

[49] Kruse, W. G. & Heiser, J. G. 2002. Computer forensics: incident response essentials.

[50] Mandia, K. & Prosise, C. 2001. Incident Response: Investigating Computer Crime.

[51] Reith, M., Carr, C., & Gunsch, G. H. 2002. An Examination of Digital Forensic Models. *International Journal of Digital Evidence*, 1.

[52] NIJ. 2001. *Electronic Crime Scene Investigation:A Guide for First Responders*. National Institute of Justice, Office of Justice Programs, U.S. Department of Justice.

[53] Berkeley, U. 1999. Seti@home. http://setiathome.berkeley.edu/. BOINC.

[54] e.V., R. 2009. Rna world. http://www.rnaworld.de/rnaworld/. BOINC.

[55] Cisco. 2013. Cisco visual networking index:global mobile data traffic forecast update,2012–2017. http://goo.gl/cGvZ9. Cisco Visual Networking Index.

[56] ITU-T. 2003. Security in telecommunications and information technology – an overview the issues and the deployment of existing itu-t recommendations for secure telecommunications. http://www.itu.int/itudoc/itu-t/85097.pdf. ITU-T.

[57] Issenberg, S. 2012. How president obama's campaign used big data to rally individual voters. http://www.technologyreview.com/featuredstory/509026/how-obamas-team-used-big-data-to-rally-voters. MIT Technology Review.

[58] Arak, V. 2007. On the worm that affects skype for windows users. http://heartbeat.skype.com/2007/09/the_worm_that_affects_skype_fo.html. Skype.

[59] Famili, F., min Shen, W., Weber, R., & Simoudis, E. 1997. Data Preprocessing and Intelligent Data Analysis. *Intelligent Data Analysis*, 1, 3–23. `doi:10.1016/S1088-467X(98)00007-9`.

[60] EL. Bontrager, J. Perry, R. B. J. G. L. B. G. B. & Kim, J. 1990. GAIT-ER-AID: An Expert System for Analysis of Gait with Automatic Intelligent Pre-Processing of Data.

[61] Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*.

[62] Quinlan, J. R. 1987. Simplifying Decision Trees. *International Journal of Human-computer Studies / International Journal of Man-machine Studies*, 27, 221–234. doi:10.1016/S0020-7373(87)80053-6.

[63] CML, University of California, I. Seti@home. http://archive.ics.uci.edu/ml/datasets.html. Center for Machine Learning and Intelligent Systems, University of California, Irvine.

[64] John, G. H. 1995. Robust Decision Trees: Removing Outliers from Databases. In *Knowledge Discovery and Data Mining*, 174–179.

[65] Aggarwal, C. C. & Yu, P. S. 2001. Outlier detection for high dimensional data. *Sigmod Record*, 37–46. doi:10.1145/376284.375668.

[66] Blake, C. L Merz, C. J. 1998. Uci repository of machine learning databases. http://archive.ics.uci.edu/ml/datasets.html. University of California, Irvine, Department of Information and Computer Sciences.

[67] Rousseeuw, P. J. & Leroy, A. M. 1987. Robust regression and outlier detection.

[68] Torr, P. H. S. & Murray, D. W. 1993. Outlier detection and motion segmentation. In *Storage and Retrieval for Image and Video Databases*.

[69] Japkowicz, N., Myers, C., & Gluck, M. A. 1995. A Novelty Detection Approach to Classification. In *International Joint Conference on Artificial Intelligence*, 518–523.

[70] Fawcett, T. & Provost, F. J. 1999. Activity monitoring: noticing interesting changes in behavior. In *Knowledge Discovery and Data Mining*, 53–62. doi:10.1145/312129.312195.

[71] Silberschatz, A., Galvin, P., & Gagne, G. 2003. Operating System Concepts.

[72] Hatch, B. April 24, 2003. Linux file permission confusion pt 2. http://www.hackinglinuxexposed.com/articles/20030424.html.

[73] Tolman, R. C. 1979. The principles of statistical mechanics.

[74] Dean, R. B. & Dixon, W. J. 1951. Simplified Statistics for Small Numbers of Observations. *Analytical Chemistry*, 23, 636–638. doi:10.1021/ac60052a025.

[75] Lloyd, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28, 129–136. doi:10.1109/TIT.1982.1056489.

[76] Viola, P. A. & Jones, M. J. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Computer Vision and Pattern Recognition*, volume 1, 511–518. doi:10.1109/CVPR.2001.990517.

[77] Agrawal, R. & Srikant, R. 1994. Fast Algorithms for Mining Association Rules. In *Very Large Data Bases*.

[78] Dempster, A. P., Laird, N. M., & Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm.

[79] Brin, S. & Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and Isdn Systems*, 30, 107–117. doi:10.1016/S0169-7552(98)00110-X.

[80] Freund, Y. & Schapire, R. E. 1995. *A decision-theoretic generalization of on-line learning and an application to boosting*. doi:10.1007/3-540-59119-2_166.

[81] Domingos, P. & Pazzani, M. J. 1997. On the Optimality of the Simple Bayesian Classifier under Zero-OneLoss. *Machine Learning*, 29, 103–130.

[82] Breiman, L., Freidman, J. H., Olshen, R. A., & Stone, C. J. 1984. CART: Classification and Regression Trees.

[83] Fix, E. & Hodges, J. Discriminatory analysis–nonparametric discrimination: Consistency properties.

[84] ning Tan, P., Steinbach, M., & Kumar, V. 2005. *Introduction to Data Mining*.

[85] Poggio, T., Rifkin, R., Mukherjee, S., & Niyogi, P. 2004. General conditions for predictivity in learning theory. *Nature*, 428, 419–422. doi:10.1038/nature02341.

[86] Hart, P. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13, 21–27. doi:10.1109/TIT.1967.1053964.

[87] Hall, D. J. & Ball, G. B. 1965. ISODATA : A Novel Method of Data Analysis and Pattern Classification.

[88] Arthur, D. & Vassilvitskii, S. 2007. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035. doi:10.1145/1283383.1283494.

[89] Ostrovsky, R., Rabani, Y., Schulman, L. J., & Swamy, C. 2006. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. In *IEEE Symposium on Foundations of Computer Science*, 165–176. doi:10.1109/FOCS.2006.75.

[90] Drineas, P., Frieze, A. M., Kannan, R., Vempala, S., & Vinay, V. 2004. Clustering Large Graphs via the Singular Value Decomposition. *Machine Learning*, 56, 9–33. doi:10.1023/B:MACH.0000033113.59016.96.

[91] Mitchell, T. M. 1997. *Machine Learning*.

[92] Macleod, J., Luk, A., & Titterington, D. 1987. A Re-Examination of the Distance-Weighted k-Nearest Neighbor Classification Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 17, 689–696. doi:10.1109/TSMC.1987.289362.

[93] Zavrel, J. 1997. An Empirical Re-Examination of Weighted Voting for k-NN.

[94] Zeng, Y., Yang, Y., & Zhao, L. 2009. Pseudo nearest neighbor rule for pattern classification. *Expert Systems With Applications*, 36, 3587–3595. doi:10.1016/j.eswa.2008.02.003.

[95] Bermejo, S. & Cabestany, J. 2001. *Large Margin Nearest Neighbor Classifiers*. doi:10.1007/3-540-45720-8_80.

[96] Zuo, W., Zhang, D., & Wang, K. 2008. On kernel difference-weighted k-nearest neighbor classification. *Pattern Analysis and Applications*, 11, 247–257. doi:10.1007/s10044-007-0100-z.

[97] Hechenbichler, K. & Schliep, K. 2004. Weighted k-Nearest-Neighbor Techniques and Ordinal Classification.

[98] Dudani, S. A. 1976. The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6, 325–327. doi:10.1109/TSMC.1976.5408784.

[99] Chernoff, K. & Nielsen, M. 2010. Weighting of the k-Nearest-Neighbors. In *International Conference on Pattern Recognition*, 666–669. doi:10.1109/ICPR.2010.168.

[100] Mercer, J. 1909. Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations. *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, 83, 69–70. doi:10.1098/rspa.1909.0075.

[101] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., & Gruber, R. E. 2006. Bigtable: a distributed storage system for structured data. In *Operating Systems Design and Implementation*, 205–218.

[102] Dean, J. & Ghemawat, S. 2004. MapReduce: Simplied Data Processing on Large Clusters. In *Operating Systems Design and Implementation*, 137–150.

[103] Ghemawat, S., Gobioff, H., & Leung, S.-T. 2003. The Google file system. In *ACM Symposium on Operating Systems Principles*, 29–43. doi:10.1145/945445.945450.

[104] Rice, J. 1995. Mathematical statistics and data analysis.

A Appendix

A.1 Grubbs' Critical Value Table

Grubbs' critical value table:

N	0.1	0.075	0.05	0.025	0.01	N	0.1	0.075	0.05	0.025	0.01
3	1.15	1.15	1.15	1.15	1.15	53	0	0	2.981	3.151	999
4	1.42	1.44	1.46	1.48	1.49	54	0	0	2.988	3.158	999
5	1.6	1.64	1.67	1.71	1.75	55	0	0	2.995	3.165	999
6	1.73	1.77	1.82	1.89	1.94	56	0	0	3.002	3.172	999
7	1.83	1.88	1.94	2.02	2.1	57	0	0	3.009	3.179	999
8	1.91	1.96	2.03	2.13	2.22	58	0	0	3.016	3.186	999
9	1.98	2.04	2.11	2.21	2.32	59	0	0	3.023	3.193	999
10	2.03	2.1	2.18	2.29	2.41	60	0	0	3.03	3.2	999
11	2.09	2.14	2.23	2.36	2.48	61	0	0	3.036	3.206	999
12	2.13	2.2	2.29	2.41	2.55	62	0	0	3.042	3.212	999
13	2.17	2.24	2.33	2.46	2.61	63	0	0	3.048	3.218	999
14	2.21	2.28	2.37	2.51	2.66	64	0	0	3.054	3.224	999
15	2.25	2.32	2.41	2.55	2.71	65	0	0	3.06	3.23	999
16	2.28	2.35	2.44	2.59	2.75	66	0	0	3.066	3.236	999
17	2.31	2.38	2.47	2.62	2.79	67	0	0	3.072	3.242	999
18	2.34	2.41	2.5	2.65	2.82	68	0	0	3.078	3.248	999
19	2.36	2.44	2.53	2.68	2.85	69	0	0	3.084	3.254	999
20	2.38	2.46	2.56	2.71	2.88	70	0	0	3.09	3.26	999
21	0	0	2.58	2.73	2.91	71	0	0	3.095	3.265	999
22	0	0	2.6	2.76	2.94	72	0	0	3.1	3.27	999
23	0	0	2.62	2.78	2.96	73	0	0	3.105	3.275	999
24	0	0	2.64	2.8	2.99	74	0	0	3.11	3.28	999
25	0	0	2.66	2.82	3.01	75	0	0	3.115	3.285	999
26	0	0	2.68	2.84	999	76	0	0	3.12	3.29	999
27	0	0	2.7	2.86	999	77	0	0	3.125	3.295	999
28	0	0	2.72	2.88	999	78	0	0	3.13	3.3	999
29	0	0	2.73	2.9	999	79	0	0	3.135	3.305	999
30	0	0	2.75	2.91	999	80	0	0	3.14	3.31	999
31	0	0	2.76	2.93	999	81	0	0	3.144	3.314	999
32	0	0	2.78	2.95	999	82	0	0	3.148	3.318	999
33	0	0	2.79	2.96	999	83	0	0	3.152	3.322	999
34	0	0	2.81	2.97	999	84	0	0	3.156	3.326	999
35	0	0	2.82	2.98	999	85	0	0	3.16	3.33	999
36	0	0	2.83	2.992	999	86	0	0	3.164	3.334	999
37	0	0	2.84	3.004	999	87	0	0	3.168	3.338	999
38	0	0	2.85	3.016	999	88	0	0	3.172	3.342	999
39	0	0	2.86	3.028	999	89	0	0	3.176	3.346	999
40	0	0	2.87	3.04	999	90	0	0	3.18	3.35	999
41	0	0	2.88	3.05	999	91	0	0	3.183	3.353	999
42	0	0	2.89	3.06	999	92	0	0	3.186	3.356	999
43	0	0	2.9	3.07	999	93	0	0	3.189	3.359	999
44	0	0	2.91	3.08	999	94	0	0	3.192	3.362	999
45	0	0	2.92	3.09	999	95	0	0	3.195	3.365	999
46	0	0	2.928	3.098	999	96	0	0	3.198	3.368	999
47	0	0	2.936	3.106	999	97	0	0	3.201	3.371	999
48	0	0	2.944	3.114	999	98	0	0	3.204	3.374	999
49	0	0	2.952	3.122	999	99	0	0	3.207	3.377	999
51	0	0	2.967	3.137	999	100	0	0	3.21	3.38	999
52	0	0	2.974	3.144	999						

Figure 14: Grubbs' critical value table

A.2 Data set illustration for Grubbs' detection

Data set illustration

Correct Data	50 outliers	20 outliers	5 outliers
0.246636021	0.246636021	0.246636021	0.246636021
0.160113788	-0.160113788	0.160113788	0.160113788
0.210831019	0.210831019	0.410831019	0.210831019
0.090334241	-3.090334241	0.090334241	0.090334241
0.30912883	0.30912883	0.30912883	0.30912883
0.19187195	5.19187195	0.19187195	0.19187195
0.085123008	-0.185123008	0.085123008	0.085123008
0.236757547	0.736757547	0.236757547	10.23675755
0.032929701	0.032929701	0.032929701	0.032929701
0.257869706	0.457869706	0.657869706	0.257869706
0.236629796	-0.236629796	0.236629796	0.236629796
0.209487151	0.209487151	0.209487151	0.209487151
0.063040367	0.063040367	0.063040367	0.063040367
0.038088677	1.038088677	0.038088677	0.038088677
0.217809562	0.217809562	0.217809562	0.217809562
0.329951166	0.829951166	0.329951166	0.329951166
0.360353945	0.360353945	0.360353945	0.360353945
0.344817988	-0.344817988	0.344817988	0.344817988
0.32130244	1.32130244	0.32130244	0.32130244
0.052086917	0.052086917	0.052086917	0.052086917
0.272752887	4.272752887	0.272752887	0.272752887
0.181445167	0.181445167	0.181445167	0.181445167
0.190003869	2.190003869	0.190003869	0.190003869
0.199964324	0.199964324	0.999964324	0.199964324
0.288924399	10.2889244	0.288924399	-0.488924399

Page 1

Figure 15: Grubbs' detction data set demonstration, part 1

Data set illustration

0.160113319	0.160113319	0.160113319	0.160113319
0.144168255	5.144168255	0.144168255	0.144168255
0.358934334	0.358934334	0.358934334	0.358934334
0.318850194	0.818850194	0.318850194	0.318850194
0.002724068	0.002724068	0.802724068	0.002724068
0.167704529	-0.167704529	0.167704529	0.167704529
0.226002711	3.226002711	0.226002711	0.226002711
-0.005653937	-0.005653937	-0.005653937	-0.005653937
0.039874753	0.903987475	0.039874753	0.039874753
0.10707927	-0.17707927	0.10707927	0.10707927
0.196429578	0.296429578	0.196429578	0.196429578
0.398256648	4.398256648	-0.398256648	0.398256648
0.194966165	0.194966165	0.194966165	0.194966165
0.236263867	0.536263867	-0.236263867	0.236263867
0.210329259	0.210329259	0.210329259	0.210329259
0.146423253	-0.146423253	-0.346423253	0.146423253
0.238924479	-0.238924479	0.238924479	0.238924479
0.278938277	0.278938277	0.278938277	0.278938277
0.181658642	0.381658642	0.181658642	0.181658642
0.129880069	-0.229880069	0.129880069	0.129880069
0.230287079	0.230287079	0.230287079	0.230287079
0.188998927	0.888998927	0.188998927	0.188998927
0.191582856	-0.191582856	0.191582856	0.191582856
0.252918492	0.252918492	0.252918492	0.252918492
0.114324743	-0.314324743	0.714324743	0.114324743
0.126603189	5.126603189	0.126603189	0.126603189

Page 2

Figure 16: Grubbs' detction data set demonstration, part 2

Data set illustration

0.310278733	0.310278733	0.310278733	-0.470278733
0.130945409	-4.130945409	0.530945409	0.130945409
0.134966154	0.134966154	0.134966154	0.134966154
0.201529341	6.201529341	0.201529341	0.201529341
0.190060047	0.190060047	0.490060047	0.190060047
0.152379851	9.152379851	0.152379851	0.152379851
0.224237814	0.224237814	0.224237814	0.224237814
0.164800359	0.164800359	0.164800359	0.164800359
0.006574268	0.006574268	0.006574268	0.006574268
0.343412229	8.343412229	0.343412229	0.343412229
0.215134841	0.215134841	-0.215134841	0.215134841
0.193742337	0.193742337	0.193742337	0.193742337
0.044163061	-2.044163061	0.044163061	0.044163061
0.286161206	0.286161206	0.286161206	0.286161206
0.078057979	-2.078057979	0.078057979	0.078057979
0.258840077	0.258840077	0.258840077	0.258840077
0.119304545	0.119304545	0.119304545	0.519304545
0.237096041	-0.237096041	0.237096041	0.237096041
0.077591673	0.077591673	0.077591673	0.077591673
0.228255665	-0.328255665	-0.128255665	0.228255665
0.223983102	7.223983102	0.623983102	0.223983102
0.165257895	0.165257895	0.165257895	0.165257895
0.040523837	0.040523837	0.740523837	0.040523837
0.356073613	0.356073613	0.356073613	0.356073613
0.146156263	-4.146156263	0.546156263	0.146156263
0.19457084	0.19457084	0.19457084	0.19457084

Page 3

Figure 17: Grubbs' detction data set demonstration, part 3

Data set illustration

0.242380883	7.242380883	0.242380883	0.242380883
0.201953463	979.2019535	0.201953463	0.201953463
0.050820797	0.050820797	0.050820797	0.050820797
0.045568371	0.045568371	-0.545568371	0.045568371
0.137359013	-0.337359013	0.137359013	0.137359013
0.156650712	0.415665071	0.156650712	0.156650712
0.22042762	-0.22042762	0.22042762	0.22042762
0.276788458	0.276788458	0.676788458	-0.276788458
0.234734033	0.234734033	0.234734033	0.234734033
0.354078839	-0.354078839	-0.354078839	0.354078839
0.176948629	0.176948629	0.176948629	0.176948629
0.104679587	0.404679587	0.104679587	0.104679587
0.247715152	0.247715152	0.247715152	0.247715152
0.264096399	0.364096399	-0.264096399	0.264096399
-0.015096731	-0.015096731	-0.015096731	-0.015096731
0.272010685	0.272010685	0.272010685	0.272010685
0.281670236	2.281670236	0.281670236	0.281670236
0.310063628	0.310063628	-0.310063628	0.310063628
-0.01026995	-0.01026995	-0.01026995	-0.01026995
0.177335653	0.177335653	0.177335653	0.177335653
0.290018852	-0.190018852	0.290018852	0.290018852
-0.366802326	-0.366802326	0.166802326	-0.366802326
0.244278549	-0.244278549	0.244278549	0.244278549

Page 4

Figure 18: Grubbs' detction data set demonstration, part 4

A.3 Grubbs' detection results (N=100, P=0.05)

G100 detection results

Grubbs Test Report (n=100, p=0.05)		
50-outlier Set	20-outlier Set	5-outlier Set
-4.146156263	-0.545568371	-0.488924399
-4.130945409	-0.398256648	-0.470278733
-3.090334241	-0.354078839	-0.366802326
-2.078057979	-0.346423253	-0.276788458
-2.044163061	-0.310063628	-0.015096731
-0.366802326	-0.264096399	-0.01026995
-0.354078839	-0.236263867	-0.005653937
-0.344817988	-0.215134841	0.002724068
-0.337359013	-0.128255665	0.006574268
-0.328255665	-0.015096731	0.032929701
-0.314324743	-0.01026995	0.038088677
-0.244278549	-0.005653937	0.039874753
-0.238924479	0.006574268	0.040523837
-0.237096041	0.032929701	0.044163061
-0.236629796	0.038088677	0.045568371
-0.229880069	0.039874753	0.050820797
-0.22042762	0.044163061	0.052086917
-0.191582856	0.050820797	0.063040367
-0.190018852	0.052086917	0.077591673
-0.185123008	0.063040367	0.078057979
-0.17707927	0.077591673	0.085123008
-0.167704529	0.078057979	0.090334241
-0.160113788	0.085123008	0.104679587
-0.146423253	0.090334241	0.10707927

Page 5

Figure 19: Grubbs' detection results table (N=100, P=0.05), part 1

G100 detection results

-0.015096731	0.104679587	0.114324743
-0.01026995	0.10707927	0.126603189
-0.005653937	0.119304545	0.129880069
0.002724068	0.126603189	0.130945409
0.006574268	0.129880069	0.134966154
0.032929701	0.134966154	0.137359013
0.040523837	0.137359013	0.144168255
0.045568371	0.144168255	0.146156263
0.050820797	0.152379851	0.146423253
0.052086917	0.156650712	0.152379851
0.063040367	0.160113319	0.156650712
0.077591673	0.160113788	0.160113319
0.119304545	0.164800359	0.160113788
0.134966154	0.165257895	**0.164800359**
0.160113319	0.166802326	**0.165257895**
0.164800359	0.167704529	**0.167704529**
0.165257895	0.176948629	**0.176948629**
0.176948629	0.177335653	**0.177335653**
0.177335653	0.181445167	**0.181445167**
0.181445167	0.181658642	**0.181658642**
0.190060047	0.188998927	**0.188998927**
0.193742337	0.190003869	**0.190003869**
0.19457084	0.191582856	**0.190060047**
0.194966165	0.19187195	**0.191582856**
0.199964324	0.193742337	**0.19187195**

Page 6

Figure 20: Grubbs' detection results table (N=100, P=0.05), part 2

G100 detection results

0.209487151	0.19457084	**0.193742337**
0.210329259	**0.194966165**	**0.19457084**
0.210831019	**0.196429578**	**0.194966165**
0.215134841	**0.201529341**	**0.196429578**
0.217809562	**0.201953463**	**0.199964324**
0.224237814	**0.209487151**	**0.201529341**
0.230287079	**0.210329259**	**0.201953463**
0.234734033	**0.217809562**	**0.209487151**
0.246636021	**0.22042762**	**0.210329259**
0.247715152	**0.224237814**	**0.210831019**
0.252918492	**0.226002711**	**0.215134841**
0.258840077	**0.230287079**	**0.217809562**
0.272010685	**0.234734033**	**0.22042762**
0.276788458	**0.236629796**	**0.223983102**
0.278938277	**0.236757547**	**0.224237814**
0.286161206	**0.237096041**	**0.226002711**
0.296429578	**0.238924479**	**0.228255665**
0.30912883	**0.242380883**	**0.230287079**
0.310063628	**0.244278549**	**0.234734033**
0.310278733	**0.246636021**	**0.236263867**
0.356073613	**0.247715152**	**0.236629796**
0.358934334	**0.252918492**	**0.237096041**
0.360353945	**0.258840077**	**0.238924479**
0.364096399	**0.272010685**	**0.242380883**
0.381658642	**0.272752887**	**0.244278549**

Page 7

Figure 21: Grubbs' detection results table (N=100, P=0.05), part 3

G100 detection results

0.404679587	0.278938277	0.246636021
0.415665071	0.281670236	0.247715152
0.457869706	0.286161206	0.252918492
0.536263867	0.288924399	0.257869706
0.736757547	0.290018852	0.258840077
0.818850194	0.30912883	0.264096399
0.829951166	0.310278733	0.272010685
0.888998927	0.318850194	0.272752887
0.903987475	0.32130244	0.278938277
1.038088677	0.329951166	0.281670236
1.32130244	0.343412229	0.286161206
2.190003869	0.344817988	0.290018852
2.281670236	0.356073613	0.30912883
3.226002711	0.358934334	0.310063628
4.272752887	0.360353945	0.318850194
4.398256648	0.410831019	0.32130244
5.126603189	0.490060047	0.329951166
5.144168255	0.530945409	0.343412229
5.19187195	0.546156263	0.344817988
6.201529341	0.623983102	0.354078839
7.223983102	0.657869706	0.356073613
7.242380883	0.676788458	0.358934334
8.343412229	0.714324743	0.360353945
9.152379851	0.740523837	0.398256648
10.2889244	0.802724068	0.519304545

Page 8

Figure 22: Grubbs' detection results table (N=100, P=0.05), part 4

Page 9

Figure 23: Grubs detection results table (N=100, P=0.05), part 5

A.4 Grubbs' detection results (N=50, P=0.05)

G50 detection results

Grubbs Test Report (n=50, p=0.05)		
50-outlier Set	20-outlier Set	5-outlier Set
-4.146156263	-0.545568371	-0.488924399
-4.130945409	-0.398256648	-0.470278733
-3.090334241	-0.354078839	-0.366802326
-2.078057979	-0.346423253	-0.276788458
-2.044163061	-0.310063628	-0.015096731
-0.366802326	-0.264096399	-0.01026995
-0.354078839	-0.236263867	-0.005653937
-0.344817988	-0.215134841	0.002724068
-0.337359013	-0.128255665	0.006574268
-0.328255665	-0.015096731	0.032929701
-0.314324743	-0.01026995	0.038088677
-0.244278549	-0.005653937	0.039874753
-0.238924479	0.006574268	0.040523837
-0.237096041	0.032929701	0.044163061
-0.236629796	0.038088677	0.045568371
-0.229880069	0.039874753	0.050820797
-0.22042762	0.044163061	0.052086917
-0.191582856	0.050820797	0.063040367
-0.190018852	0.052086917	0.077591673
-0.185123008	0.063040367	0.078057979
-0.17707927	0.077591673	0.085123008
-0.167704529	0.078057979	0.090334241
-0.160113788	0.085123008	0.104679587
-0.146423253	0.090334241	0.10707927

Page 10

Figure 24: Grubbs' detection results table (N=50, P=0.05), part 1

G50 detection results

-0.015096731	0.104679587	**0.114324743**
-0.01026995	0.10707927	0.126603189
-0.005653937	0.119304545	**0.129880069**
0.002724068	0.126603189	0.130945409
0.006574268	0.129880069	0.134966154
0.032929701	0.134966154	0.137359013
0.040523837	0.137359013	**0.144168255**
0.045568371	0.144168255	0.146156263
0.050820797	0.152379851	**0.146423253**
0.052086917	0.156650712	0.152379851
0.063040367	0.160113319	0.156650712
0.077591673	0.160113788	**0.160113319**
0.119304545	0.164800359	**0.160113788**
0.134966154	0.165257895	0.164800359
0.160113319	0.166802326	0.165257895
0.164800359	0.167704529	**0.167704529**
0.165257895	0.176948629	0.176948629
0.176948629	0.177335653	**0.177335653**
0.177335653	0.181445167	**0.181445167**
0.181445167	0.181658642	**0.181658642**
0.190060047	0.188998927	**0.188998927**
0.193742337	0.190003869	**0.190003869**
0.19457084	0.191582856	**0.190060047**
0.194966165	0.19187195	**0.191582856**
0.199964324	0.193742337	**0.19187195**

Page 11

Figure 25: Grubbs' detection results table (N=50, P=0.05), part 2

G50 detection results

0.209487151	0.19457084	**0.193742337**
0.210329259	0.194966165	**0.19457084**
0.210831019	0.196429578	**0.194966165**
0.215134841	0.201529341	**0.196429578**
0.217809562	0.201953463	**0.199964324**
0.224237814	0.209487151	**0.201529341**
0.230287079	0.210329259	**0.201953463**
0.234734033	0.217809562	**0.209487151**
0.246636021	0.22042762	**0.210329259**
0.247715152	0.224237814	**0.210831019**
0.252918492	0.226002711	**0.215134841**
0.258840077	0.230287079	**0.217809562**
0.272010685	0.234734033	**0.22042762**
0.276788458	0.236629796	**0.223983102**
0.278938277	0.236757547	**0.224237814**
0.286161206	0.237096041	**0.226002711**
0.296429578	0.238924479	**0.228255665**
0.30912883	0.242380883	**0.230287079**
0.310063628	0.244278549	**0.234734033**
0.310278733	0.246636021	**0.236263867**
0.356073613	0.247715152	**0.236629796**
0.358934334	0.252918492	**0.237096041**
0.360353945	0.258840077	**0.238924479**
0.364096399	0.272010685	**0.242380883**
0.381658642	0.272752887	**0.244278549**

Page 12

Figure 26: Grubbs' detection results table (N=50, P=0.05), part 3

G50 detection results

0.404679587	0.278938277	**0.246636021**
0.415665071	0.281670236	**0.247715152**
0.457869706	0.286161206	**0.252918492**
0.536263867	0.288924399	**0.257869706**
0.736757547	0.290018852	**0.258840077**
0.818850194	0.30912883	**0.264096399**
0.829951166	0.310278733	**0.272010685**
0.888998927	0.318850194	**0.272752887**
0.903987475	0.32130244	**0.278938277**
1.038088677	0.329951166	**0.281670236**
1.32130244	0.343412229	**0.286161206**
2.190003869	0.344817988	**0.290018852**
2.281670236	0.356073613	**0.30912883**
3.226002711	0.358934334	**0.310063628**
4.272752887	0.360353945	**0.318850194**
4.398256648	0.410831019	**0.32130244**
5.126603189	0.490060047	**0.329951166**
5.144168255	0.530945409	**0.343412229**
5.19187195	0.546156263	**0.344817988**
6.201529341	0.623983102	**0.354078839**
7.223983102	0.657869706	**0.356073613**
7.242380883	0.676788458	**0.358934334**
8.343412229	0.714324743	**0.360353945**
9.152379851	0.740523837	**0.398256648**
10.2889244	0.802724068	**0.519304545**

Page 13

Figure 27: Grubbs' detection results table (N=50, P=0.05), part 4

G50 detection results

979.2019535	**0.999964324**	**10.23675755**

Page 14

Figure 28: Grubbs' detection results table (N=50, P=0.05), part 5

A.5 Grubbs' detection results (N=20, P=0.05)

G20 detection results

Grubbs Test Report (n=20, p=0.05)		
50-outlier Set	20-outlier Set	5-outlier Set
-4.146156263	-0.545568371	-0.488924399
-4.130945409	-0.398256648	-0.470278733
-3.090334241	-0.354078839	-0.366802326
-2.078057979	-0.346423253	-0.276788458
-2.044163061	-0.310063628	-0.015096731
-0.366802326	-0.264096399	-0.01026995
-0.354078839	-0.236263867	-0.005653937
-0.344817988	-0.215134841	0.002724068
-0.337359013	-0.128255665	0.006574268
-0.328255665	-0.015096731	0.032929701
-0.314324743	-0.01026995	0.038088677
-0.244278549	-0.005653937	0.039874753
-0.238924479	0.006574268	0.040523837
-0.237096041	0.032929701	0.044163061
-0.236629796	0.038088677	0.045568371
-0.229880069	0.039874753	0.050820797
-0.22042762	0.044163061	0.052086917
-0.191582856	0.050820797	0.063040367
-0.190018852	0.052086917	0.077591673
-0.185123008	0.063040367	0.078057979
-0.17707927	0.077591673	0.085123008
-0.167704529	0.078057979	0.090334241
-0.160113788	0.085123008	0.104679587
-0.146423253	0.090334241	0.10707927

Page 15

Figure 29: Grubbs' detection results table (N=20, P=0.05), part 1

G20 detection results

-0.015096731	0.104679587	0.114324743
-0.01026995	0.10707927	0.126603189
-0.005653937	0.119304545	0.129880069
0.002724068	0.126603189	0.130945409
0.006574268	0.129880069	0.134966154
0.032929701	0.134966154	0.137359013
0.040523837	0.137359013	0.144168255
0.045568371	0.144168255	0.146156263
0.050820797	0.152379851	0.146423253
0.052086917	0.156650712	**0.152379851**
0.063040367	0.160113319	0.156650712
0.077591673	0.160113788	0.160113319
0.119304545	0.164800359	0.160113788
0.134966154	0.165257895	**0.164800359**
0.160113319	0.166802326	0.165257895
0.164800359	0.167704529	0.167704529
0.165257895	0.176948629	0.176948629
0.176948629	0.177335653	0.177335653
0.177335653	0.181445167	0.181445167
0.181445167	0.181658642	**0.181658642**
0.190060047	0.188998927	**0.188998927**
0.193742337	0.190003869	**0.190003869**
0.19457084	**0.191582856**	**0.190060047**
0.194966165	0.19187195	**0.191582856**
0.199964324	0.193742337	0.19187195
0.209487151	0.19457084	0.193742337

Page 16

Figure 30: Grubbs' detection results table (N=20, P=0.05), part 2

G20 detection results

0.210329259	0.194966165	0.19457084
0.210831019	0.196429578	**0.194966165**
0.215134841	**0.201529341**	**0.196429578**
0.217809562	0.201953463	**0.199964324**
0.224237814	0.209487151	**0.201529341**
0.230287079	0.210329259	0.201953463
0.234734033	0.217809562	0.209487151
0.246636021	0.22042762	**0.210329259**
0.247715152	**0.224237814**	0.210831019
0.252918492	**0.226002711**	0.215134841
0.258840077	**0.230287079**	0.217809562
0.272010685	0.234734033	0.22042762
0.276788458	0.236629796	0.223983102
0.278938277	0.236757547	**0.224237814**
0.286161206	0.237096041	**0.226002711**
0.296429578	**0.238924479**	0.228255665
0.30912883	0.242380883	**0.230287079**
0.310063628	0.244278549	0.234734033
0.310278733	0.246636021	**0.236263867**
0.356073613	0.247715152	0.236629796
0.358934334	**0.252918492**	0.237096041
0.360353945	0.258840077	**0.238924479**
0.364096399	0.272010685	0.242380883
0.381658642	**0.272752887**	0.244278549
0.404679587	**0.278938277**	0.246636021
0.415665071	0.281670236	0.247715152

Page 17

Figure 31: Grubbs' detection results table (N=20, P=0.05), part 3

G20 detection results

0.457869706	0.286161206	**0.252918492**
0.536263867	**0.288924399**	0.257869706
0.736757547	0.290018852	0.258840077
0.818850194	0.30912883	0.264096399
0.829951166	**0.310278733**	0.272010685
0.888998927	**0.318850194**	**0.272752887**
0.903987475	0.32130244	**0.278938277**
1.038088677	0.329951166	0.281670236
1.32130244	0.343412229	0.286161206
2.190003869	0.344817988	**0.290018852**
2.281670236	0.356073613	0.30912883
3.226002711	**0.358934334**	**0.310063628**
4.272752887	0.360353945	0.318850194
4.398256648	0.410831019	0.32130244
5.126603189	**0.490060047**	0.329951166
5.144168255	**0.530945409**	0.343412229
5.19187195	0.546156263	0.344817988
6.201529341	0.623983102	**0.354078839**
7.223983102	**0.657869706**	0.356073613
7.242380883	0.676788458	**0.358934334**
8.343412229	**0.714324743**	0.360353945
9.152379851	0.740523837	**0.398256648**
10.2889244	**0.802724068**	**0.519304545**
979.2019535	**0.999964324**	**10.23675755**

Page 18

Figure 32: Grubbs' detection results table (N=20, P=0.05), part 4

A.6 Grubbs' detection results (Outliers=50)

O50 detection results

Grubbs Test Report (50 outliers)			
50-Sample	N=100	N=50	N=20
-4.146156263	-4.146156263	-4.146156263	-4.146156263
-4.130945409	-4.130945409	-4.130945409	-4.130945409
-3.090334241	-3.090334241	**-3.090334241**	-3.090334241
-2.078057979	-2.078057979	-2.078057979	-2.078057979
-2.044163061	-2.044163061	-2.044163061	-2.044163061
-0.366802326	-0.366802326	-0.366802326	-0.366802326
-0.354078839	-0.354078839	-0.354078839	-0.354078839
-0.344817988	-0.344817988	**-0.344817988**	-0.344817988
-0.337359013	-0.337359013	-0.337359013	-0.337359013
-0.328255665	-0.328255665	-0.328255665	-0.328255665
-0.314324743	-0.314324743	**-0.314324743**	-0.314324743
-0.244278549	-0.244278549	-0.244278549	-0.244278549
-0.238924479	-0.238924479	**-0.238924479**	-0.238924479
-0.237096041	-0.237096041	-0.237096041	-0.237096041
-0.236629796	-0.236629796	**-0.236629796**	-0.236629796
-0.229880069	-0.229880069	**-0.229880069**	-0.229880069
-0.22042762	-0.22042762	-0.22042762	-0.22042762
-0.191582856	-0.191582856	**-0.191582856**	-0.191582856
-0.190018852	-0.190018852	-0.190018852	-0.190018852
-0.185123008	-0.185123008	**-0.185123008**	-0.185123008
-0.17707927	-0.17707927	**-0.17707927**	-0.17707927
-0.167704529	-0.167704529	**-0.167704529**	-0.167704529
-0.160113788	-0.160113788	**-0.160113788**	-0.160113788

Page 19

Figure 33: Grubbs' detection results table (Outlier =50), part 1

O50 detection results

-0.146423253	-0.146423253	**-0.146423253**	-0.146423253
-0.015096731	-0.015096731	-0.015096731	-0.015096731
-0.01026995	-0.01026995	-0.01026995	-0.01026995
-0.005653937	-0.005653937	**-0.005653937**	-0.005653937
0.002724068	0.002724068	**0.002724068**	0.002724068
0.006574268	0.006574268	0.006574268	0.006574268
0.032929701	0.032929701	**0.032929701**	0.032929701
0.040523837	0.040523837	0.040523837	0.040523837
0.045568371	0.045568371	0.045568371	0.045568371
0.050820797	0.050820797	0.050820797	0.050820797
0.052086917	0.052086917	**0.052086917**	0.052086917
0.063040367	0.063040367	**0.063040367**	0.063040367
0.077591673	**0.077591673**	0.077591673	0.077591673
0.119304545	**0.119304545**	0.119304545	0.119304545
0.134966154	**0.134966154**	0.134966154	0.134966154
0.160113319	**0.160113319**	**0.160113319**	0.160113319
0.164800359	**0.164800359**	0.164800359	**0.164800359**
0.165257895	**0.165257895**	0.165257895	0.165257895
0.176948629	**0.176948629**	0.176948629	0.176948629
0.177335653	**0.177335653**	0.177335653	0.177335653
0.181445167	**0.181445167**	**0.181445167**	0.181445167
0.190060047	**0.190060047**	**0.190060047**	**0.190060047**
0.193742337	**0.193742337**	**0.193742337**	0.193742337
0.19457084	**0.19457084**	**0.19457084**	0.19457084
0.194966165	**0.194966165**	**0.194966165**	0.194966165

Page 20

Figure 34: Grubbs' detection results (Outlier =50), part 2

O50 detection results

0.199964324	**0.199964324**	**0.199964324**	0.199964324
0.209487151	**0.209487151**	**0.209487151**	**0.209487151**
0.210329259	**0.210329259**	**0.210329259**	0.210329259
0.210831019	**0.210831019**	**0.210831019**	**0.210831019**
0.215134841	**0.215134841**	0.215134841	0.215134841
0.217809562	**0.217809562**	**0.217809562**	**0.217809562**
0.224237814	**0.224237814**	0.224237814	**0.224237814**
0.230287079	**0.230287079**	**0.230287079**	**0.230287079**
0.234734033	**0.234734033**	0.234734033	0.234734033
0.246636021	**0.246636021**	**0.246636021**	**0.246636021**
0.247715152	**0.247715152**	0.247715152	0.247715152
0.252918492	**0.252918492**	**0.252918492**	**0.252918492**
0.258840077	**0.258840077**	0.258840077	0.258840077
0.272010685	**0.272010685**	0.272010685	0.272010685
0.276788458	**0.276788458**	0.276788458	0.276788458
0.278938277	**0.278938277**	**0.278938277**	**0.278938277**
0.286161206	**0.286161206**	0.286161206	0.286161206
0.296429578	**0.296429578**	**0.296429578**	0.296429578
0.30912883	**0.30912883**	**0.30912883**	**0.30912883**
0.310063628	**0.310063628**	0.310063628	0.310063628
0.310278733	**0.310278733**	0.310278733	**0.310278733**
0.356073613	**0.356073613**	0.356073613	0.356073613
0.358934334	**0.358934334**	**0.358934334**	0.358934334
0.360353945	**0.360353945**	**0.360353945**	**0.360353945**
0.364096399	**0.364096399**	0.364096399	0.364096399

Page 21

Figure 35: Grubbs' detection results (Outlier =50), part 3

O50 detection results

0.381658642	**0.381658642**	**0.381658642**	**0.381658642**
0.404679587	**0.404679587**	0.404679587	0.404679587
0.415665071	**0.415665071**	0.415665071	0.415665071
0.457869706	**0.457869706**	**0.457869706**	**0.457869706**
0.536263867	**0.536263867**	**0.536263867**	0.536263867
0.736757547	**0.736757547**	**0.736757547**	**0.736757547**
0.818850194	**0.818850194**	**0.818850194**	0.818850194
0.829951166	**0.829951166**	**0.829951166**	**0.829951166**
0.888998927	**0.888998927**	**0.888998927**	**0.888998927**
0.903987475	**0.903987475**	**0.903987475**	0.903987475
1.038088677	**1.038088677**	**1.038088677**	**1.038088677**
1.32130244	**1.32130244**	**1.32130244**	**1.32130244**
2.190003869	**2.190003869**	**2.190003869**	2.190003869
2.281670236	**2.281670236**	2.281670236	**2.281670236**
3.226002711	**3.226002711**	**3.226002711**	3.226002711
4.272752887	**4.272752887**	**4.272752887**	4.272752887
4.398256648	**4.398256648**	**4.398256648**	4.398256648
5.126603189	**5.126603189**	5.126603189	**5.126603189**
5.144168255	**5.144168255**	**5.144168255**	5.144168255
5.19187195	**5.19187195**	**5.19187195**	**5.19187195**
6.201529341	**6.201529341**	6.201529341	**6.201529341**
7.223983102	**7.223983102**	7.223983102	7.223983102
7.242380883	**7.242380883**	7.242380883	7.242380883
8.343412229	**8.343412229**	8.343412229	8.343412229
9.152379851	**9.152379851**	9.152379851	**9.152379851**

Page 22

Figure 36: Grubbs' detection results (Outlier =50), part 4

O50 detection results

10.2889244	**10.2889244**	**10.2889244**	**10.2889244**
979.2019535	**979.2019535**	979.2019535	**979.2019535**

Page 23

Figure 37: Grubbs' detection results (Outlier =50), part 5

A.7 Grubbs' detection results (Outliers =20)

O20 dectection results

Grubbs Test Report (20 outliers)			
20-Sample	N=100	N=50	N=20
-0.545568371	-0.545568371	-0.545568371	-0.545568371
-0.398256648	-0.398256648	-0.398256648	-0.398256648
-0.354078839	-0.354078839	-0.354078839	-0.354078839
-0.346423253	-0.346423253	-0.346423253	-0.346423253
-0.310063628	-0.310063628	-0.310063628	-0.310063628
-0.264096399	-0.264096399	-0.264096399	-0.264096399
-0.236263867	-0.236263867	-0.236263867	-0.236263867
-0.215134841	-0.215134841	-0.215134841	-0.215134841
-0.128255665	-0.128255665	-0.128255665	-0.128255665
-0.015096731	-0.015096731	-0.015096731	-0.015096731
-0.01026995	-0.01026995	-0.01026995	-0.01026995
-0.005653937	-0.005653937	-0.005653937	-0.005653937
0.006574268	0.006574268	0.006574268	0.006574268
0.032929701	0.032929701	0.032929701	0.032929701
0.038088677	0.038088677	0.038088677	0.038088677
0.039874753	0.039874753	0.039874753	0.039874753
0.044163061	0.044163061	0.044163061	0.044163061
0.050820797	0.050820797	0.050820797	0.050820797
0.052086917	0.052086917	0.052086917	0.052086917
0.063040367	0.063040367	0.063040367	0.063040367
0.077591673	0.077591673	0.077591673	0.077591673
0.078057979	0.078057979	0.078057979	0.078057979
0.085123008	0.085123008	0.085123008	0.085123008

Page 24

Figure 38: Grubbs' detection results (Outlier =20), part 1

O20 dectection results

0.090334241	0.090334241	0.090334241	0.090334241
0.104679587	0.104679587	0.104679587	0.104679587
0.10707927	0.10707927	0.10707927	0.10707927
0.119304545	0.119304545	0.119304545	0.119304545
0.126603189	0.126603189	0.126603189	0.126603189
0.129880069	0.129880069	0.129880069	0.129880069
0.134966154	0.134966154	0.134966154	0.134966154
0.137359013	0.137359013	0.137359013	0.137359013
0.144168255	0.144168255	0.144168255	0.144168255
0.152379851	0.152379851	0.152379851	0.152379851
0.156650712	0.156650712	0.156650712	0.156650712
0.160113319	0.160113319	0.160113319	0.160113319
0.160113788	0.160113788	0.160113788	0.160113788
0.164800359	0.164800359	0.164800359	0.164800359
0.165257895	0.165257895	0.165257895	0.165257895
0.166802326	0.166802326	0.166802326	0.166802326
0.167704529	0.167704529	0.167704529	0.167704529
0.176948629	0.176948629	0.176948629	0.176948629
0.177335653	0.177335653	0.177335653	0.177335653
0.181445167	0.181445167	0.181445167	0.181445167
0.181658642	0.181658642	0.181658642	0.181658642
0.188998927	0.188998927	0.188998927	0.188998927
0.190003869	0.190003869	0.190003869	0.190003869
0.191582856	0.191582856	0.191582856	**0.191582856**
0.19187195	0.19187195	0.19187195	0.19187195

Page 25

Figure 39: Grubbs' detection results (Outlier =20), part 2

O20 dectection results

0.193742337	0.193742337	0.193742337	0.193742337
0.19457084	0.19457084	0.19457084	0.19457084
0.194966165	**0.194966165**	0.194966165	0.194966165
0.196429578	**0.196429578**	0.196429578	0.196429578
0.201529341	**0.201529341**	0.201529341	**0.201529341**
0.201953463	**0.201953463**	0.201953463	0.201953463
0.209487151	**0.209487151**	0.209487151	0.209487151
0.210329259	**0.210329259**	0.210329259	0.210329259
0.217809562	**0.217809562**	0.217809562	0.217809562
0.22042762	**0.22042762**	0.22042762	0.22042762
0.224237814	**0.224237814**	0.224237814	**0.224237814**
0.226002711	**0.226002711**	0.226002711	**0.226002711**
0.230287079	**0.230287079**	0.230287079	**0.230287079**
0.234734033	**0.234734033**	0.234734033	0.234734033
0.236629796	**0.236629796**	0.236629796	0.236629796
0.236757547	**0.236757547**	0.236757547	0.236757547
0.237096041	**0.237096041**	0.237096041	0.237096041
0.238924479	**0.238924479**	0.238924479	**0.238924479**
0.242380883	**0.242380883**	0.242380883	0.242380883
0.244278549	**0.244278549**	0.244278549	0.244278549
0.246636021	**0.246636021**	0.246636021	0.246636021
0.247715152	**0.247715152**	0.247715152	0.247715152
0.252918492	**0.252918492**	0.252918492	**0.252918492**
0.258840077	**0.258840077**	0.258840077	0.258840077
0.272010685	**0.272010685**	0.272010685	0.272010685

Page 26

Figure 40: Grubbs' detection results (Outlier =20), part 3

O20 dectection results

0.272752887	**0.272752887**	0.272752887	**0.272752887**
0.278938277	**0.278938277**	0.278938277	**0.278938277**
0.281670236	**0.281670236**	0.281670236	0.281670236
0.286161206	**0.286161206**	0.286161206	0.286161206
0.288924399	**0.288924399**	0.288924399	**0.288924399**
0.290018852	**0.290018852**	0.290018852	0.290018852
0.30912883	**0.30912883**	0.30912883	0.30912883
0.310278733	**0.310278733**	0.310278733	**0.310278733**
0.318850194	**0.318850194**	0.318850194	**0.318850194**
0.32130244	**0.32130244**	0.32130244	0.32130244
0.329951166	**0.329951166**	0.329951166	0.329951166
0.343412229	**0.343412229**	0.343412229	0.343412229
0.344817988	**0.344817988**	0.344817988	0.344817988
0.356073613	**0.356073613**	0.356073613	0.356073613
0.358934334	**0.358934334**	0.358934334	**0.358934334**
0.360353945	**0.360353945**	0.360353945	0.360353945
0.410831019	**0.410831019**	0.410831019	0.410831019
0.490060047	**0.490060047**	0.490060047	**0.490060047**
0.530945409	**0.530945409**	0.530945409	**0.530945409**
0.546156263	**0.546156263**	0.546156263	0.546156263
0.623983102	**0.623983102**	0.623983102	0.623983102
0.657869706	**0.657869706**	0.657869706	**0.657869706**
0.676788458	**0.676788458**	0.676788458	0.676788458
0.714324743	**0.714324743**	0.714324743	**0.714324743**
0.740523837	**0.740523837**	0.740523837	0.740523837

Page 27

Figure 41: Grubbs' detection results (Outlier =20), part 4

O20 dectection results

0.802724068	**0.802724068**	0.802724068	**0.802724068**
0.999964324	**0.999964324**	**0.999964324**	**0.999964324**

Page 28

Figure 42: Grubbs' detection results (Outlier =20), part 5

A.8 Grubbs' detection results (Outliers=5)

O5 detection results

Grubbs Test Report (5 outliers)			
20-Sample	N=100	N=50	N=20
-0.488924399	-0.488924399	-0.488924399	-0.488924399
-0.470278733	-0.470278733	-0.470278733	-0.470278733
-0.366802326	-0.366802326	-0.366802326	-0.366802326
-0.276788458	-0.276788458	-0.276788458	-0.276788458
-0.015096731	-0.015096731	-0.015096731	-0.015096731
-0.01026995	-0.01026995	-0.01026995	-0.01026995
-0.005653937	-0.005653937	-0.005653937	-0.005653937
0.002724068	0.002724068	0.002724068	0.002724068
0.006574268	0.006574268	0.006574268	0.006574268
0.032929701	0.032929701	0.032929701	0.032929701
0.038088677	0.038088677	0.038088677	0.038088677
0.039874753	0.039874753	0.039874753	0.039874753
0.040523837	0.040523837	0.040523837	0.040523837
0.044163061	0.044163061	0.044163061	0.044163061
0.045568371	0.045568371	0.045568371	0.045568371
0.050820797	0.050820797	0.050820797	0.050820797
0.052086917	0.052086917	0.052086917	0.052086917
0.063040367	0.063040367	0.063040367	0.063040367
0.077591673	0.077591673	0.077591673	0.077591673
0.078057979	0.078057979	0.078057979	0.078057979
0.085123008	0.085123008	0.085123008	0.085123008
0.090334241	0.090334241	0.090334241	0.090334241
0.104679587	0.104679587	0.104679587	0.104679587

Page 29

Figure 43: Grubbs' detection results (Outlier =5), part 1

O5 detection results

0.10707927	0.10707927	0.10707927	0.10707927
0.114324743	0.114324743	**0.114324743**	0.114324743
0.126603189	0.126603189	0.126603189	0.126603189
0.129880069	0.129880069	**0.129880069**	0.129880069
0.130945409	0.130945409	0.130945409	0.130945409
0.134966154	0.134966154	0.134966154	0.134966154
0.137359013	0.137359013	0.137359013	0.137359013
0.144168255	0.144168255	**0.144168255**	0.144168255
0.146156263	0.146156263	0.146156263	0.146156263
0.146423253	0.146423253	**0.146423253**	0.146423253
0.152379851	0.152379851	0.152379851	**0.152379851**
0.156650712	0.156650712	0.156650712	0.156650712
0.160113319	0.160113319	**0.160113319**	0.160113319
0.160113788	0.160113788	**0.160113788**	0.160113788
0.164800359	**0.164800359**	0.164800359	**0.164800359**
0.165257895	**0.165257895**	0.165257895	0.165257895
0.167704529	**0.167704529**	**0.167704529**	0.167704529
0.176948629	**0.176948629**	0.176948629	0.176948629
0.177335653	**0.177335653**	**0.177335653**	0.177335653
0.181445167	**0.181445167**	**0.181445167**	0.181445167
0.181658642	**0.181658642**	**0.181658642**	**0.181658642**
0.188998927	**0.188998927**	**0.188998927**	**0.188998927**
0.190003869	**0.190003869**	**0.190003869**	**0.190003869**
0.190060047	**0.190060047**	**0.190060047**	**0.190060047**
0.191582856	**0.191582856**	**0.191582856**	**0.191582856**

Page 30

Figure 44: Grubbs' detection results (Outlier =5), part 2

O5 detection results

0.19187195	**0.19187195**	**0.19187195**	0.19187195
0.193742337	**0.193742337**	**0.193742337**	0.193742337
0.19457084	**0.19457084**	**0.19457084**	0.19457084
0.194966165	**0.194966165**	**0.194966165**	**0.194966165**
0.196429578	**0.196429578**	**0.196429578**	**0.196429578**
0.199964324	**0.199964324**	**0.199964324**	**0.199964324**
0.201529341	**0.201529341**	**0.201529341**	**0.201529341**
0.201953463	**0.201953463**	**0.201953463**	0.201953463
0.209487151	**0.209487151**	**0.209487151**	0.209487151
0.210329259	**0.210329259**	**0.210329259**	**0.210329259**
0.210831019	**0.210831019**	**0.210831019**	0.210831019
0.215134841	**0.215134841**	**0.215134841**	0.215134841
0.217809562	**0.217809562**	**0.217809562**	0.217809562
0.22042762	**0.22042762**	**0.22042762**	0.22042762
0.223983102	**0.223983102**	**0.223983102**	0.223983102
0.224237814	**0.224237814**	**0.224237814**	**0.224237814**
0.226002711	**0.226002711**	**0.226002711**	**0.226002711**
0.228255665	**0.228255665**	**0.228255665**	0.228255665
0.230287079	**0.230287079**	**0.230287079**	**0.230287079**
0.234734033	**0.234734033**	**0.234734033**	0.234734033
0.236263867	**0.236263867**	**0.236263867**	**0.236263867**
0.236629796	**0.236629796**	**0.236629796**	0.236629796
0.237096041	**0.237096041**	**0.237096041**	0.237096041
0.238924479	**0.238924479**	**0.238924479**	**0.238924479**
0.242380883	**0.242380883**	**0.242380883**	0.242380883

Page 31

Figure 45: Grubbs' detection results (Outlier =5), part 3

O5 detection results

0.244278549	**0.244278549**	**0.244278549**	0.244278549
0.246636021	**0.246636021**	**0.246636021**	0.246636021
0.247715152	**0.247715152**	**0.247715152**	0.247715152
0.252918492	**0.252918492**	**0.252918492**	**0.252918492**
0.257869706	**0.257869706**	**0.257869706**	0.257869706
0.258840077	**0.258840077**	**0.258840077**	0.258840077
0.264096399	**0.264096399**	**0.264096399**	0.264096399
0.272010685	**0.272010685**	**0.272010685**	0.272010685
0.272752887	**0.272752887**	**0.272752887**	**0.272752887**
0.278938277	**0.278938277**	**0.278938277**	**0.278938277**
0.281670236	**0.281670236**	**0.281670236**	0.281670236
0.286161206	**0.286161206**	**0.286161206**	0.286161206
0.290018852	**0.290018852**	**0.290018852**	**0.290018852**
0.30912883	**0.30912883**	**0.30912883**	0.30912883
0.310063628	**0.310063628**	**0.310063628**	**0.310063628**
0.318850194	**0.318850194**	**0.318850194**	0.318850194
0.32130244	**0.32130244**	**0.32130244**	0.32130244
0.329951166	**0.329951166**	**0.329951166**	0.329951166
0.343412229	**0.343412229**	**0.343412229**	0.343412229
0.344817988	**0.344817988**	**0.344817988**	0.344817988
0.354078839	**0.354078839**	**0.354078839**	**0.354078839**
0.356073613	**0.356073613**	**0.356073613**	0.356073613
0.358934334	**0.358934334**	**0.358934334**	**0.358934334**
0.360353945	**0.360353945**	**0.360353945**	0.360353945
0.398256648	**0.398256648**	**0.398256648**	**0.398256648**

Page 32

Figure 46: Grubbs' detection results (Outlier =5), part 4

O5 detection results

0.519304545	**0.519304545**	**0.519304545**	**0.519304545**
10.23675755	**10.23675755**	**10.23675755**	**10.23675755**

Page 33

Figure 47: Grubbs' detection results (Outlier =5), part 5

A.9 KNN outlier correction list (K =3)

KNN(K=3) test report

Classification results (K=3)	Correction	Correct Class
{'Sample': 4.272752887, 'Label': '0'}		4.272752887
{'Sample': 0.181445157, 'Label': '1'}		0.181445167
{'Sample': 2.190003869, 'Label': '0'}		2.190003869
{'Sample': 0.199964324, 'Label': '1'}		0.199964324
{'Sample': 10.2889244, 'Label': '0'}		10.2889244
{'Sample': 0.160113319, 'Label': '1'}		0.160113319
{'Sample': 5.144168255, 'Label': '0'}		5.144168255
{'Sample': 0.358934334, 'Label': '1'}		0.358934334
{'Sample': 0.818850194, 'Label': '0'}		0.818850194
{'Sample': 0.002724068, 'Label': '1'}		0.002724068
{'Sample': -0.167704529, 'Label': '0'}		-0.167704529
{'Sample': 3.226002711, 'Label': '0'}		3.226002711
{'Sample': -0.005653937, 'Label': '1'}		-0.005653937
{'Sample': 0.903987475, 'Label': '0'}		0.903987475
{'Sample': -0.17707927, 'Label': '0'}		-0.17707927
{'Sample': 0.296429578, 'Label': '1'}		0.296429578
{'Sample': 4.398256648, 'Label': '0'}		4.398256648
{'Sample': 0.194966165, 'Label': '1'}		0.194966165
{'Sample': 0.536263857, 'Label': '0'}		0.536263867
{'Sample': 0.210329259, 'Label': '1'}		0.210329259
{'Sample': -0.146423253, 'Label': '0'}		-0.146423253
{'Sample': -0.238924479, 'Label': '0'}		-0.238924479
{'Sample': 0.278938277, 'Label': '1'}		0.278938277
{'Sample': 0.381658642, 'Label': '1'}	0	0.381658642
{'Sample': -0.229880069, 'Label': '0'}		-0.229880069
{'Sample': 0.230287079, 'Label': '1'}		0.230287079
{'Sample': 0.888998927, 'Label': '0'}		0.888998927
{'Sample': -0.191582856, 'Label': '0'}		-0.191582856

Figure 48: KNN outlier correction list (K =3), part 1

KNN(K=3) test report

{'Sample': 0.252918492, 'Label': '1'}		0.252918492
{'Sample': -0.314324743, 'Label': '0'}		-0.314324743
{'Sample': 5.126603189, 'Label': '0'}		5.126603189
{'Sample': 0.310278733, 'Label': '1'}		0.310278733
{'Sample': -4.130945409, 'Label': '0'}		-4.130945409
{'Sample': 0.134966154, 'Label': '1'}		0.134966154
{'Sample': 6.201529341, 'Label': '0'}		6.201529341
{'Sample': 0.190060047, 'Label': '1'}		0.190060047
{'Sample': 9.152379851, 'Label': '0'}		9.152379851
{'Sample': 0.224237814, 'Label': '1'}		0.224237814
{'Sample': 0.164800359, 'Label': '1'}		0.164800359
{'Sample': 0.006574268, 'Label': '1'}		0.006574268
{'Sample': 8.343412229, 'Label': '0'}		8.343412229
{'Sample': 0.215134841, 'Label': '1'}		0.215134841
{'Sample': 0.193742337, 'Label': '1'}		0.193742337
{'Sample': -2.044163061, 'Label': '0'}		-2.044163061
{'Sample': 0.286161206, 'Label': '1'}		0.286161206
{'Sample': -2.078057979, 'Label': '0'}		-2.078057979
{'Sample': 0.258840077, 'Label': '1'}		0.258840077
{'Sample': 0.119304545, 'Label': '1'}		0.119304545
{'Sample': -0.237096041, 'Label': '0'}		-0.237096041
{'Sample': 0.077591673, 'Label': '1'}		0.077591673
{'Sample': -0.328255665, 'Label': '0'}		-0.328255665
{'Sample': 7.223983102, 'Label': '0'}		7.223983102
{'Sample': 0.165257895, 'Label': '1'}		0.165257895
{'Sample': 0.040523837, 'Label': '1'}		0.040523837
{'Sample': 0.356073613, 'Label': '1'}	0	0.356073613
{'Sample': -4.146156263, 'Label': '0'}		-4.146156263
{'Sample': 0.19457084, 'Label': '1'}		0.19457084

Figure 49: KNN outlier correction list (K =3), part 2

KNN(K=3) test report

{'Sample': 7.242380883, 'Label': '0'}		7.242380883
{'Sample': -0.244278549, 'Label': '0'}		-0.244278549
{'Sample': 0.050820797, 'Label': '1'}		0.050820797
{'Sample': 0.045568371, 'Label': '1'}		0.045568371
{'Sample': -0.337359013, 'Label': '0'}		-0.337359013
{'Sample': 0.415665071, 'Label': '1'}	0	0.415665071
{'Sample': -0.22042762, 'Label': '0'}		-0.22042762
{'Sample': 0.276788458, 'Label': '1'}		0.276788458
{'Sample': 0.234734033, 'Label': '1'}		0.234734033
{'Sample': -0.354078839, 'Label': '0'}		-0.354078839
{'Sample': 0.176948629, 'Label': '1'}		0.176948629
{'Sample': 0.404679587, 'Label': '1'}	0	0.404679587
{'Sample': 0.247715152, 'Label': '1'}		0.247715152
{'Sample': 0.364096399, 'Label': '1'}	0	0.364096399
{'Sample': -0.015096731, 'Label': '1'}		-0.015096731
{'Sample': 0.272010685, 'Label': '1'}		0.272010685
{'Sample': 2.281670236, 'Label': '0'}		2.281670236
{'Sample': 0.310063628, 'Label': '1'}		0.310063628
{'Sample': -0.01026995, 'Label': '1'}		-0.01026995
{'Sample': 0.177335653, 'Label': '1'}		0.177335653
{'Sample': -0.190018852, 'Label': '0'}		-0.190018852
{'Sample': -0.366802326, 'Label': '0'}		-0.366802326
{'Sample': -0.244278549, 'Label': '0'}		-0.244278549

Figure 50: KNN outlier correction list (K =3), part 3

A.10 KNN outlier correction list (K=5)

K=5

Classification results (K=5)	Correction	Correct Class
{'Sample': 4.272752887, 'Label': '0'}		4.272752887
{'Sample': 0.181445157, 'Label': '1'}		0.181445167
{'Sample': 2.190003869, 'Label': '0'}		2.190003869
{'Sample': 0.199964324, 'Label': '1'}		0.199964324
{'Sample': 10.2889244, 'Label': '0'}		10.2889244
{'Sample': 0.160113319, 'Label': '1'}		0.160113319
{'Sample': 5.144168255, 'Label': '0'}		5.144168255
{'Sample': 0.358934334, 'Label': '1'}		0.358934334
{'Sample': 0.818850194, 'Label': '0'}		0.818850194
{'Sample': 0.002724068, 'Label': '1'}		0.002724068
{'Sample': -0.167704529, 'Label': '0'}		-0.167704529
{'Sample': 3.226002711, 'Label': '0'}		3.226002711
{'Sample': -0.005653937, 'Label': '1'}		-0.005653937
{'Sample': 0.903987475, 'Label': '0'}		0.903987475
{'Sample': -0.17707927, 'Label': '0'}		-0.17707927
{'Sample': 0.296429578, 'Label': '1'}		0.296429578
{'Sample': 4.398256648, 'Label': '0'}		4.398256648
{'Sample': 0.194966165, 'Label': '1'}		0.194965165
{'Sample': 0.536263857, 'Label': '1'}		0.536263867
{'Sample': 0.210329259, 'Label': '1'}		0.210329259
{'Sample': -0.146423253, 'Label': '0'}		-0.146423253
{'Sample': -0.238924479, 'Label': '0'}		-0.238924479
{'Sample': 0.278938277, 'Label': '1'}		0.278938277
{'Sample': 0.381658642, 'Label': '1'}		0.381658642
{'Sample': -0.229880069, 'Label': '0'}		-0.229880069
{'Sample': 0.230287079, 'Label': '1'}		0.230287079
{'Sample': 0.888998927, 'Label': '0'}		0.888998927
{'Sample': -0.191582856, 'Label': '0'}		-0.191582856

Figure 51: KNN outlier correction list (K =5), part 1

K=5

{'Sample': 0.252918492, 'Label': '1'}		0.252918492
{'Sample': -0.314324743, 'Label': '0'}		-0.314324743
{'Sample': 5.126603189, 'Label': '0'}		5.126603189
{'Sample': 0.310278733, 'Label': '1'}		0.310278733
{'Sample': -4.130945409, 'Label': '0'}		-4.130945409
{'Sample': 0.134966154, 'Label': '1'}		0.134966154
{'Sample': 6.201529341, 'Label': '0'}		6.201529341
{'Sample': 0.190060047, 'Label': '1'}		0.190060047
{'Sample': 9.152379851, 'Label': '0'}		9.152379851
{'Sample': 0.224237814, 'Label': '1'}		0.224237814
{'Sample': 0.164800359, 'Label': '1'}		0.164800359
{'Sample': 0.006574268, 'Label': '1'}		0.006574268
{'Sample': 8.343412229, 'Label': '0'}		8.343412229
{'Sample': 0.215134841, 'Label': '1'}		0.215134841
{'Sample': 0.193742337, 'Label': '1'}		0.193742337
{'Sample': -2.044163061, 'Label': '0'}		-2.044163061
{'Sample': 0.286161206, 'Label': '1'}		0.286161206
{'Sample': -2.078057979, 'Label': '0'}		-2.078057979
{'Sample': 0.258840077, 'Label': '1'}		0.258840077
{'Sample': 0.119304545, 'Label': '1'}		0.119304545
{'Sample': -0.237096041, 'Label': '0'}		-0.237096041
{'Sample': 0.077591673, 'Label': '1'}		0.077591673
{'Sample': -0.328255665, 'Label': '0'}		-0.328255665
{'Sample': 7.223983102, 'Label': '0'}		7.223983102
{'Sample': 0.165257895, 'Label': '1'}		0.165257895
{'Sample': 0.040523837, 'Label': '1'}		0.040523837
{'Sample': 0.356073613, 'Label': '1'}	0	0.356073613
{'Sample': -4.146156263, 'Label': '0'}		-4.146156263
{'Sample': 0.19457084, 'Label': '1'}		0.19457084

Figure 52: KNN outlier correction list (K =5), part 2

K=5

{'Sample': 7.242380883, 'Label': '0'}		7.242380883
{'Sample': -0.244278549, 'Label': '0'}		-0.244278549
{'Sample': 0.050820797, 'Label': '1'}		0.050820797
{'Sample': 0.045568371, 'Label': '1'}		0.045568371
{'Sample': -0.337359013, 'Label': '0'}		-0.337359013
{'Sample': 0.415665071, 'Label': '1'}	0	0.415665071
{'Sample': -0.22042762, 'Label': '0'}		-0.22042762
{'Sample': 0.276788458, 'Label': '1'}		0.276788458
{'Sample': 0.234734033, 'Label': '1'}		0.234734033
{'Sample': -0.354078839, 'Label': '0'}		-0.354078839
{'Sample': 0.176948629, 'Label': '1'}		0.176948629
{'Sample': 0.404679587, 'Label': '1'}	0	0.404679587
{'Sample': 0.247715152, 'Label': '1'}		0.247715152
{'Sample': 0.364096399, 'Label': '1'}	0	0.364095399
{'Sample': -0.015096731, 'Label': '1'}		-0.015096731
{'Sample': 0.272010685, 'Label': '1'}		0.272010685
{'Sample': 2.281670236, 'Label': '0'}		2.281670236
{'Sample': 0.310063628, 'Label': '1'}		0.310063628
{'Sample': -0.01026995, 'Label': '1'}		-0.01026995
{'Sample': 0.177335653, 'Label': '1'}		0.177335653
{'Sample': -0.190018852, 'Label': '0'}		-0.190018852
{'Sample': -0.366802326, 'Label': '0'}		-0.366802326
{'Sample': -0.244278549, 'Label': '0'}		-0.244278549

Figure 53: KNN outlier correction list (K =5), part 3

A.11 KNN outlier correction list (K =7)

KNN(K=7) test report

Classification results (K=7)	Correction	Correct Class
{'Sample': 4.272752857, 'Label': '0'}		4.272752887
{'Sample': 0.181445167, 'Label': '1'}		0.181445167
{'Sample': 2.190003859, 'Label': '0'}		2.190003869
{'Sample': 0.199964324, 'Label': '1'}		0.199964324
{'Sample': 10.2889244, 'Label': '0'}		10.2889244
{'Sample': 0.160113319, 'Label': '1'}		0.160113319
{'Sample': 5.144168255, 'Label': '0'}		5.144168255
{'Sample': 0.358934334, 'Label': '1'}		0.358934334
{'Sample': 0.818850194, 'Label': '0'}		0.818850194
{'Sample': 0.002724068, 'Label': '1'}		0.002724068
{'Sample': -0.167704529, 'Label': '0'}		-0.167704529
{'Sample': 3.226002711, 'Label': '0'}		3.226002711
{'Sample': -0.005653937, 'Label': '1'}	0	-0.005653937
{'Sample': 0.903987475, 'Label': '0'}		0.903987475
{'Sample': -0.17707927, 'Label': '0'}		-0.17707927
{'Sample': 0.296429578, 'Label': '1'}		0.296429578
{'Sample': 4.398256648, 'Label': '0'}		4.398256648
{'Sample': 0.194966165, 'Label': '1'}		0.194966165
{'Sample': 0.536263857, 'Label': '1'}	0	0.536263867
{'Sample': 0.210329259, 'Label': '1'}		0.210329259
{'Sample': -0.146423253, 'Label': '0'}		-0.146423253
{'Sample': -0.238924479, 'Label': '0'}		-0.238924479
{'Sample': 0.278938277, 'Label': '1'}		0.278938277
{'Sample': 0.381658642, 'Label': '1'}	0	0.381658642
{'Sample': -0.229880069, 'Label': '0'}		-0.229880069
{'Sample': 0.230267079, 'Label': '1'}		0.230267079
{'Sample': 0.888998927, 'Label': '0'}		0.888998927
{'Sample': -0.191582856, 'Label': '0'}		-0.191582856
{'Sample': 0.252918492, 'Label': '1'}		0.252918492
{'Sample': -0.314324743, 'Label': '0'}		-0.314324743

Figure 54: KNN outlier correction list (K =7), part 1

KNN(K=7) test report

{'Sample': 5.126603189, 'Label': '0'}		5.126603189
{'Sample': 0.310278733, 'Label': '1'}		0.310278733
{'Sample': -4.130945409, 'Label': '0'}		-4.130945409
{'Sample': 0.134966154, 'Label': '1'}		0.134966154
{'Sample': 6.201529341, 'Label': '0'}		6.201529341
{'Sample': 0.190060047, 'Label': '1'}		0.190060047
{'Sample': 9.152379851, 'Label': '0'}		9.152379851
{'Sample': 0.224237814, 'Label': '1'}		0.224237814
{'Sample': 0.164800359, 'Label': '1'}		0.164800359
{'Sample': 0.006574268, 'Label': '1'}		0.006574268
{'Sample': 8.343412229, 'Label': '0'}		8.343412229
{'Sample': 0.215134841, 'Label': '1'}		0.215134841
{'Sample': 0.193742337, 'Label': '1'}		0.193742337
{'Sample': -2.044163061, 'Label': '0'}		-2.044163061
{'Sample': 0.286161206, 'Label': '1'}		0.286161206
{'Sample': -2.078057979, 'Label': '0'}		-2.078057979
{'Sample': 0.258840077, 'Label': '1'}		0.258840077
{'Sample': 0.119304545, 'Label': '1'}		0.119304545
{'Sample': -0.237096041, 'Label': '0'}		-0.237096041
{'Sample': 0.077591673, 'Label': '1'}		0.077591673
{'Sample': -0.328255665, 'Label': '0'}		-0.328255665
{'Sample': 7.223983102, 'Label': '0'}		7.223983102
{'Sample': 0.165257895, 'Label': '1'}		0.165257895
{'Sample': 0.040523837, 'Label': '1'}		0.040523837
{'Sample': 0.356073613, 'Label': '1'}	0	0.356073613
{'Sample': -4.146156263, 'Label': '0'}		-4.146156263
{'Sample': 0.19457084, 'Label': '1'}		0.19457084
{'Sample': 7.242380883, 'Label': '0'}		7.242380883
{'Sample': -0.244278549, 'Label': '0'}		-0.244278549
{'Sample': 0.050820797, 'Label': '1'}		0.050820797
{'Sample': 0.045568371, 'Label': '1'}		0.045568371

Figure 55: KNN outlier correction list (K =7), part 2

KNN(K=7) test report

{'Sample': -0.337359013, 'Label': '0'}		-0.337359013
{'Sample': 0.415665071, 'Label': '1'}	0	0.415665071
{'Sample': -0.22042762, 'Label': '0'}		-0.22042762
{'Sample': 0.276788458, 'Label': '1'}		0.276788458
{'Sample': 0.234734033, 'Label': '1'}		0.234734033
{'Sample': -0.354078839, 'Label': '0'}		-0.354078839
{'Sample': 0.176948629, 'Label': '1'}		0.176948629
{'Sample': 0.404679587, 'Label': '1'}	0	0.404679587
{'Sample': 0.247715152, 'Label': '1'}		0.247715152
{'Sample': 0.364096399, 'Label': '1'}	0	0.364096399
{'Sample': -0.015096731, 'Label': '1'}		-0.015096731
{'Sample': 0.272010685, 'Label': '1'}		0.272010685
{'Sample': 2.281670236, 'Label': '0'}		2.281670236
{'Sample': 0.310063628, 'Label': '1'}		0.310063628
{'Sample': -0.01026995, 'Label': '1'}		-0.01026995
{'Sample': 0.177335653, 'Label': '1'}		0.177335653
{'Sample': -0.190018852, 'Label': '0'}		-0.190018852
{'Sample': -0.366802326, 'Label': '0'}		-0.366802326
{'Sample': -0.244278549, 'Label': '0'}		-0.244278549

Figure 56: KNN outlier correction list (K =7), part 3

A.12 Source code in Python - Grubbs' Test

```
import random
import math
import sys

#########################################################################
#Define the significant level for Grubbs' test. (N=50)
Gp=2.956
#########################################################################
#Read from file
List=[]
RawSet=[]
f=open("test2A.txt")  # Raw data file
for line in f.readlines():
        line=line.strip('\n')    #Remove '\n' per line.
        List.append(line)
RawSet = [float(_s) for _s in List]  #Convert from literal to value.
f.close()
#########################################################################
class Stats:

    def __init__(self, sequence):
        # sequence of numbers to process
        # convert all items to floats for numerical processing
        self.sequence = [float(item) for item in sequence]

    def sum(self):
        if len(self.sequence) < 1:
            return None
        else:
            return sum(self.sequence)

    def count(self):
        return len(self.sequence)

    def min(self):
        if len(self.sequence) < 1:
            return None
        else:
            return min(self.sequence)

    def max(self):
        if len(self.sequence) < 1:
            return None
        else:
            return max(self.sequence)

    def avg(self):
        if len(self.sequence) < 1:
            return None
        else:
            return sum(self.sequence) / len(self.sequence)
```

```
    def stdev(self):
        if len(self.sequence) < 1:
            return None
        else:
            avg = self.avg()
            stdsq = sum([(i - avg) ** 2 for i in self.sequence])
            stdev = (stdsq / (len(self.sequence) - 1)) ** .5
            return stdev

#####################################################################
print "The␣%d-element␣raw␣data␣set␣are␣:␣\n" %len(RawSet)
print RawSet
print ">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>"
#####################################################################
RawSet.sort()
SortedSet=RawSet

i=True
while (i):
        print '''
        >>>>>>>>>>>>>>>>>>>>>>>>
        Iteration Begins !!!
        >>>>>>>>>>>>>>>>>>>>>>>>\n'''
        print 'Now␣the␣%d-element␣SortedSet␣is␣:␣\n' %len(SortedSet)
        print SortedSet
        s=Stats(SortedSet)
        average=s.avg()
        print 'mean␣␣␣␣␣:␣', average
        stdev=s.stdev()
        print 'Stdev␣␣␣␣:␣', stdev
        maximum=s.max()
        print 'max␣␣␣␣␣␣:␣', maximum
        minimum=s.min()
        print 'min␣␣␣␣␣␣:␣', minimum
        devmax = maximum-average
        print 'devmax␣␣␣:␣', devmax
        devmin = average-minimum
        print 'devmin␣␣␣:␣', devmin
        GreaterDev = max(devmax, devmin)
        print 'GreatDev␣:␣', GreaterDev
        def suspect(GreaterDev):
                if      GreaterDev == devmax:
                        print 'SUSPECT␣␣:␣', maximum
                        return maximum
                else:
                        print 'SUSPECT␣␣:␣', maximum
                        return minimum
        g=math.fabs ((suspect(GreaterDev)-average)/stdev)
        print 'Gi␣␣␣␣␣␣␣:␣', g
        print 'Gp␣␣␣␣␣␣␣:␣', Gp
        def criteria(g,Gp):
            if g>Gp:
```

```
                print 'Gi : ',g,'>','Gp : ',Gp
                print '''
                **************************
                LOOK ! SUSPECT CONFIRMED !
                **************************'''
                if cmp(minimum,g)==0:
                    print 'The suspect', minimum ,' will be removed.'
                    del SortedSet[0]
                else:
                    print 'The suspect', maximum, 'will be removed.'
                    del SortedSet[-1]
                    return True
            else:
                print 'Gi : ',g,'<','Gp : ',Gp
                print '''
                **************************
                Congratulation ! Seems no suspect...
                **************************'''
                return False

        i=criteria(g,Gp)
#####################################################################
#Save to file
length=len(SortedSet)
k=0
f=open('test2A_result.txt','a+')
while k<length:
    print >> f,SortedSet[k]      # Redirect every element to a text file line by line
    k=k+1
f.close()
print "Save to file successfully !\n"
```

A.13 Source code in Python - K-Nearest Neighbour

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

import sys
import math
import heapq
#-----------------------------------------------------------------------------
#Trainning set
#20-element training set consists of 9 normal labelled elements and 11 labelled
st=[{'Sample': 0.246636021,'Label': 1},
    {'Sample': -0.160113788 , 'Label': 0},
    {'Sample': 0.210831019 , 'Label': 1},
    {'Sample': -3.090334241 , 'Label': 0},
    {'Sample': 0.30912883 , 'Label': 1},
    {'Sample': 5.19187195 , 'Label': 0},
    {'Sample': -0.185123008 , 'Label': 0},
    {'Sample': 0.736757547 , 'Label': 0},
    {'Sample': 0.032929701 , 'Label': 1},
    {'Sample': 0.457869706 , 'Label': 0},
    {'Sample': -0.236629796 , 'Label': 0},
    {'Sample': 0.209487151 , 'Label': 1},
    {'Sample': 0.063040367 , 'Label': 1},
    {'Sample': 1.038088677 , 'Label': 0},
    {'Sample': 0.217809562 , 'Label': 1},
    {'Sample': 0.829951166 , 'Label': 0},
    {'Sample': 0.360353945 , 'Label': 1},
    {'Sample': -0.344817988 , 'Label': 0},
    {'Sample': 1.32130244 , 'Label': 0},
    {'Sample': 0.052086917 , 'Label': 1}]
#-----------------------------------------------------------------------------
#Calculate Euclidean
def euclidean(S1,S2):
        distance=0.0
        distance=(S1-S2)**2
        return math.sqrt(distance)
#-----------------------------------------------------------------------------
#Testing set
#80 elements to be tested
ts=[{'Sample':  4.272752887     ,},
{'Sample':      0.181445167     ,},
{'Sample':      2.190003869     ,},
{'Sample':      0.199964324     ,},
{'Sample':      10.2889244      ,},
{'Sample':      0.160113319     ,},
{'Sample':      5.144168255     ,},
{'Sample':      0.358934334     ,},
{'Sample':      0.818850194     ,},
{'Sample':      0.002724068     ,},
{'Sample':      -0.167704529,},
{'Sample':      3.226002711     ,},
{'Sample':      -0.005653937,},
{'Sample':      0.903987475     ,},
{'Sample':      -0.17707927     ,},
```

```
{'Sample':      0.296429578     ,},
{'Sample':      4.398256648     ,},
{'Sample':      0.194966165     ,},
{'Sample':      0.536263867     ,},
{'Sample':      0.210329259     ,},
{'Sample':      -0.146423253,},
{'Sample':      -0.238924479,},
{'Sample':      0.278938277     ,},
{'Sample':      0.381658642     ,},
{'Sample':      -0.229880069,},
{'Sample':      0.230287079     ,},
{'Sample':      0.888998927     ,},
{'Sample':      -0.191582856,},
{'Sample':      0.252918492     ,},
{'Sample':      -0.314324743,},
{'Sample':      5.126603189     ,},
{'Sample':      0.310278733     ,},
{'Sample':      -4.130945409,},
{'Sample':      0.134966154     ,},
{'Sample':      6.201529341     ,},
{'Sample':      0.190060047     ,},
{'Sample':      9.152379851     ,},
{'Sample':      0.224237814     ,},
{'Sample':      0.164800359     ,},
{'Sample':      0.006574268     ,},
{'Sample':      8.343412229     ,},
{'Sample':      0.215134841     ,},
{'Sample':      0.193742337     ,},
{'Sample':      -2.044163061,},
{'Sample':      0.286161206     ,},
{'Sample':      -2.078057979,},
{'Sample':      0.258840077     ,},
{'Sample':      0.119304545     ,},
{'Sample':      -0.237096041,},
{'Sample':      0.077591673     ,},
{'Sample':      -0.328255665,},
{'Sample':      7.223983102     ,},
{'Sample':      0.165257895     ,},
{'Sample':      0.040523837     ,},
{'Sample':      0.356073613     ,},
{'Sample':      -4.146156263,},
{'Sample':      0.19457084      ,},
{'Sample':      7.242380883     ,},
{'Sample':      -0.244278549,},
{'Sample':      0.050820797     ,},
{'Sample':      0.045568371     ,},
{'Sample':      -0.337359013,},
{'Sample':      0.415665071     ,},
{'Sample':      -0.22042762     ,},
{'Sample':      0.276788458     ,},
{'Sample':      0.234734033     ,},
{'Sample':      -0.354078839,},
{'Sample':      0.176948629     ,},
```

```
{'Sample':      0.404679587     ,},
{'Sample':      0.247715152     ,},
{'Sample':      0.364096399     ,},
{'Sample':      -0.015096731,},
{'Sample':      0.272010685     ,},
{'Sample':      2.281670236     ,},
{'Sample':      0.310063628     ,},
{'Sample':      -0.01026995     ,},
{'Sample':      0.177335653     ,},
{'Sample':      -0.190018852,},
{'Sample':      -0.366802326,},
{'Sample':      -0.244278549,}]

#-------------------------------------------------------------------------------
#Measure distance to all trainning samples
#ts = test set, st = tranning sample set
def metric(testpoint,st):
    distlist=[]
    for i in range(len(st)):
        temp=st[i]['Sample']
        distlist.append((euclidean(temp,testpoint),i))
#         print distlist
    distlist.sort()
    print 'Sorted distance list : \n',distlist
    return distlist
#-------------------------------------------------------------------------------
#Derive label for test data by KNN algorithm, assign value 5 to K.
def KNN(testpoint, st, K=5):
    dis=metric(testpoint, st)
    DIS=heapq.nsmallest(K,dis)  #return the most smallest K elements by calling
#     print 'K = ', K
    print 'Current top K-neareast neighbours are : \n', DIS

    label=[]
    def labellist(DIS):
        for i in range(K):
            j=DIS[i][1]
            label.append(st[j]['Label'])
        return label

    KNNlabel=labellist(DIS)
    print 'The label of K-nearest neighbours are : ', KNNlabel

    def decision(label):
        if(label.count(1) > label.count(0)):    #compare the occurrence number
            print 'Classification result : Label 1'
            return '1'
        else:
            print 'Classification result : Label 0'
            return '0'
    return decision(label)

#-------------------------------------------------------------------------------
```

```
#Launch KNN-classification
print '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n'
print '*****************␣␣␣KNN␣classification␣starts␣!!!␣␣␣*****************\n'
print 'TEST␣DATA␣SET␣:␣\n\n', ts, '\n'
print 'Size␣of␣training␣data␣set␣is␣:␣', len(ts)
print 'Current␣K␣=␣5␣\n'
print '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n'
for i in range(len(ts)):
    newlabel=KNN(ts[i]['Sample'],st,K=5)
    print '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n'
    ts[i]['Label']=newlabel

print 'CLASSIFIED␣TEST␣DATA␣SET␣:␣\n\n', ts
print '****************␣␣KNN␣classification␣terminated␣!!!␣␣****************\n'

#--------------------------------------------------------------------------------
#Save to file
length=len(ts)
k=0
f=open('KNN_after_K5.txt','a+')
while k<length:
    print >> f,ts[k]      # Redirect every element to a text file line by line
    k=k+1
f.close()
print "Save␣to␣file␣successfully␣!␣\nPlease␣check␣file␣KNN_after_K5.txt␣for␣details.␣
print '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n'
```