LOG DATA ANOMALY DETECTION AND ANALYSIS FOR AN AIOPS SYSTEM

A TECHNICAL REPORT SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF COMPUTER AND INFORMATION SCIENCES

Supervisors

Dr. Jing Ma

Prof. Edmund M-K Lai

08 February 2022

By

Yubo Huang

School of Engineering, Computer and Mathematical Sciences

Abstract

With the rapid development of information technology and the increasing scale of networks, the security, efficiency, and high-quality operation and maintenance of IT systems are areas of concern. In order to reduce the burden on human operators, AIOps (Artificial Intelligence for IT Operations), which attempts to combine artificial intelligence techniques with IT operation and maintenance, has emerged as a promising approach.

This thesis focuses on anomaly detection and analysis through computer logs. The aim is to construct an AIOps system model based on this study by deconstructing log data through the analysis of historical log data, clarifying the algorithm's feasibility. The following objectives have been achieved to address the mentioned issues.

First, the state of the logging research field, operation and maintenance concepts and ideas on AIOps are analysed. It then analysed the characteristics from different system logs in AIOps scenarios and design a log detection framework, including collecting log data, decoding them, extracting them by LDA(Latent Dirichlet Allocation) topic model. Using "T-SNE"(t-distributed stochastic neighbor embedding) reduces the high-dimensional features to two-dimensional to observe the grouping effect. Three unsupervised algorithms, K-means, DBSCAN and LOF, were chosen to train models for log anomaly detection. They will select the optimal clusters, reduce redundant features and improve model performance. Numerical experiment results show K-means performs better in several tests and can delineate more finely and detect log anomalies

earlier. Finally, an architecture diagram of the AIOps operating and management system was designed, and a preliminary requirements analysis of the AIOps system was conducted.

Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of student

Acknowledgements

First and foremost, I want to express my gratitude to Dr. Jing Ma, under whose supervision I was able to complete this thesis. Academically, her careful instruction over the last year has taught me how to analyze and write scientifically. She has also offered me much advice in terms of career development.

Second, I would like to thank Professor Edmund M-K Lai for pointing out the direction of my thesis, the faults in the experiments, and the appropriate methodologies, which helped me grasp and address certain conceptual difficulties.

I also want to thank Stone Tao, Mrs. Sara, Mrs. Chai, and Mr. Craig as well. These are the most significant people in my life, both as friends and mentors. They have assisted and mentored me through every step of my life.

Finally, I would like to express my gratitude to my parents for their unwavering support and encouragement over the last ten years, which has helped me get to where I am now. I owe them a lot.

I hope that this thesis is not the end of my academic career. Only by reading and understanding more will I be able to keep my career on track.

Chapter 1

Introduction

The subject matter of this thesis is to automatic anomaly detect computer system log files. The main motivation of this research is to detect anomalous data and find cluster patters from system operating log files. Therefore, the aims and objectives of this research project are related to studying different clustering algorithms and selecting the most effective algorithm to detect anomalous data, which will find potential and reduce the chance of operating system failure and predict the operating system failure.

1.1 Background and Motivation

Software industry has been transformed from delivering products to releasing online services. The term AIOps is to represent Artificial Intelligence for IT Operations (*What is AIOps?*, 2021), that combines big data and machine learning to automate IT operations processes, including event correlation, anomaly detection and causality determination (*Market Guide for AIOps Platforms*, n.d.). Through algorithmic analysis of IT data, AIOps helps IT team to detect digital-service issues earlier and resolve them quickly. Although IT experiences inevitably operational failures in its daily operations, detecting and locating faults is essential to ensure systems' robustness, reliability, and

accessibility. With AIOps, companies can control and manage the quantity of data generated by their IT environments, and thus prevent outages, maintain operating time and attain continuous service assurance. Machine learning is used in AIOps to give IT staff a real-time understanding of any issues. The complexity of data from logs, metrics, event records will locate unforeseen problems to assure continuous service.

System faults prediction is an essential part of many operations and maintenance tasks. However, existing fault handling methods fix faults after they have occurred and are inadequate in proactive detection and rapid prediction. As a result, operating system failures can cause unavoidable damage to applications, related systems and organisations. It also has a highly negative impact and raises a massive challenge to operations and maintenance management.

There are a couple of potential issues to cause operating system failure. The first issue is caused by transmission or network failure. For example,

500 zettabytes of data are uploaded to a dedicated data centre for analysis in traditional Internet of Things (IOT) event monitoring systemAnd in the Internet of Thingsdata is measured in zettabytes, a unit equal to one trillion gigabytes.However, when the transmission device is damaged, or the data centre is disconnected, the data centre will not be able to collect accurate information. While in a remote area, this makes maintenance time continuously extended, and the system cannot accurately predict such failures, leaving the technical team in a reactive state in the face of a series of failures (Nedelkoski, Cardoso & Kao, 2019).

The second potential issue is the server storage fault caused issue. Server storage is a significant frequency of using the device, especially the monitoring of disks. The damage to disks is often hidden but can be devastating to the whole system. With AIOps, the Inspur Company builds a storage intelligence platform (H. Wang & Zhang, 2020). It obtains O&M data from the system such as disk information, failure data, configuration data and other data to analyse and predict these data. Using machine learning algorithms

is to analyse and optimise system failures, performance, configuration and alarm functions, improving maintenance capabilities and responding quickly to hard disk failure problems.

Nowadays, with the continuous development of information technology and the market's growth, systemic network failures occur from time to time, disrupting people's lives and work. However, the current solution is still to receive alerts from the user side and then dispatch professional staff to repair and maintain the situation. In recent years, various organisations and companies have proposed a more intelligent and advanced approach in O&M management to discover and solve potential system failure problems to a certain extent. It also helps to reduce the complexity and repetition of the workload for operations and maintenance workers.

Gartner developed the concept of intelligent operations in 2016 to use big data, machine learning, and other methods to improve operations capabilities, which would further reduce human intervention in automated operations and ultimately make operations human-free fully automatic (?, ?). With the continuous development of artificial intelligence technology, it is hoped that intelligent operation and maintenance can be implemented, and many enterprises are now actively exploring this. Internet companies such as GAVS and Moogsoft have released white papers on AIOps. In the white paper, GAVS identifies algorithms as a competitive tool and identifies some of the critical elements that make up an intelligent Operation system, including critical components such as monitoring ecosystems, analytic, logging systems, automated scripts and data pools. GAVS also sets out its vision for AIOps in the white paper, including increased visibility of operations, information, networks and facilities. It can diagnose real-time problems, provide solutions, real-time notifications of existing issues, and information monitoring and behavioural prediction. Moogsoft suggests that nowadays computing power has become efficient, convenient, and inexpensive. It also indicates nowadays's algorithms, such as supervised/unsupervised learning, which can derive relevant meanings from big

data and assist people in IT operations.

Furthermore, in collaboration with Baidu, Alibaba, Tencent and Huawei, the Efficient Operations Community in China drafted the white paper "Recommendations for Enterprise AIOps Implementation" v.06 (Tianguo, 2021) in April 2018. The white paper explains the goal of AIOps, that is to leverage big data, machine learning and other analytical tools, directly and indirectly, enhance the technical capabilities of IT operations through proactive forecasting, personalisation and dynamic analysis to achieve higher quality, reasonable cost and efficient support for the products or services maintained. The White Paper also suggests that AIOps can be built from nothing to a localised single point of exploration and capability refinement.

Therefore, the thesis is focused on system log anomaly detection in the context of intelligent operations and maintenance (Gaikwad, Deshpande, Vaidya & Bhate, 2021). Anomaly detection is used to identify abnormal behaviour during the operation of a system. Before the O&M platform monitors the IT system in real-time, some indicators are defined in advance. The platform collects monitoring data for each system component according to these indicators during the system operation. There are deviations from historical data in the logs or violations of set rules such as sudden rises, sudden falls, alarms. It indicates a fault in the system, including an attack on the network, damaged hardware device or severe bug in the software. Reading the logs can help IT staff locate the problem. Log anomaly detection is the process of analysing changes in specific parameters in the records to determine potential system problems (Chandola, Banerjee & Kumar, 2009).

1.2 Aims And Objectives

The main aim of the thesis is to study unsupervised learning algorithms, that are clustering algorithms. The clustering algorithms will anomalously detect operation

system data from log files with unlabeled, unintelligible conditions. In view of the main aim of this research, three objectives have been identified.

A log file is an event that took place at a particular time. Logs files generally are a historical record of everything and anything that happens within a system, including events transactions, errors and intrusions. That data can be transmitted in different ways and can be structured, semi-structured, and unstructured. Although logs are designed by different companies or organizations, they contain similar parameter attributes. such as “ERROR","WARNING" and “INFORM", that will be parsed with different content when decomposing logs.

In order to solve the system issues, the first objective in this study is to process the log files into text format. The log files contain incomplete data and uneven structure, so they will be converted error messages or events into text format based on log pre-processing methods. After that, the log format is normalised and de-parameterized to cleanse the data for subsequent log analysis. Then, the input word sequences after TF-IDF processing are used for TF-IDF calculation, and the high weighted terms are screened to build retention. The logs are finally extracted from the LDA topic model based on the valid content of the logs. This approach addresses the interpretation of unlabelled data, so the analysis capability of logs will be enhanced.

The second objective is to study unsupervised learning algorithms, and identify the better clustering algorithm that can perform well in processing such data. Based on unsupervised learning, DBSCAN, Local Outlier Factor (LOF), and K-means algorithms are further studied. The features will be extracted from the log data, and then they are transformed and classified. Numerical experiments are conducted to cluster patterns, and results will be proved the effectiveness of the predetermined cluster algorithm.

The third objective is to preliminary design an AIOps log monitoring system for anomaly detection based on the supported theory. The framework of the log data and the log monitoring system based on anomaly detection will be given afterwards. The design

and schematic of a log monitoring system based on behavioural anomaly detection is based on previously approved research and data collection. After analyzing the AIOps system requirements, the log collecting module, log data storage, anomaly detection, and functional framework diagrams and flowchart tables will be given.

1.3 Organization of Thesis

Chapter 1 introduces the research's background and significance, as well as the research's content and thesis structure. This chapter introduced the vulnerability of an operating system, network, hardware and software systems, which caused serve accidents in many industries. The significant need for anomaly detection with the rapid of information technology is expected. This Chapter also illustrated the problems encountered in current research on anomaly detection of log data, and the importance of research on these problems.

Chapter 2 first reviewed the AIOps model, and the elements of AIOps. Then, the need for anomaly detection in the rapid development of information technology is reviewed. The problems confronting current research on log anomaly detection and the importance of researching to address these problems are highlighted. The current state of log anomaly detection and existing anomaly detection schemes are examined. The definition, functions, and characteristics of logging are explained in the logging overview. The notion of detecting logging anomalies is also defined in this Chapter.

Chapter 3 introduces and explains the relevant tools used in the experiments for this study. The proper programming language will be used at the start of the project. The procurement of gear appropriate for large data processing. In programming, different types of libraries are utilized. Natural language processing algorithms and all of the algorithms will be employed in this experiment.

In the Chapter 4, the researcher first discusses the DBSCAN algorithm's background,

process steps, and history of evolution. After the test log file data was cleaned and sliced into 20 copies, each with 500,000 rows of content, an initial test experiment was carried out to test the clustering algorithm. One of the logs was fed into the DBSCAN algorithm and tested to see how well it could adapt to the data and how sensitive it was.

In the Chapter 5, the background of LOF algorithm, the algorithm process steps and the implementation code are described and explained. In LOF testing, the same processed data when used in DBSCAN clustering algorithm are selected, which ensured a consistent amount of data to be used in experiments. A set of 500,000 and 100,000 rows of data are used to compare the effectiveness of clustering algorithm.

Chapter 6 illustrates the K-means algorithm and tests the method in data analysis. The researcher employed a mixture of TFIDF, LDA and K-means in the K-means method to cluster the retrieved one million rows of valid data. The features were analyzed for correlation to eliminate redundant characteristics. The remaining features were checked for clustering separately, resulting in a valid cluster map that enabled the researcher to identify the outliers in this log.

Chapter 7 was first designed an AIOps system architecture model by researcher. The five functional modules required for log-based anomaly detection functionality were later on summarized. A general framework was designed based on these five functional modules after comparing the three algorithms and reviewing multiple types of literature. By combining commercially accessible tools, the AIOps system model based on anomaly log data analysis was proposed.

Finally, Chapter 8 concludes this thesis with a summary of its original contributions and discuss their significance. Several directions of further research are also suggested.

Chapter 2

Review of AIOps Model and Unsupervised Anomaly Detection Algorithms

2.1 The Review of AIOps Model

AIOps is accomplished sequentially at several levels, with several modules and functions. The elements of AIOps is shown in Figure 2.1 according to (Singh, 2019), First, the I T operation and maintenance data is collected using various underlying tools such as various events, indicators, logs, monitoring, so on, then these collected data streams are passed into the real-time data processing module; finally, the fundamental laws of the data are discovered using rules, pattern recognition, industry algorithms, machine learning, or other artificial intelligence methods, which can be extended to unknown environments.

In the white paper “Recommendations for the Implementation of Enterprise AIOps" (Tianguo, 2021), the AIOps Capability Framework is given. It can be seen in Figure 2.2, in which data was first collected and cleaned for analysing use. In specific, AIOps

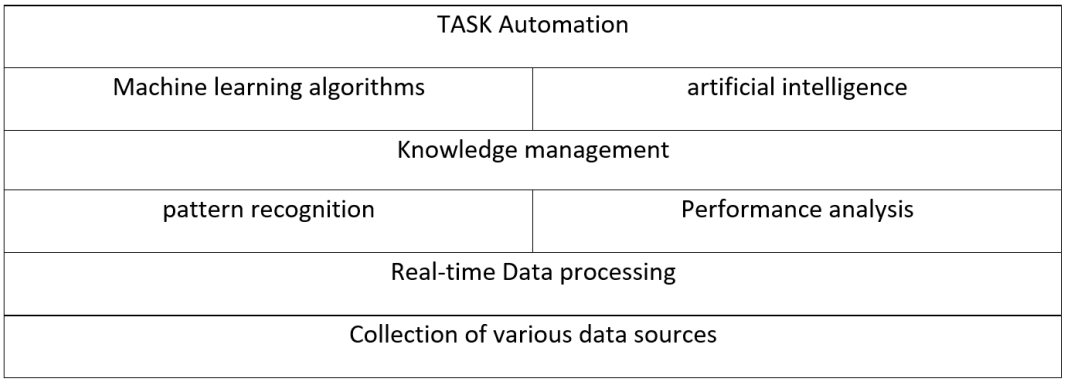


Figure 2.1: The Elements of AIOps

technology has many research points, such as those focusing on efficiency improvement, including intelligent decision making, intelligent change, intelligent questioning. Some are focusing on quality assurance, such as anomaly detection fault analysis, while some focus on cost management, including resource optimisation, capacity planning and performance optimisation. This means that AIOPS technology is a collection of several functional modules that must be combined in order to achieve a holistic and perfectly functioning AIOPS system, beginning with the most basic data collection, cleaning the data, and then analyzing the data using various types of algorithms. According to the AIOPS capability framework, AIOPS is built from nothing to a local single point of exploration, then to a single point of capability improvement, forming an O&M(operation and maintenance) module to solve a local problem, and finally combining multiple single O&M capability points with AI capability into an intelligent O&M process.

intelligent change: Intelligent questioning: The purpose of operation and maintenance is to maintain the stable operation of the system and make the business operation reliable. AIOPS intelligent question and answer uses machine learning and natural language processing to reply to some simple questions, while building a knowledge base of standard questions and answers, so that when meeting the same similar problems, a standard can be given and a unified reply can be given. In this way, the human cost is

solved and the problem is solved quickly.

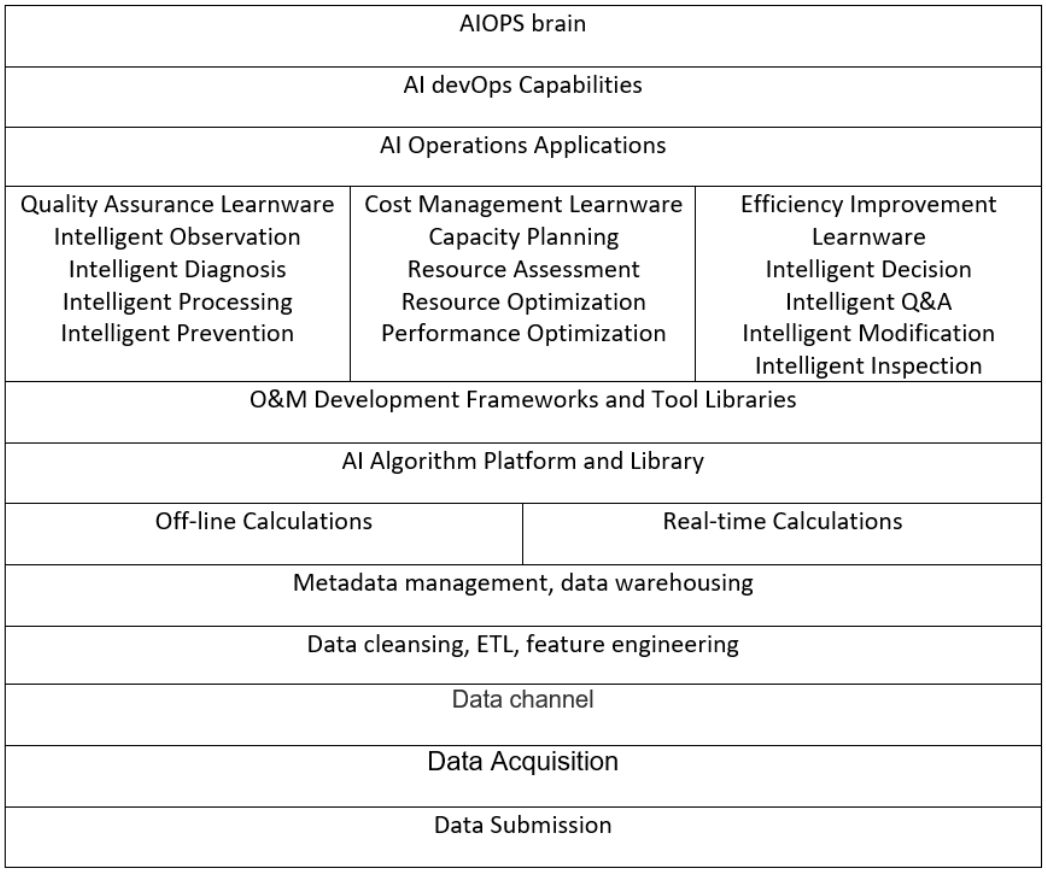


Figure 2.2: AIOps Capability Framework

Furthermore, in the white paper “Recommendations for Enterprise AIOps Implementation" (Bao et al., 2018), a framework for the capabilities of AIOps is given, as shown in Figure 2.3. Within each layer of AIOps, anomaly detection is an essential part of the architecture. Anomaly detection is the process of finding data anomalies that do not match the expected behaviour by comparing the normal behaviour observed in the past with the behaviour of the system or network when an attack compromises it. The anomaly detection process finds data anomalies that do not match the expected behaviour. Anomaly diagnosis is designed to help the user analyze the causes of anomalies for further decision making. Anomaly detection is the prerequisite and foundation of anomaly diagnosis, and anomaly diagnosis is an essential complement to anomaly detection. During the operation of networks, systems and applications, various types of logs are generated to record the status of networks, systems and applications, and important events. These logs contain a wealth of information about the dynamics of network operation. On the other hand, log-based anomaly detection methods are increasingly becoming the mainstream method for detecting anomalous behaviour of networks or systems due to their features, such as accurate analysis of attack problems and reconfigurability of attack chains. On the other hand, the large data volume and heterogeneity of logs pose significant challenges for analysis.

With the gradual upgrading of systems and the complexity of information, more complex errors, vulnerabilities, and attacks will emerge. Therefore, a solution is needed to deal with analysing and mining terabytes of logs quickly. To achieve the goal of system fault prediction, fusing big data, transforming passive fault acceptance into active fault prediction in an intelligent way, realising early warning of faults, and then discovering potential hidden system or equipment vulnerabilities through fault. The analysis of faults reveals potential system or equipment vulnerability problems, thus further improving system reliability and security and providing strong support for intelligent operation and maintenance.

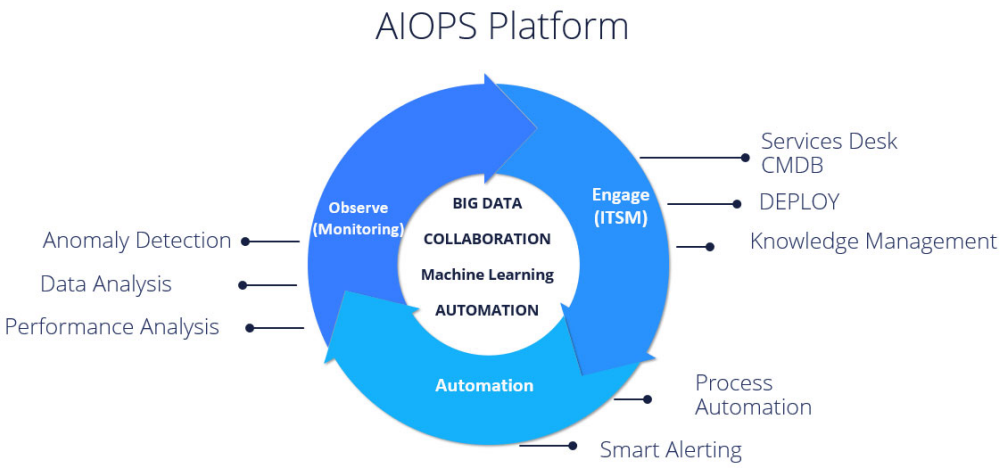


Figure 2.3: AIOps Framework

2.2 The Review of Unsupervised Anomaly Detection Algorithms

Machine learning algorithms are to analyse and cluster unlabelled datasets in general. It can mine large amounts of historical data for implied patterns and use them for prediction or classification. More specifically, machine learning can be thought of as finding a function where the input is sample data and the desired outcome.

These algorithms discover hidden patterns or groupings of data without human intervention. Their ability to find similarities and differences in information makes them ideal for a range of solutions such as exploratory data analysis, category segmentation and image recognition. Unsupervised learning is divided into several different types of algorithms that are implemented to discover similarities and patterns within the data, with a few different types of algorithms commonly used in the following (Goldstein & Uchida, 2016).

1. Statistical Distributions

Defining clusters as objects is most likely to belong to the same distribution. A convenient feature of this approach is that it is very similar to how artificial datasets are generated by sampling random objects from the distribution.

2. Distance-based Clustering

   Clustering is partitioning a dataset into classes or clusters according to a particular criterion, for example, distance criterion. Data objects within the same cluster are basically as similar as possible, while those not in the same cluster are as different as possible. In other words, the data in the same class are clustered together as close as possible after clustering, and the different data are separated as far as possible.

   As a representative of unsupervised clustering algorithms, the K-means algorithm is a distance-based clustering algorithm whose primary function is automatically assigning similar samples to a class. Given a K value and K initial class cluster centroids, each point (i.e., data record) is assigned to the class cluster represented by the centroid of the class cluster closest to it. Following the assignment of all points, the centroids of all class clusters are recalculated (averaged) based on all points within a class cluster. The steps of assigning points and updating class centroids are repeated iteratively until the change in class centroids is minimal, or a specified number of iterations is reached.

3. Density-based Clustering

   Density clustering algorithms generally assume that the closeness of the sample distribution can determine categories. Samples in the same category are closely related, i.e. there must be samples of the same category not far from any sample in that category. By grouping all the closely related samples into different categories,

the final result of all the clustering categories is obtained. DBSCAN (Density-based Clustering of Application with Noise) is a popular density-based clustering algorithm. It is a classical density-based clustering method, but it requires the user to set MinPts and Eps parameters without prior knowledge. It significantly impacts the quality of the final clustering results and leads to worse clustering results in datasets with uneven density due to global parameters.

Moreover, another of the more typical strategies for density-based outlier detection is the Local Outlier Factor (LOF) detection algorithm. It computes the local density deviation of a data point in relation to its neighbours. Outliers are samples having a density substantially lower than that of their neighbours. For each point in the dataset, the method computes an outlier LOF and assesses whether it is an outlier by calculating the LOF near 1. If the LOF is significantly more than one, it is termed an outlier, and if it is near to one, it is considered a normal point.

4. Clustering-based Methods

   Clustering-based algorithms are used to classify raw and unclassified data objects by processing them into groups represented by structures or patterns in the information. The data probably are clustered in several different places to form clusters. The closer a point is to a large cluster, the higher the probability of a standard point, and vice versa.

5. Tree-based Algorithm

   The algorithm is built around the theory of decision trees and random forests. The algorithm divides the data into two parts based on an arbitrary threshold when processing the data. This process continues recursively until each data point is isolated. Once the algorithm has traversed the entire data, it filters out fewer data points than the other steps to be isolated.

6. Other Algorithms

   One-Class SVM and RNN as an example are to cluster data. One-Class-SVM finds a hyperplane to circle the positive examples in the sample, that is used to make decisions, and the circled samples are considered positive. Kernel functions are time-consuming to compute, so they are not used much in massive data scenarios. The RNN algorithm, a widely used self-coding algorithm in deep learning, can be applied to a single classification.

The Figure 2.4 shows the listed unsupervised anomaly detection algorithms. It used clustering analysis to analyze human behaviour. And it is through the continual development of subconscious clustering patterns. Individuals learn how to differentiate between dogs and cats, animals and plants. It is currently widely investigated and applied in various disciplines, including pattern recognition, data analysis, image processing, market research, customer segmentation, web content categorization.

Unsupervised anomaly Detection Algorithm's					
Distribution Based	Distance Based	Density based	Clustering Based	Tree Based	Classifier Based/Other
HBOS	KNN	LOF	CBLOF	IHCUDT	One-classSVM
GOE-GMM	KNNW	COF	LDCOF	iForest	RNN
Dixon	ODIN	DBSCAN	CMGOS	SciForest	

Figure 2.4: Unsupervised Anomaly Detection Algorithms

2.3 The Review of Logs Related Concepts

Throughout the software system, logs exist to document the system's present status and reflect the system developer's record of notable or uncommon occurrences. It

demonstrates that systems encounter a succession of non-fatal exceptions and faults before failing or crashing, which may be logged in real-time in the log stream as events occur. Therefore, monitoring and analyzing log streams is an excellent method to identify aberrant system behaviour and provide early alerts of system problems, spare system administrators vital time to take appropriate action.

2.3.1 The Definition of Log

A log is a record that describes the behaviour of a system, usually a chronologically ordered collection of specific actions and the results of those actions, generated by specific objects such as application software, operating systems, server devices, network devices, and security devices (*What is a Log File?*, 2021).

Logs are typically stored on storage devices as log files, either directly readable text files or machine-readable binary files. Each log file consists of a single line of log records, one or several consecutive records describing a single event. An event happens in a particular environment, usually involving changing its state. Events usually include the time of occurrence, the content of the event, and any details relating to the time or environment that may help explain the cause and effect of the event. A log is a collection of event records.

Although log records come in a variety of formats, a log record usually contains a timestamp of when the event occurred, the specific content of the log, the type of log, the log level and, in some cases, the node ID, source address, destination address, process ID. The content can be divided into three components: Subject, Object and Action (Winding, Wright & Chapple, 2006).

With a log representing an action performed by a subject on an object and the result of that action, the subject represents the user or action initiated by the user in some intrusion detection models. The object represents various resources such as hardware

and software facilities or the network environment. The action represents a system service or an application behaviour of a client software system.

2.3.2 Features of Logging

The development of computer logging systems has resulted in various types, formats and volumes, and several mature, configurable, commercial or open-source logging components, such as Log4j, Sl4j, Apache Commons Logging, Logback and many others. The typical features of logging systems can be summarised below (Chuvakin, Schmidt & Phillips, 2012).

1. The diversity of logging shows how logs are transmitted, the syntax and format of logs, and how logs are stored. Firstly, there are many different transport protocols for logs, such as Syslog, SOAP over HTTP, SNMP, WS-Management, many proprietary product-specific log transport protocols, and local storage methods. Secondly, there are different syntaxes and formats for logs, with well-known log formats such as W3C Extended Log File Format, Apache Access Log, Cisco SDEE/CIDEE, ArcSight Public Event Format, Syslog, IDMEF and so on. However, many logs still do not follow any common format and exist as free-form text, making log analysis difficult. This makes log analysis difficult. Finally, logs are stored in either human-readable text files or machine-readable binary formats.

2. In the past, development managers analysed logs manually using simple text processing tools, such as searching and filtering. This was mainly due to the small size of the system, the small range of users, the lack of complexity in processing events and the fact that the size of the logs was still within manual control. Today's big websites and the business systems behind them are of unprecedented scale and complexity, generating logs of terabytes in size every day, making storage

and analysis very difficult. It is even more impossible to detect system problems by manually analysing the logs one by one.

3. The lack of a uniform logging standard results in logs is not structured in a fixed format and are sometimes inconsistent, incomplete or misleading. Many log files are stored in binary form and can only be opened by special software, making reading the logs difficult. Many logs contain only crucial data, which can only be understood with the help of appropriate documentation. Good logging should include what happened when it happened, where it happened, the source and the target.

4. Although logging is an essential part of the software system and provides a critical record of its status, it has a low priority. It can be inconsistent and incomplete when the system is under high load or exceptions. Log files are also generally not encrypted or otherwise protected against tampering, and logging can be turned off by modifying configurations and killing processes.

2.3.3 Anomaly Detection in Logs

Logs can be considered a reference for measuring system performance. It can be used to analyse any anomalies in a particular system area. Traditionally, log analysis has been done empirically by setting thresholds (Linders, 2021). With AIOps, the system is expected to intelligently analyse the logs for anomalies. While there are different approaches to log anomaly analysis, this thesis will focus on unsupervised learning-based log anomaly detection.

Log analysis is the process of examining, interpreting, and understanding computer performance and recording it, which includes logging network devices, operating systems and applications. A log is a series of messages arranged in chronological order that describes the activities running inside a computer. As distributed and cloud

computing continues to evolve, large-scale systems consisting of hundreds of software components run centrally on thousands of computing nodes. Their runtime data is continuously collected and stored in log files, which are often used to analyse the causes of system failures and to locate system faults. In large asynchronous and concurrent systems, logs are of great help because they are easy to sample due to their large sample size for testing purposes.

Existing methods of log anomaly detection usually categorise logs based on the characteristics of the log data itself or the client system that generated the log files. When a single log is labelled as either a regular log or an error log, the data is learned using supervised or unsupervised learning machine learning methods to derive a classification model. Yamanish *et al.* (Yamanishi & Maruyama, 2005) analysed web logs to detect network errors using a hidden Markov model and an online discounting algorithm. In (Sahoo et al., 2003), it constructed an active prediction and control system using a time-series algorithm. A rule-based classification algorithm and a Bayesian network model are to investigate the use of stochastic mapping and SVM (support vector machine) to classify log data (Fronza, Sillitti, Succi, Terho & Vlasenko, 2013).

However, the size of most log files is so large that manually tagging them one by one is highly time-consuming and impossible. Therefore, more practical anomaly detection techniques are based on unsupervised learning or other statistical learning methods. For example, Lim *et al.* (Lim, Singh & Yajnik, 2008) clustered logs by the similarity of their payloads and then used frequency features to predict the system state. The frequency of different log classes generated in a fixed time window is used. In (He, Zhu, He & Lyu, 2016), three different types of session windows (fixed, sliding and session) to slice and display log features are applied and analyzed. Xu *et al.* (W. Xu, Huang, Fox, Patterson & Jordan, 2009) obtained log patterns through source code analysis and used a principal component analysis-based algorithm to flag outliers as system anomalies, which yielded promising results. Moreover, the drawback was that the source code of

the client program had to be obtained.

Meanwhile, Bao *et al.* (Bao et al., 2018) summarises the following types of commonly used methods in 2018, which includes classification-based methods, statistical-based methods, rule-based, graph model-based and machine learning-based methods.

1. Classification-based log anomaly detection methods have two main phases. The first phase is the training phase, which learns a classifier from a collection of useful logs with normal or abnormal labels. The second phase is the testing phase, which uses the classifier to split the collection of logs into two groups of abnormal and normal logs.

2. The statistical-based approach is to anomaly detect logs that devise a statistical model that matches the system's expected behaviour. If the statistical model does not reach certain set thresholds, then there is a high probability that the data is abnormal.

3. The rule-based processing method proposed by (W. Xu, 2010), which suffers from the problem that some rules are challenging to maintain and improve. Another direction of improvement in anomaly detection is full-text search-based tools, such as the commercial analytical tool "Splunk". While offering significant performance and stability advantages, the operator still requires keywords for retrieval. However, these keywords often need to be defined manually. If the keywords are often determined beyond the operator's control or iterative relationships, this may result in incorrect information or unavailable queries. Since log messages often indicate the source of logs, log analysis is formalised as an anomaly detection problem in machine learning.

4. The graph model-based log anomaly detection method constructs a finite automatic state machine that represents normal behaviour, and then matches log data

with finite automatic state machine. If this state machine fails to match some log data, then these log data are likely to be anomalous log data. On the other hand, it applies that graph model-based advantages are threefold in (Bao et al., 2018), which are diagnosing sequence problems buried deep within the logs, providing the researcher or IT engineer with a correlation of the problem data, and accurately provide the correct log sequence and clarify the event occurrence problem.

An unstructured log analysis technique for anomaly detection was proposed by Fu *et al.*in (Fu, Lou, Wang & Li, 2009), which reduces the level of application-specific knowledge dependency and converts free-form text messages in log files into the form of log keys. In (He et al., 2016), it summarises systematic detection methods using principal component analysis and provides a detailed review and evaluation of six state-of-the-art log-based anomaly detection methods. Although these machine learning methods are efficient, they perform less in anomaly detection when the data changes, as traditional machine learning for anomaly log detection only outputs simple predictions. Liu *et al.* (Liu & Wang, 2011) mined log data based on association rules, using the a prior algorithm to clean and remove complex data from the network to find hidden information relationships between the data. We are not going into too much detail about the algorithm, although it is a valuable illustration of the preliminaries of data mining. On the other hand, it helps researchers find good hidden relationships in the data.

A survey of information about anomaly detection in industrial networks (IS) in large systems was presented in (Iturbe, Garitano, Zurutuza & Uribeetxeberria, 2017). It outlines the main problems in large systems such as botnets, spam, DDOS, and other network problems. It further describes system security problems, using ADS to create monitoring in IS. The created classification method is to help the system to distinguish between different levels of attacks through unsupervised or semi-supervised detection

methods to detect outliers. It helps us understand the approach to big data detection in large systems. It also suggests future research directions geared towards the volume of data to be processed, the speed of processing and the diversity of errors to be addressed.

2.4 Summary

This chapter focuses on the AIOps model and the anomaly detection algorithms. It begins with an introduction to AIOps, followed by information on why AIOps is required for enterprise extensive data operations. The critical role of AIOps plays in processes and information about the model structures proposed by Ganter and others. Then, clustering methods are investigated and reviewed. Unsupervised and supervised learning are the two types of standard clustering algorithms. However, unsupervised techniques are more adaptive due to the data's particular, so that many types of algorithms and their properties are given in relation to the data attributes.

Chapter 3

Research Related Methodology

This research analyses the log data and aims to propose a suitable method for anomaly log detection. In Chapter 2, the researcher examined the kind of anomaly handled in log files and unlabelled data processing methods. Different deep learning methods are investigated and studied, while supervised and unsupervised learning methods are analysed and summarised. A textual topic extraction approach is used to label the anomalous data. The choice of algorithms is more focused on which algorithms help deliver the best accurate results for that type of log file.

In the experiment, we followed a similar approach from the literature and combined it with other approaches, as the manipulation of variables involved was a necessary factor in obtaining the results of this experiment. As the algorithm aims to anomaly detection in log files, there are different steps in dealing with this type of problem. The original log data is not fully structured and labelled, so it needs to be pre-processed with transformations, dimensionality reduction and labelling before parsing. The processed data is then substituted into the algorithm, and unsupervised learning is then applied in this study, which will be discussed in the following.

Keyword extraction methods will be further used in this study. It can be divided into two main categories: supervised keyword extraction and unsupervised keyword

extraction. The typical supervised keyword extraction method treats it as a binary classification problem, using more existing information and making it relatively effective. However, it requires high-quality training data annotated in advance and is expensive to pre-process manually. In general, keyword extraction based on TF-IDF statistical features is simple and easy. Therefore, the current keyword extraction research focuses on unsupervised keyword extraction based on TF-IDF statistical features.

3.1 Preliminary Research Preparation

3.1.1 Log Data Pre-processing

In general, log analysis techniques include two main sections, which are pre-processing of log data and detection of behavioural anomalies. The pre-processing technique consists of log regularisation and log information clustering. The first step is to unify the log format through log transformation coding and clean the logs. The number of log types with different contents is significantly reduced by removing parameters. The traditional clustering of logs is to find outliers in the clustering results and detect anomalies accordingly, while this study will explore the behavioural patterns of different types of logs. The proposed clustering similarity definition and pruning strategy improve the clustering accuracy. Data mining and analysis, in general, can be divided into four steps: problem definition, data pre-processing, feature extraction and cluster analysis.

Before data analysis and evaluation of results, the data is first subjected to pre-processing operations. It is mainly because real-world data is susceptible to noise, data loss, and redundancy, leading to low-quality data mining and analysis results. Data quality includes many factors such as accuracy, completeness, data consistency, timeliness, credibility and interpretability. It can be influenced by the following factors.

1. The system log data are written by the developers. It was intended to be read

and queried by humans to diagnose functional errors and analyse the causes of mistakes after they have occurred or crashed during system operation. Pre-processing is also necessary to automate the analysis and mining log data. The need for log data pre-processing is reflected in the fact that the logs may be missing data. When the monitored customer system is under high load or in the event of an error, log information may be lost or incomplete. It is inevitable, and the logging system is usually a lower priority than other functional modules running in the design and is often the first part to be discarded to ensure system performance and stability. However, missing logs do not occur often. Studies have shown that a small amount of missing data has little impact on the final analysis, so ignoring this data is usually a good idea.

2. The structure of logs is not fixed. Logs do not follow a common standard across platforms and exist in a consistent form. The most important part of the log content, the payload, is still written manually using natural semantics, and the programmer determines the format and content. It is an important issue for this experiment.

3. The volume of logs is vast. Today's logs have been scaled up to terabytes in size, and traditional log analysis systems based on a semantic line-by-line review of log content can no longer handle such a large stream of logs. When faced with a situation where log content cannot be interpreted manually and data traffic is high, researchers propose to analyse the pattern of log data generated from a different perspective and use it as a basis for anomaly detection. The researcher needs to clean the logs, identify the patterns in the logs, clarify the composition of the logs and assign each log to a specific type.

3.1.2 Log Data Clustering

If each log record in the dataset represents a type, the researcher's detection algorithm would be deal with a feature space containing millions of feature variables. Furthermore, as each record represents a unique type, it would be difficult for the researcher to obtain any log features. A type may only occur once or twice over a while, making it impossible to draw any patterns. Only a sufficient sample of data exists to discover the behavioural characteristics hidden behind a particular log type. This is the purpose of log clustering by researchers.

Clustering is typically an unsupervised learning process that divides a collection of objects into groups or clusters based on their similarity, with objects in each group being similar to one another and objects in different groups being less similar. Typical clustering steps include feature selection, feature extraction, similarity selection, grouping, and evaluation of clustering results. First of all, it is the feature selection. The data is quantified and downscaled, and the most practical features are selected. Data transformation of the features chosen forms new salient features, which is the feature extraction. The result can be represented as a matrix where the rows represent the sample and the feature variables. The similarity criteria are selected to find the most appropriate distance function for the feature type or construct a new distance function. A clustering algorithm is executed to group the data. The input to the algorithm is a sample matrix, and the output can be a tree diagram or a specific classification scheme, reflecting the classification at different levels of granularity. The final clustering results are obtained by drawing on empirical or domain knowledge to define classification thresholds and evaluate the validity of the clusters.

Usually, different datasets have different types, structures and representations. No universal clustering algorithm can cluster all datasets and achieve good classification results. The most appropriate clustering method should be chosen for the characteristics

of each dataset. There are different datasets sizes. Some are small datasets with a few hundred objects. Some are large datasets with millions or hundreds of millions of objects. Handling different attribute types will be a challenging job, which include numeric, positive/negative, categorical, text and other data types. A group or a cluster can be any shape, and a good clustering algorithm can discover clusters of any shape.

Moreover, most datasets may contain noisy data, which can significantly impact the clustering results, so the algorithm's sensitivity to noise needs to be reduced. Some algorithms are only good at handling low-dimensional data, whereas high-dimensional data contains detailed features and algorithms should improve their ability to cluster high-dimensional data. The output of clustering should be interpretable and usable by the end-user. Understanding the concept of clustering and analysing the characteristics of several major clustering algorithms are the most critical work. Combined with the type, structure, and format of the log data, the researcher decided to cluster the logs using several different methods and explore which of the three clusters would be more appropriate for the researcher's dataset.

Unlike numerical data, most software system logs are made up of plain text. Logs have no concept of dimensionality. Unlike the common practice of clustering, for example, p-dimensional multivariate data matrices, most log types do not have a concept of dimensionality. Most of the information in a log is concentrated in the payload, and it is challenging to extract multidimensional features. The number of log types remains unknown. It is often unknown what types the logs contain, and it is equally difficult to determine the log type data by expert or knowledge domain.

3.2 Experimental Environment

The experimental equipment is shown in Table 3.1. For better efficiency and accuracy, the researcher purchased a new Lenovo Y7000 laptop with an i7-10750H CPU, 16GB

of RAM, NVIDIA RTX2060 graphics card, and a 1TB storage hard drive for all experiments and data processing will be deployed.

Operation systems	Windows 10
Processor	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
RAM	16GB
Core	6
Hard Drive	Samsung MZVLB1T0HBLR-000L2 1TB
Graphic card	NVIDIA GeForce RTX 2060

Table 3.1: Experimental Equipment Table

In this experiment, python is used as the machine learning language of choice. For the main reason, its syntax is simple. Developers can rely on the various algorithms and modules in the python library to build prototypes, allowing users to focus more on machine learning than on language differences. The following Python Library are also used.

NumPy is a Python library used for working with arrays. It also has functions for working in linear algebra, fourier transform, and matrices. NumPy (Numeric Python) provides advanced numerical programming tools, such as matrix data types, vector manipulation, and sophisticated arithmetic libraries. It is designed for rigorous numerical manipulation.

Pandas is a robust set of tools for analysing structured data based on Numpy (which provides high-performance matrix operations). Pandas can import data from various file formats such as CSV, JSON, SQL, Microsoft Excel. Pandas provide support for multi-dimensional arrays.

Matplotlib is a graphing library for Python. It can be used with NumPy and can also be used with the Graphics Toolkit. Pyplot is a collection of functions that make Matplotlib work like MATLAB. Each Pyplot function makes some change to a figure. For example, it creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels.

Seaborn is a Python data visualisation library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn is a higher-level API wrapper around Matplotlib, making it easier to draw graphics in most cases, while using Matplotlib makes it possible to produce more distinctive graphics. Seaborn should be seen as a complement to Matplotlib rather than a replacement. It is also highly compatible with NumPy and pandas data structures and statistical models such as Scipy and statsmodels.

Sklearn is a powerful machine learning library provided by a third party in Python that covers everything from data pre-processing to training models. Using scikit-learn in practice saves the researcher much time writing code and reduces the amount of code the researcher has to write, giving the researcher more time to analyse the data distribution, tune the model, and modify the super parameters.

The testing log data for this experiment is a collected from a private company firewall log data. which are no longer to use and do not need an ethic approval. It contains all events logged from 24/08/2014 to 16/09/2014 for 23 days. The sample logs show in the following Figure 3.1.

Figure 3.1: Experimental Testing Log Data

The testing log dataset size is 5.41GB, with a total of 10201343 rows of logged data. Although there are different structural formats for log messages, the content of

the fields is similar except for the core part. Most log messages consist of between 3 and 10 fields, including timestamp, payload, log level, node ID, process ID, source address, destination address, error code. Spaces, commas or other symbols separate these fields, and line generally breaks separate lines. Some normalisation methods have been discussed in the previous section, such as removing redundant characters, identifying log boundaries, and standardising timestamp formats. In order to make logs easier to automate, researchers have summarised the existing research and proposed their conversion methods.

3.3 Log Analysis Methodology

3.3.1 TF-IDF Methodology

Term frequency (TF) is the frequency with which a word appears in a document. There is a positive correlation between the term count and the importance of a word, which tends to occur more frequently in the text. However, whether or not a word is essential, it is likely to have a higher word count in a longer document than in a shorter one. TF is a normalisation of word count to prevent it from being biased towards longer documents. Therefore, TF is the total number of times a word appears in a log file. However, considering the length of each log file is different, the researcher could set a criterion for TF, which is the total number of occurrences of a word in the log divided total number of words. The formula for calculating this is in the following:

$$\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{3.1}$$

Where tf_{ij} represents the frequency of feature item n_{ij} in document k. $\sum_k n_{kj}$ is the total number of words for the document k.

It relates to the first time the notion of IDF was established in (Robertson, 2004).

Salton eventually altered “Frequency" to “Inverse Document Frequency," and IDF (Inverse Document Frequency) has been used ever since. When a word appears in a large number of documents, its corresponding IDF value should be lower. However, when a word appears in a small number of documents, its corresponding IDF value should be more significant, as expressed by Equation 3.2. The mathematical equation for IDF is log(total number of documents in the corpus / (number of documents containing the term + 1).

$$\mathrm{IDF} = \mathrm{lb}N - \mathrm{lb}(n+1) \tag{3.2}$$

On the other hand, Shannon’s information theory defines the IDF as the cross-entropy of a keyword’s probability distribution under particular conditions. If a feature appears in the majority of papers, it has less information entropy. If it appears in only a few documents, it has greater information entropy. Once the TF and IDF are understood, the TF-IDF may be calculated as “TF-IDF=TF*IDF".

Stilton proposed the TF-IDF (Term Frequency & Inverse Documentation Frequency) technique. The basic idea is that if a word appears more frequently in a specific document, the larger the TF value, the better it is at expressing the content of that document and should be given more weight. If a word appears less frequently in a group of documents, the smaller the calculated IDF value, the better it is to distinguish the document’s content and be given more weight. If a term appears in a lower number of documents, the smaller the estimated IDF value, the better it is in distinguishing its content and should be given more weight (D. D. Xu & Wu, 2014). The mathematical equation for TFIDF is in the following:

$$\mathrm{w_{ij}} = \mathrm{tf_{ij}} \times \mathrm{idf_j} = \mathrm{n_{ij}} \times \log \frac{\mathrm{N}}{\mathrm{n_j}} \tag{3.3}$$

Where n_{ij} is the frequency of occurrence of feature item W_j in document d_i, N

denotes the total number of documents, and n_j denotes the number of documents containing feature item w_j. The base of the function log can be taken as 10, 2 or the natural logarithm of e.

The log text was formatted using sublime text and the output format was utf-8 as the standard format. To facilitate the measurement of similarity in log clustering, the researcher has assigned the following “0", “1", “2" and “3" to each level by unique hot coding. For example, Virusname is equal to 1, and non-Virusname is equal to 0. In (Vaarandi, 2003), a count of words appearing in the log text shows that most words appear only once or twice in the log sample, while a small number of terms that appear very frequently in the document sample are associations between these frequent words. This is mainly because the developers formatted the output using parameter templates when writing the log content.

These parameters may be formed by IP addresses, process IDs, simple numbers or memory addresses. Log analysis systems can misclassify the same type of log as a different type because of the other parameter values in the log message. To solve this problem, the researchers used a de-parameterisation approach, replacing the numeric parameters that appear in each log with keyword placeholders. For example, replacing the value of the IP address with the placeholder “IP" and the value of the process ID with “processid". The researcher will take out multiple parameters and convert different types of log data into the same text. The feature extraction program can be illustrated in the following Figure 3.2. After vectorisation by TF-IDF, the researcher extracted the word cloud from the text part of the 20 cut logs using the “wordcloud" to present the content.

The internal log data can be divided into data content and other content. The reason for dividing the data into two separate topics is that in this log, data content can be considered as the practical load part of the data, which is the core information area within the log, recording the adequate information of the system behaviour. Other

```
print("   start extracting content...")
# extract the "content" from the raw log and clean it.
df0['content'] = df0['str'].astype(str).apply(lambda x: find_content(x,'content=".*"')) # extract content from the log text
df0['content'] = df0['content'].fillna(" ")
df0['content'] = df0['content'].apply(lambda x: re.sub('content',' ',x)) # keep the content of the index
df0['content'] = df0['content'].apply(lambda x: re.sub("[¯!@#$%^&*()_\-+=○?:[]|,.\/:'\\[\] · ¯! @#¥%……&* () ——\-+=[]|《》?： "" 【】、;
df0['content'] = df0['content'].apply(lambda x: re.sub("",' ',x)) # remove the punctuations
df0['content'] = df0['content'].apply(lambda x: " ".join([" if s0 in STOPWORDS else s0 for s0 in x.split()])) # remove stop words
df0['content'] = df0['content'].apply(lambda x: re.sub(' +',' ',x)) # replace the multiple space to signle space for tokenise

print("   start extracting other_info...")
# extract the info other than "content" from the raw log and clean it.
df0['other_info'] = df0['str'].apply(lambda x: re.sub(' ([0-9]+\.){3}[0-9]+(:[0-9]+|)','',x)) # remove ip
df0['other_info'] = df0['other_info'].apply(lambda x: re.sub(' [0-9]+[\S]+([0-9]+)[ ]','',x)) # remove machine address
df0['other_info'] = df0['other_info'].apply(lambda x: re.sub('content=".*",' ',x)) # remove content
df0['other_info'] = df0.apply(lambda x: re.sub('virusname="[\w\d]*",' ',x['other_info']), axis=1) # virusname
df0['other_info'] = df0['other_info'].apply(lambda x: re.sub("[¯!@#$%^&*()_\-+=○?::[]|,.\/:'\\[\] · ¯! @#¥%……&* () ——\-+=[]|《》?： "" 【
df0['other_info'] = df0['other_info'].apply(lambda x: re.sub("",' ',x)) # remove the punctuations
df0['other_info'] = df0['other_info'].apply(lambda x: " ".join([" if s0 in STOPWORDS else s0 for s0 in x.split()])) # remove stopwords
df0['other_info'] = df0['other_info'].apply(lambda x: re.sub(' [0-9]+ +',' ',x)) # remove pure numbers
df0['other_info'] = df0['other_info'].apply(lambda x: re.sub(' +',' ',x)) # replace the multiple space to signle space for tokenise
```

Figure 3.2: Feature Extraction Program

Figure 3.3: 5th/20 Log “Wordcloud"

content is used as a categorical representation for extracting other parts. As we are dealing with natural language, the main deletions in the denoising process include: singular and plural unification, merging of synonyms, use of characters, full names and abbreviations, proprietary phrases and so on. Words specific to this log and meaningless words such as “GenTime", “SMAC", “FwPolicyID", “Protocol" are also removed. “Protocol" and other words called “Stop words" are normalised to ensure that the results are as objective and scientific as possible. The listed “Stop words" can be seen in the following Table 3.2.

3.3.2 Feature Extraction and Processing

The TF-IDF is usually used to measure the importance of a word or phrase in a text set to the text containing the word or phrase. IDF is the product of TF and IDF, and since

GenTime	SerialNum	SrcIP
ProtocolTYPE	SrcPort	DstPort
DMAC	FwPolicyID	EventName
SecurityType	SecurityID	Action
TCP	policy	file
DstIP	Protocol	ProtocolID
InInterface	OutInterface	SMAC
EventID	EventLevel	EventsetName
VirusFileName	Content	EvtCount
xge1	zip	

Table 3.2: Listed “Stop Words"

TF-IDF has been described in detail in the method chapter, we will not repeat it here. The researcher randomly selected the features within log data No. 0 and counted a total of 54 features in Figure 3.4, which were vectorized by TF-IDF and roughly categorized as numeric features, text features and categorized features.

An example can be seen in Gentime. In general, Gentime in Numberic Feature converts Datetime to timestamp. The text features can be processed in the following. First, information is extracted from “content" and “other-content" other than content as mentioned above. Using TF-IDF vectorisation, and The Latent Dirichlet Allocation (LDA) topic model was then used to extract topics from all text-like data. After iterative debugging, three topics were extracted from both features, resulting in six numeric features: “other-info-topic-1", “other-info-topic-2", “other-info-topic-3", “content-topic-1", “contenttopic2", “content-topic-3".

3.3.3 Latent Dirichlet Allocation (LDA)

Blei *et al.*introduced LDA model in 2003, which gives the topic of each document as a probability distribution. By analysing some documents and extracting their topics (distribution), it is possible to perform topic clustering or text classification based on the topics (distribution). At the same time, it is a typical bag-of-words model where a

Numeric Features	Text Features	Categorize Feature
Gentime 'timeStamp'	**Content** '1', '2', '3 **other_info** '1', '2', '3	**head_type** ' alarm', ' antiarp', ' attack', ' av', ' command', ' config', ' filter', ' ha', ' if', ' ips', ' keepalived',' no', ' ospf', ' pldapp', ' pldrun', ' scan', ' sensor', ' sys',' system', ' utm', ' warning', ' xapp', **event_level** '0', '2', '3 **event_id** 152323843','152325093', '152329342', '152330019', '152519066', '152519913', '152520200', '152520540', '152520629', '152520782', '152520785', '152520822', '152521921', '152521938', '152521969', '152522098', '152525939', '152526080', '152526081', '152543971', 'unknown_eventid', **virus name**
Metric Data Features		
'other_info_topic_1' 'other_info_topic_2' 'other_info_topic_3' 'Content_Topic_1' 'Content_Topic_2' 'Content_Topic_3'		

Figure 3.4: 54 Features Extracted from Testing Log File

document is composed of a set of words with no sequential relationship between words. In addition, a document can contain multiple topics, and each word in the document is generated by one of these topics (Kulshrestha, 2020).

Due to the need to pre-mark high-quality training data, the existing research on keyword extraction mainly focuses on unsupervised keyword extraction. The main approaches can be categorised into three types: TF-IDF statistical feature-based keyword extraction, topic-based keyword extraction, and word graph model-based keyword extraction. This study proposes a method that incorporates the TF-IDF statistical features into the keyword extraction process (Blei, Ng & Jordan, 2003).

The LDA topic model can mine the topics of documents. In this study, we use the LDA topic model to mine the topics of the Chinese short text database as a whole, and the distribution of the topic words are all topic-related words, which have a high degree of differentiation. Therefore, using the results of LDA topic mining to supplement the features of the retained dictionary can achieve better classification results (Blei et al., 2003).

The LDA model structure can be modeled in the Figure 3.5. θ_m denotes the topic distribution of document m, a denotes the prior distribution of θ_m, z_m, n denotes the topic of the nth word of document m sampled from θ_m, $\phi z_m, n$ denotes the word distribution, β denotes the prior distribution of the word distribution, w_m, n denotes the nth word of the final $m_t h$ article generated, and N_m denotes the total number of words in document m. There are total M documents in the LDA model.

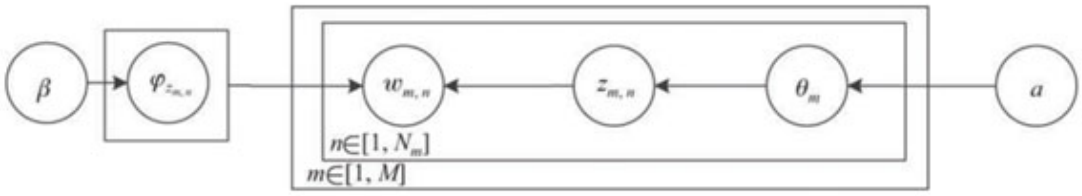


Figure 3.5: LDA Model Structure

The joint distribution of all variables in the LDA is calculated as in the following equations.

$$p(w_m, z_m \mid a, \beta) = \sum_{n=1}^{N_m} p(w_{m,n} \mid z_{m,n}, \beta)\, p(z_{m,n} \mid a) \tag{3.4}$$

$p(w_m, n|z_{m,n})$ is the probability of sampling words under the topic. The probability distribution of each word in the $m_t h$ document is given by the following Equation 3.5.

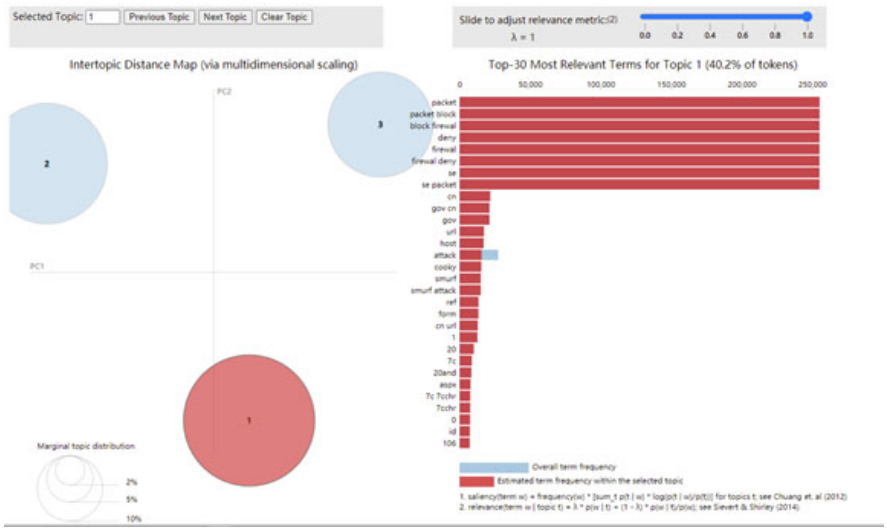
$$p(w_{m,n}) = \sum_{n=1}^{N} p(w_{m,n} \mid z_{m,n})\, p(z_{m,n}) \tag{3.5}$$

The LDA model uses the dirichlet distribution as a prior knowledge of the topic distribution information, which provides a good picture of the document generation process. As a result, the researcher obtained two LDA topic statistics, one for “LDA_Content_0", the other for “LDA_Other_Conetnt_0".

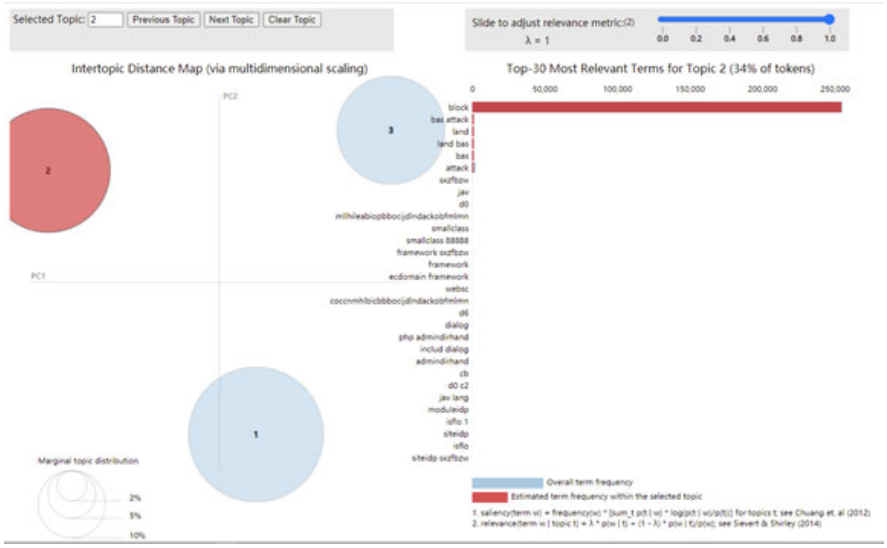
In “log_0_log" data, the six topic model obtained by LDA indicates how often these words appear in this log, and to some extent, represents the content of this topic. For example, in “content_topic_1,2,3", the words “packet", “block", “firewall" and so on can infer that this part of the log records more information about the firewall blocking records, while in “Other_content_topic 1,2,3", the words “block 201408211def2", “201408211def2 1", “deny" are more prevalent, which also indicates that this log records more “block" information. Records and matches the “content_topic" content more closely.

However, some of the words counted by the LDA could not be interpreted manually, especially the numeric type of data. However, to verify the initial observations and assumptions, the researcher brought all 60 feature values into the algorithm for grouping and visualised the results using TSNE dimensionality reduction to observe the grouping effect. “T-SNE" (t-distributed stochastic neighbour embedding) is a machine learning algorithm for dimensionality reduction proposed by Laurens van der Maaten *et al.*in 08. In addition, “T-SNE" is a non-linear dimensionality reduction algorithm suitable for visualising high-dimensional data down to 2 or 3 dimensions. The algorithm allows the distance of the t-distribution in the low-dimensional space to be slightly smaller for points with greater similarity, while for points with lower similarity, the distance of the t-distribution in the low-dimensional space needs to be further (Zhu, Webb, Mao & Romagnoli, 2019) (Pezzotti et al., 2016). The researchers invoked the “T-SNE" algorithm encapsulated within the Sklearn library to bring 60 features into 500,000 rows of data.

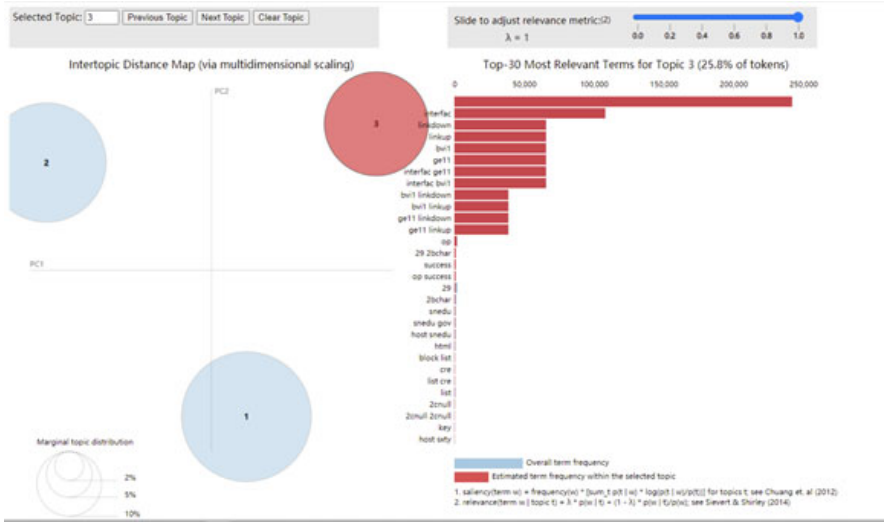
The above diagram shows the researcher labelling each row of the array of 54



(a) "Content_0_Topic 1"

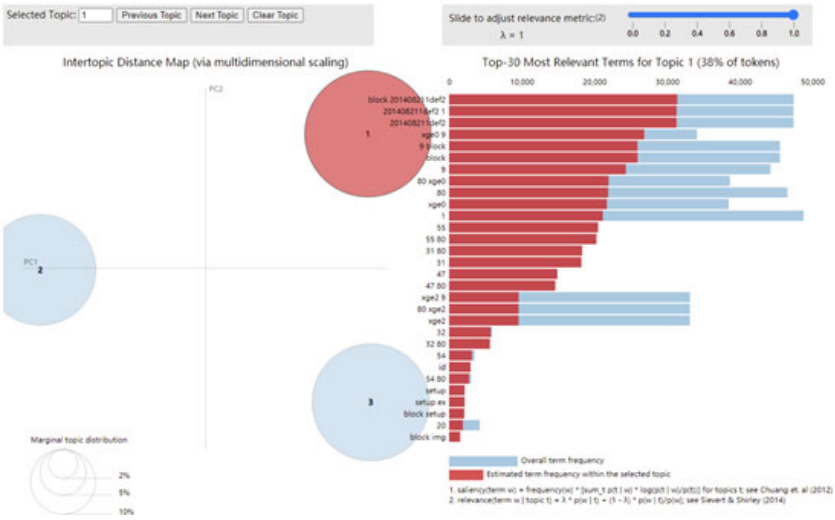


(b) "Content_0_Topic 2"

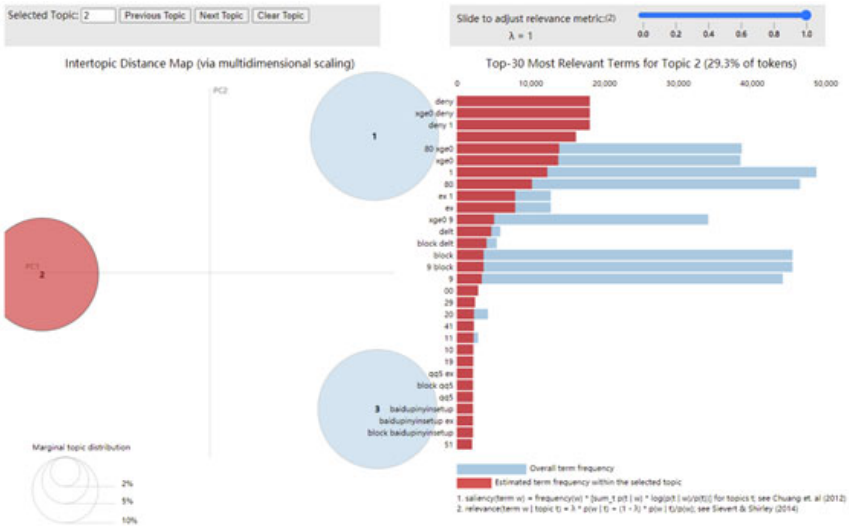


(c) "Content_0_Topic 3"

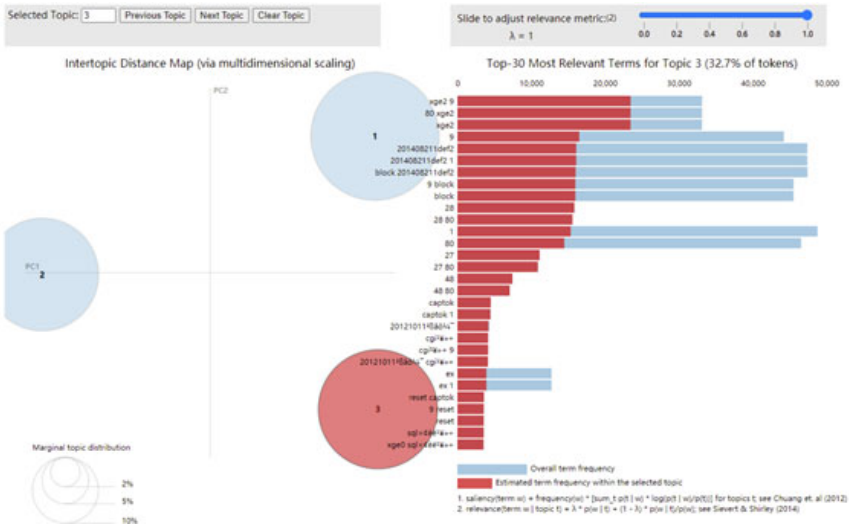
Figure 3.6: Generated “Content-topics" by LDA



(a) "other_info_0_Topic1"



(b) "other_info_0_Topic1"



(c) "other_info_0_Topic1"

Figure 3.7: Generated “Other-info-topics" by LDA

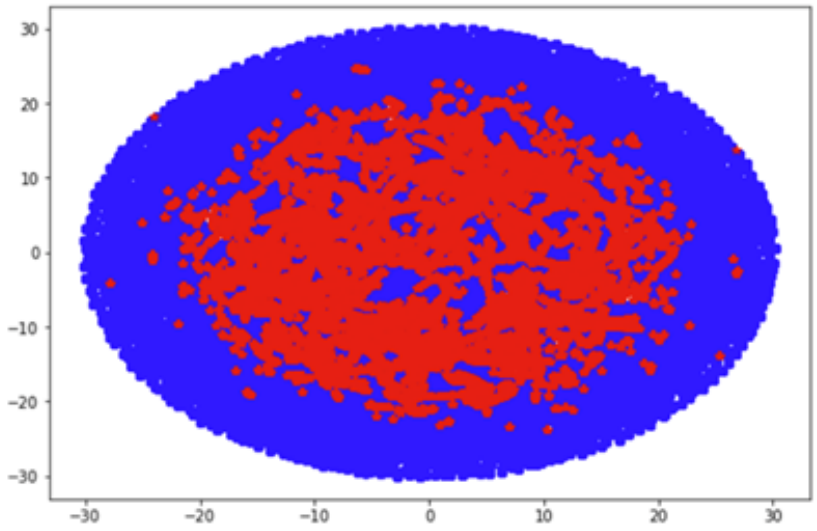


Figure 3.8: 54 Features Clustering

features for 500,000 rows of data. This diagram is not optimal because it is a forced compression of the sixty-dimensional data into two dimensions. However, the researcher's grouping effect is not very obvious from the diagram. Considering that a lot of information is lost when sixty dimensions are reduced to two dimensions, the researcher can only use it as a reference from this diagram. From this graph, we know that the grouping is probably distributed in a circular pattern in two dimensions. To reduce the burden on the algorithm and the memory pressure, the researcher reduced the data size from 500,000 rows to 100,000 rows of data. Hence, the experiment's previous 500,000 rows of data had been run for almost 2 hours.

The initial experimental goal is to test whether the algorithm could effectively classify the data. The ideal classification state would be to divide the log anomalies into two categories, anomalous and non-anomalous, and to ensure accuracy, also allow for a third category of data alert logs that may exist between the anomalous and non-anomalous groups.

3.4 Summary

This chapter highlights the background of LDA, and the implementation of LDA in this experiment. For log anomaly detection, since the raw data is not labelled in this experiment, feature extraction becomes crucial in the data analysis based on TF-IDF statistical features for keyword extraction. Keyword extraction based on TF-IDF statistical features is simple and easy, but this method ignores important low-frequency words and semantic features of the topic distribution within the document. Therefore, keyword extraction methods based on the implicit topic model of LDA (Blei et al., 2003) have gained attention in recent years. The effect of keyword extraction is closely related to the topic distribution of the training data. By combining TF-IDF and LDA, we obtained 54 features in log-0 and divided them into four categories, each of which has unique attributes and meanings and plays a decisive role in the log.

Chapter 4

AIOps Clusters using DBSCAN Algorithm

The DBSCAN clustering algorithm is a density-based clustering algorithm, proposed by Ester Martin *et al.*in 1996 (Ester, Kriegel, Sander, Xu et al., 1996). Unlike the K-Means algorithm, it does not require the number of clusters to be determined, but rather the number of clusters is inferred from the data. It can generate clusters for arbitrary shapes, which means that it defines the clusters as the most extensive set of densely connected points. It can classify regions with sufficient density into clusters, and can find clusters of arbitrary shapes in a noisy spatial database. The idea behind DBSCAN is that for each object in a cluster, the number of objects in the ε-neighborhood must be greater than or equal to a given threshold MinPts, i.e., the density in the cluster must be greater than or equal to a given threshold (Prado, 2019).

Rehman *et al.*present a detailed history of the development of the DBSCAN algorithm, the advantages and disadvantages of DBSCAN, and improvements based on the weaknesses of DBSCAN such as VDBSCAN, FDBDSCAN, GRIDBSCAN, IDBSCAN, EDBSCAN. These algorithms improve on some of the primary shortcomings of DBSCAN. The history of the DBSCAN algorithm is compiled to give the researcher a

detailed understanding of DBSCAN (Khan, Rehman, Aziz, Fong & Sarasvady, 2014). Celik *et al.*use DSBCAN to detect anomalies in time series data. The results show that DBSCAN is more advantageous in the face of anomaly detection methods (Çelik, Dadaşer-Çelik & Dokuz, 2011). Deng used DBSCAN to establish a security system to prevent unauthorized access to system resources and data (Deng, 2020).

4.1 The Algorithm of DBSCAN

There are a few definitions for DBSCAN methodology (Ester et al., 1996). First, Eps neighbourhood is needed to introduce.

Eps neighbourhood: Given an object p in a sample, its neighborhood is denoted by $N_{Eps}(p)$, which implies that the object p is the core in a d-dimensional region of radius Eps. For a more intuitive understanding, this is expressed mathematically as: $N_{Eps}(p) = \{q \in D \mid \mathrm{dist}(p,q) \leq \mathrm{Eps}\}$. In this mathematical expression, $D \in R^d$ is that d denotes the set of objects in d-dimensional space and $\mathrm{dist}(p,q)$, the distance between objects p and q in D. The figure 4.1 shows the red dot is the sample point. The red dot is the center of the circle and the blank area with ε as the radius is the ε-neighborhood.

Minpts: Denotes minimum number of points required to form a cluster.

Core points and boundary points: Suppose there is an object $p \in D$ in the sample and set the minimum number of objects minPts. If there is a $|N_{Eps}(p)| \geq \min Pts$, the object p is called a core point. Furthermore, if an object does not belong to a core point but is in the Eps neighborhood of a core point, that object is a boundary point. The other points are noise points. In other words, objects that are neither core points nor boundary points.

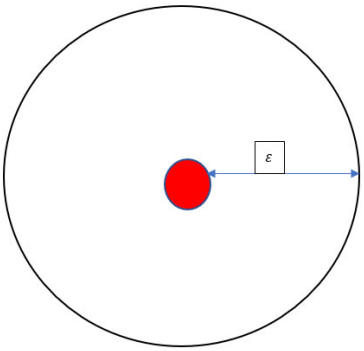


Figure 4.1: The Relationship of Eps Neighborhood

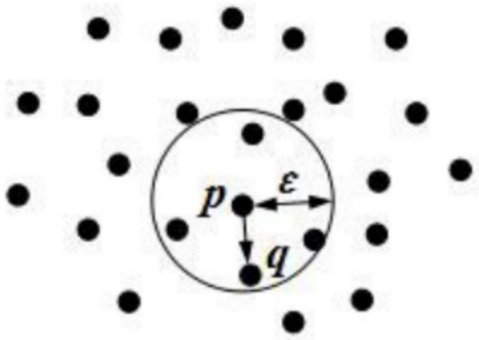
Direct Density Reachable(DDR): Next definition is direct density reachable(DDR) in Figure 4.2(a). Suppose $(p, q) \in$ dataset is D, p is a core point, and if the following condition is satisfied: $p \in N_{Eps}(q)$ and $N_{Eps}(q) \geq \min Pts$, then the object P is identified as starting from q, and the object is directly density reachable.

Density Reachable: Density reachable is defined in Figure 4.2(b) when $(p, q) \in$ data set as D, $q_i \in$ the dataset $D(1 \leqslant i \leqslant n), q_1, q_2, \cdots, q_n, q_1 = p, q_n = q$, q_{i+1} is a direct density reachable point for q_i, then q is a density reachable point for p.

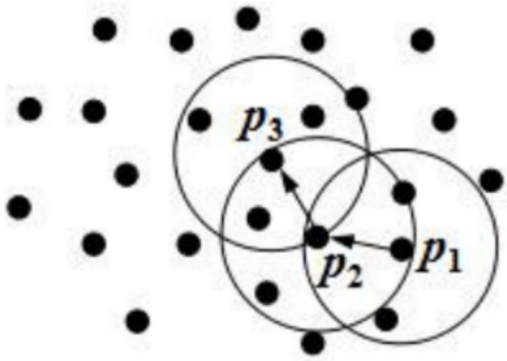
Density Connected: Similarly, density connected in Figure 4.2(c) could be defined when objects p and q are considered density connected if the dataset D contains an object O, and both objects p and q are density reachable from object O.

4.2 The Process of DBSCAN

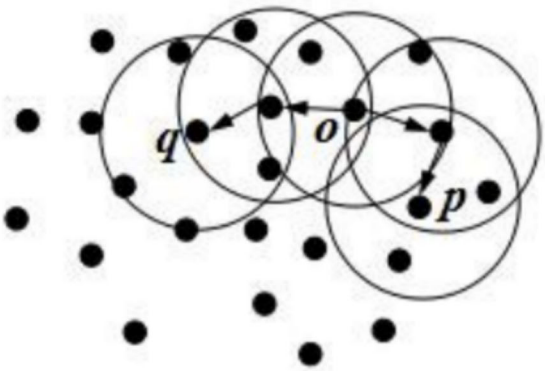
When applying DBSCAN algorithm to find a cluster, there are four steps involved in. Assume the input are dataset D, radius parameter ε and density threshold MinPts. The output are clustering results and noise data. The four steps in the execution of the DBSCAN algorithm are described as follows.



(a) Directly Density Reachable



(b) Density Reachable



(c) Density Connected

Figure 4.2: The Definition of DBSCAN Algorithm

1. Step 1: An unprocessed object p is randomly selected from the dataset D and its nearest neighbours satisfy the density threshold, that is called a kernel object.
2. Step 2: Iterate through the entire dataset to find all density reachable objects from object p to form a new cluster.
3. Step 3: Generate the final cluster result by density concatenation.
4. Step 4: Repeat steps 2 and 3 until all objects in the dataset are "processed".

The above description shows that density-based clustering is a set of “density-connected" objects to maximise the “density reachable". Density-based clustering is a set of “density connected" objects to maximise “density reachability". Objects that are not included in any clusters that are noisy data. According to the DBSCAN definition, the flow chart of DBSCAN can be shown in Figure 4.3.

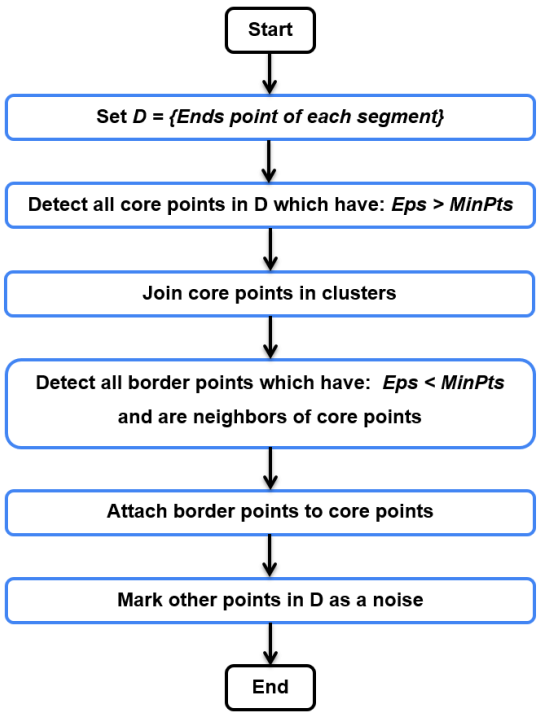


Figure 4.3: Flowchart of the DBSCAN Algorithm

Moreover, a clear demonstration of how DBSCAN performs stepwise clustering

can be found on a website called Naftaliharris (*Visualizing DBSCAN Clustering*, n.d.). On this website, the site provider offers a choice of different forms of datasets. The most classic of these datasets is the smiley face dataset. The Figure 4.4 shows the morphology of DBSCAN can be observed after clustering process. The clustering steps of DBSCAN described above can be better demonstrated.

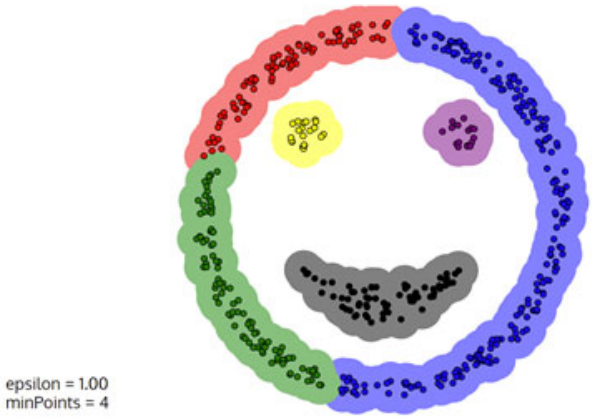


Figure 4.4: Naftaliharris Smile Face

4.3 DBSCAN Clustering Experiments

At the early stage of the experiment, system log files are used that contains 10 million rows of data. The system log files commonly consist of system operation log files, including System operation information, firewall attack information, external network access information, and virus attack information. In general, these types of data highly affect the system operation. Hence, AIOps needs to classifying the data, finding out the pattern cross the data, effectively distinguishing the abnormal information within the data and providing it to the IT staff, locating the issues,

We then divided 10 million rows of data file into 20 copies, in which there are 500,000 rows each. Also, the algorithm's feasibility will be well tested while avoiding memory shortage-related issues. We first used the syslog_0 file as a preliminary

experiment to find data patterns, and to see how well the algorithm could cluster this data. Selecting 500,000 rows of data from syslog_0, pre-process the data and substitute it into the DBSCAN algorithm. The eps value of 0.5 is the default value in this experiment, and assume that changing parameters may not have a significant impact on the final results at the initial experiment. At the beginning of the experiment, it remains unknown whether the DBSCAN algorithm will perform well on this type of data. The DBSCAN pre-clusting can be listed in the following.

Listing 4.1: DBSCAN Pre-clusting

```
y_pred = DBSCAN(eps=0.5).fit_predict(x)
df['labels']=y_pred[:1000]
fig = plt.figure(figszie=(8,8))
ax = fig.add_subplot(111)
df1 = df[df['labels']==0]
df2 = df[df['labels']==1]
df3 = df[df['labels']==2]
ax.scatter(df1[0],df1[1]),
    marker = 'x',colour='blue',s=40,label='class1'
ax.scatter(df2[0],df2[1]),
    marker = 'x',colour='green',s=40,label='class2'
ax.scatter(df3[0],df3[1]),
    marker = 'x',colour='red',s=40,label='class3'
plt.savefig('DBSCAN result.png')
plt.show()
```

It can be seen green, blue, brown, orange, pink and red in Figure 4.5. As one colour represents one type of data, the Figure 4.5 shows there are more than six types of data detected, which indicates that the clustering is not obvious when applying the DBSCAN

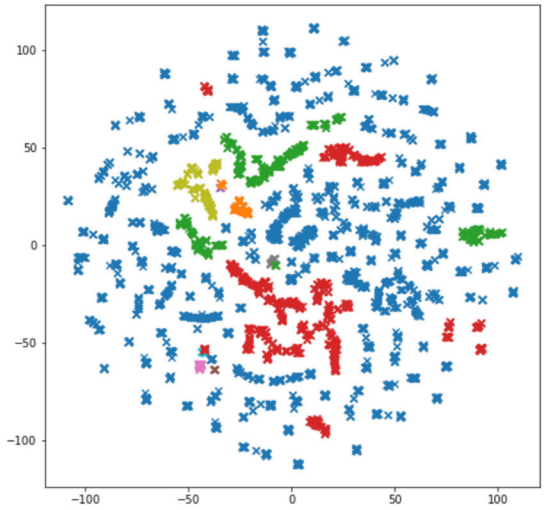


Figure 4.5: DBSCAN Clustering using 500,000 Rows

algorithm.

Next, we randomly reduced the amount of data from 500,000 rows to 100,000 rows data and stored them into a log file. The same DBSCAN algorithm testing process applied to the experiment. Interestingly, the results with better grouping compared to the previous results. It can be seen the patterns are getting obvious in Figure 4.6, only blue, red and green colors left.

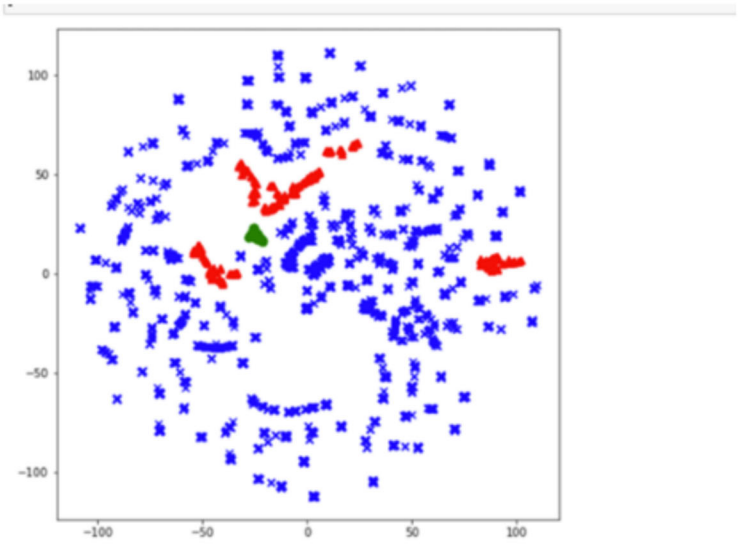


Figure 4.6: DBSCAN Clustering using 100,000 Rows

In Figure 4.6, when we reduced the amount of data from 500,000 rows to 100,000 rows, and change the eps value from 0.5 to 1, we experienced the memory overflow for our experiments. After conducting several experiments, we have found that it is hard for DBSCAN algorithm to process a large amount of data. The memory overflow issues are easily to occur.

4.4 Summary

In this chapter, we found the DBSCAN clustering algorithm is able to cluster our own data. Although it can not set the number of groupings itself, it can group clustering results using different colors. It also shows the number of cluster groups decreased when the number of testing data reduced.

Again this is part of the flaw within the DBSCA algorithm. Although the algorithm can find an arbitrary number and shape of clusters in a noisy data set, the clustering process is still flawed. The poor clustering quality also impact the clustering results, for instance, the sample set is not uniformly dense or the cluster spacing differences vary widely. Furthermore, the globally unique ε and Minpts parameters do not accommodate clustering of multi-density data with uneven densities (Ahmad & Dang, 2015). However, it can filter noise points and does not apply to other domains, such as the determination of malicious attacks in cybersecurity.

In our study of the DBSCAN algorithm, we found that the DBSCAN algorithm can discover arbitrarily shaped classes. However, the algorithmic process of DBSCAN constantly performs matrix calculations between two objects, which easily causes memory overflow when the amount of data is relatively large. High I/O consumption also causes problem when the data is not all loaded into memory. Moreover, DBSCAN deals with uneven data distribution but often does not get the correct clustering results when dealing with uneven data distribution. Due to the artificiality of the initial input

parameters and DBSCAN cannot be modified after being determined. These problems highlight the shortcomings and inadequacies of DBSCAN in the face of such data and help us to understand that the DBSCAN algorithm is not suitable for the subject of this experiment.

Chapter 5

Anomaly Detection by Local Outlier Factor

The local outlier factor in short LOF is a commonly used algorithm for anomaly detection by Breunig *et al.* (Breunig, Kriegel, Ng & Sander, 2000, 1999), which is able to find anomalous data points by measuring the local deviation of a given data point with respect to its neighbours. Most outlier detection algorithms determine outliers by density, angle, distance, or by dividing the hyperplane. The LOF algorithm is an outlier detection algorithm based on the density comparison of data points and is highly accurate.

In general, the LOF is a calculation approach that looks at the neighbors of a certain point to find out its density and compare this to the density of other points later on. LOF uses an local outlier factor to characterize the degree of local outliers of an object. In the LOF algorithm (Breunig et al., 2000), the neighborhood is determined based on the minimum number of neighbors k and the nearest neighbor distance, and the LOF is expressed as the ratio of the average reachable density of the neighborhood of a data object to its own reachable density by calculating the k-distance, reachable distance and reachable density of the object, the larger LOF is, the higher the outlier will be required.

In (T. Wang, Wei, Zhang, Zhong & Huang, 2014), Wang *et al.*grouped cloud virtual machines providing web services with load vectors, trained the workloads with incremental clustering algorithms, and finally used the LOF algorithm to detect whether the workloads were anomalous. Cheng (Cheng, Zou & Dong, 2019) first used the low-complexity iForest to quickly scan the dataset, prune the dataset, generate a candidate set of outliers, and then apply LOF to distinguish further the candidate set of outliers to obtain more real outliers. The proposed ensemble method utilizes the advantages of both algorithms, and the ensemble method can significantly improve the outlier detection rate and greatly reduce the time complexity.

5.1 The Algorithm of LOF

Based on the description of the LOF algorithm (Breunig et al., 1999), the degree of anomaly of a local outlier is related to the distribution of the surrounding samples. Several definitions are involved as follows:

k-distance: Among the nearest points p, the distance between the k^{th} nearest point and point p is called the k-distance (p).

Reachability distance: The definition of reachability distance is related to the K-neighbourhood distance. Given the parameter k, the reachability distance from data point p to data point o, reach-dist(p, o), is the maximum K-neighbourhood distance of data point o and the direct distance between data point p and point o in Equation 5.1.

$$reach\_dist_k(p,o) = \max\{k - distance(o), d(p,o)\} \tag{5.1}$$

Local reachability density: The definition of local reachability density is based on reachability distance. For a data point p, those data points whose distance from

point p is less than or equal to k-distance(p) are called its k-nearest-neighbour, denoted as, and the local reachability density of data point p is the reciprocal of its average reachability distance from neighbouring data points in Equation 5.2.

$$\mathrm{lr}\, d_k(p) = \frac{1}{\frac{\sum_{o \in N_k(p)} reach\_dist_k(p,o)}{|N_k(p)|}} \tag{5.2}$$

Local outlier factor: According to the definition of local reachable density, if a data point is relatively distant from other points, then obviously, its local reachable density is small. However, the LOF algorithm measures the anomaly of a data point not by its absolute local density, but by its relative density to the surrounding neighbouring data points. This has the advantage of allowing for uneven data distribution and varying densities. The local anomaly factor is defined in terms of the local relative density. The local relative density (local anomaly factor) of a data point p is the ratio of the average local reachable density of the neighbours of point p to the local reachable density of data point p.

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{\mathrm{lrd}(o)}{\mathrm{lrd}(p)}}{|N_k(p)|} = \frac{\sum_{o \in N_k(p)} \mathrm{lrd}(o)}{|N_k(p)|} / \mathrm{lrd}(p) \tag{5.3}$$

5.2 The Process of LOF

According to the definition of the local anomaly factor, when the LOF score of data point p is around 1, it indicates that the local density of data point p is about the same as its neighbours. When the LOF score of data point p is less than 1, it indicates that data point p is in a relatively dense region and does not look like an anomaly. When the LOF score of data point p is much greater than 1, it indicates that data point p is relatively distant from other points and is likely to be an outlier. This allows the LOF algorithm to be understood as follows.

1. For each data point, calculate its distance from all other points and order them from closest to farthest.

2. For each data point, find its k-nearest-neighbour and calculate the LOF score.

5.3 LOF Anomaly Detection Experiments

We first implement the LOF algorithm into our experiment log file. The proclustering process can be seen in the following listing.

Listing 5.1: LOF Preclusting

```
from sklearn.neigbors import LocalOutlierFactor
clf = LocalOutlierFactor(n_neighbors=700)
y_pred_lof = clf.fit_predict(x)
df['labels'] = y_pred_lof
fig = plt.figure(figsize =(8,8))
ax = fig.add_subplot(111)
df1 = df[df['labels']]== -1]
df2 = df[df['labels']==1]
ax.scatter(df1[0],df1[1]
        marker = 'x',color='blue',s=40,label='class1')
ax.scatter(df2[0],df2[1])
         marker = 'o',color='green',s=40,label='class2')
plt.show()
```

We then conducted the experiments to anomaly detect cluster using local outlier factor algorithm. In the experiment, we first kept the experiment data as DBSCAN algorithm. Using 500,000 amount of rows of data within the Syslog_0 file, and taking

a consistent data and processing approach. The data then is substituted into the encapsulation algorithm. The parameter n_neighbors is defined as the number of nearest data points selected for category prediction, and the initial value is set to 1. With continuous debugging, we increased the value to 700 and defined two clusters by blue and green cluster in Figure 5.1. It can be clearly observed the blue and green clusters are significantly different, and it was evident that the green marker cluster is much larger than the blue, and the two colors are mixed and cannot distinguish a precise classification.

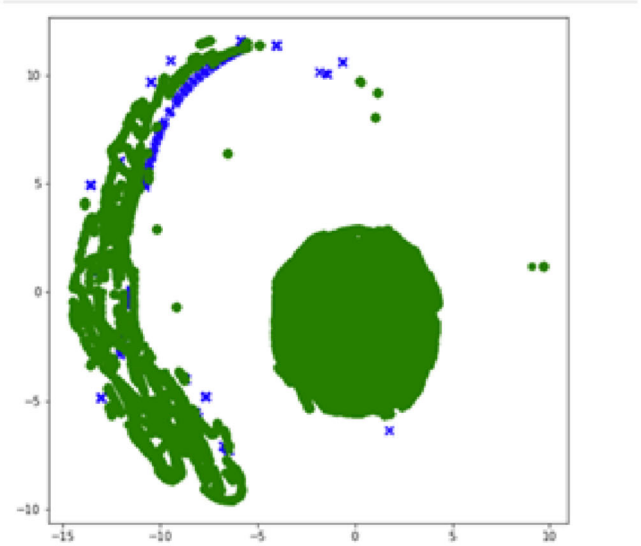


Figure 5.1: LOF Clustering Results using 500,000 Rows Data

Next, in order to find a better cluster pattern and compared with DBSCAN algorithm, we have used the same amount of 100,000 rows data in the LOF algorithm. Keeping the n_neighbors parameter is 700, and 100,000 rows of data were clustered as shown in Figure5.2. It has blue and green clusters, the green cluster is much bigger than the blue one. It shows the similar phenomenon when the data is 500,000 rows. Interestingly, when the parameter n_neighbors is reduced to 1, there is an only green cluster in the experiment result.

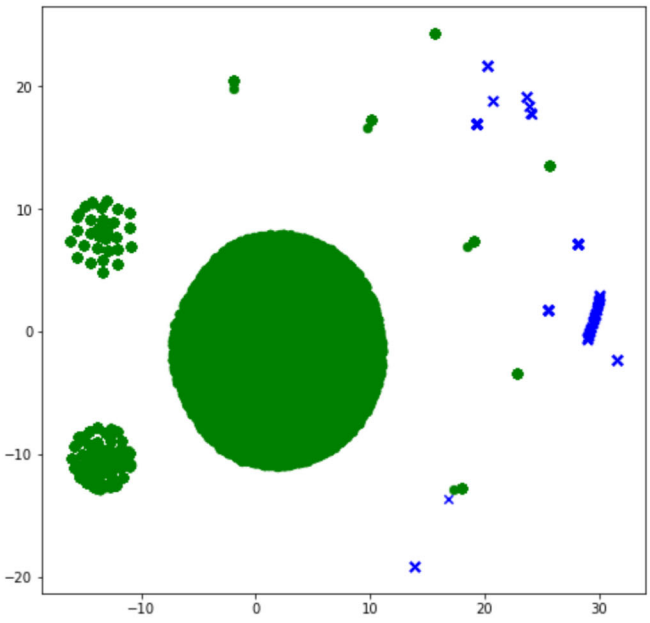


Figure 5.2: LOF Clustering Results using 100,000 Rows Data

5.4 Summary

In this chapter, anomaly detection by the local outlier factor is studied. After studying the fundamental formula of the LOF, numerical experiments were conducted. It can be seen from the experimental results using the duplicate 500,000 rows and 100,000 rows data as the DBSCAN approach, the blue and green label grouping are clustered by LOF algorithm in both experiments. However, it is not satisfied with the initial experimental objectives as the two clusters of data grouping were not prominent. When the parameter n_neighbors is set to 1, there is almost no blue cluster in both experiments.

The experimental results show the classification is not apparent, and a significant difference in size between blue and green clusters. The anomalous part of the data accounts for the data grouping large proportion. Also, it contradicts the core idea of the LOF algorithm that the lower the density, the more likely it is to be anomalous, and the two groupings in the graph overlap and cannot be grouped effectively. Therefore, the LOF clustering algorithm is not practical for detecting and clustering our data.

Chapter 6

K-means Clustering Algorithm for AIOps

The K-means clustering algorithm is an algorithm for statistical research. It has been proposed independently by Stéinhaus, Loyd and Balli & Halli (Jain, 2010; Likas, Vlassis & Verbeek, 2003) in their respective fields of study. Mac Queene summarized predecessors' work in 1967 (MacQueen et al., 1967), which laid out the stages of the K-means algorithm, and used powerful mathematical methods to establish its accuracy.

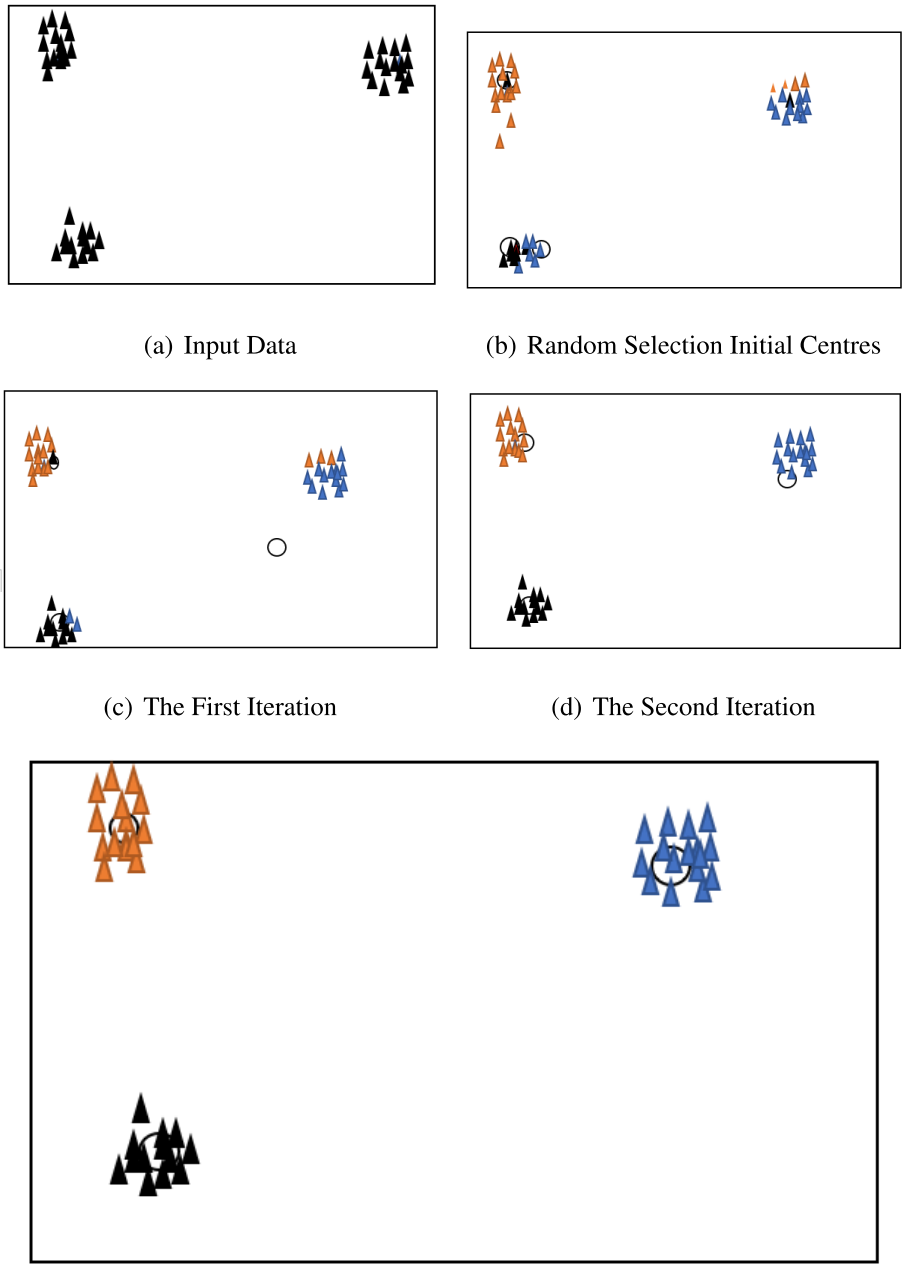
As the K-means algorithm has became a popular clustering approach, it is commonly more and more used in various fields, such as environmental science, health, biology, astronomy, and economics. Zhao *et al.* (Zhao, jun Wei & Wang, 2013) proposed a hybrid intrusion detection model by combining the advantages of K-means and Apriori algorithms. Han *et al.* (Han, 2011) proposed an algorithm based on a dynamic iterative process to calculate K-means clustering centres (MDKM). Kiss *et al.*proposed an anomaly detection method based on K-means algorithm for anomalous data in industrial control systems (Kiss, Genge, Haller & Sebestyén, 2014). Münz & Carle *et al.* (Münz, Li & Carle, 2007) on the other hand proposed a traffic anomaly detection scheme using the K-means clustering algorithm, mainly to face anomalous intrusion and attack

situations in network detection.

6.1 The Algorithm of K-means

As discussed above, the K-means algorithm is not only the most straightforward clustering algorithm but also the most popular and commonly used. In general, k-means algorithm is a simple iterative clustering algorithm that uses distance as an indicator of similarity to discover K classes in a given dataset. The centre of each class is obtained from the mean of all values in the class, and the cluster centre describes the centre of each class (Dabbura, 2020). The Euclidean distance is generally defined to measure similarity between data objects, which is inversely proportional to the distance between data objects. The more significant shows the similarity, the smaller distance could be achieved.

The basic theory of the K-means algorithm specifies the initial number of k clusters and k initial cluster centres. It updates the positions of the cluster centres according to the similarity between the data objects and the cluster centres, continuously decreasing the Sum of Squared Error (SSE) of the clusters and ending the clustering when the SSE no longer changes, or the objective function converges. The K-means clustering algorithm represents a continuous iterative process (Jain, 2010), as shown in the following Figure 6.1. The original dataset has 4 clusters, where x and y represent the values of the data points' horizontal and vertical coordinates, respectively. In Figure 6.1, the input data is the original dataset. The K-means algorithm is used to cluster the dataset. The final clustering result is obtained after two iterations of the dataset.



(a) Input Data

(b) Random Selection Initial Centres

(c) The First Iteration

(d) The Second Iteration

(e) Final Result

Figure 6.1: The K-means Algorithm Iterative Process

6.2 The Process of K-means Algorithm

The process of the K-means algorithm can be shown in the following Figure 6.2. The first step is to cluster the data into random K groups, so we initially have the number of K clusters at the beginning of the process. Each cluster has a center point, we call it cluster centroid. According to the Euclidean distance function, the cluster centre is determined by the value of K. For each sample data point, calculating the distance between it and the cluster centre are the next step. Then assign the data point to the nearest cluster centre. Repeat the process of calculating new cluster center based on the mean method, iterative calculation has been applied in the process until all cluster classes have the same points and complete the clustering.

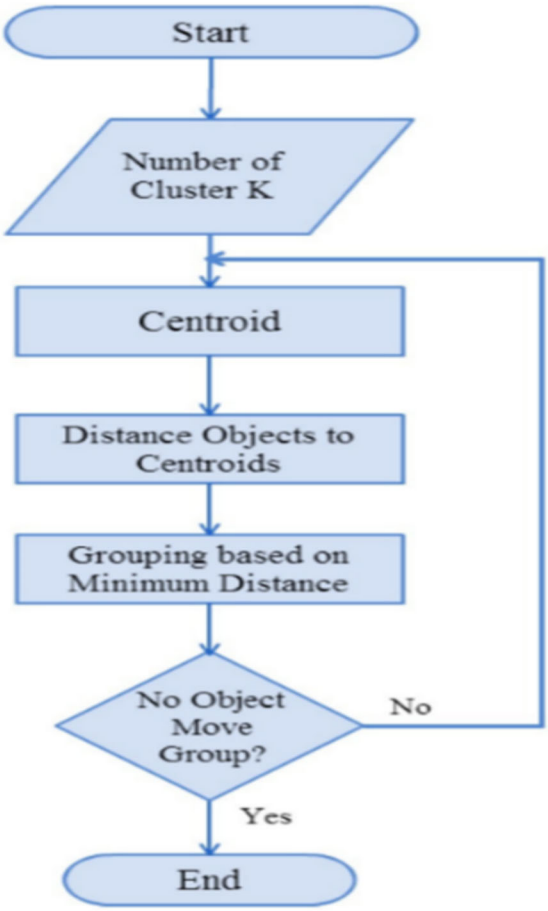


Figure 6.2: K-means Processing Steps

6.3 The Methodology of K-means

Since K-means clustering will be used to cluster our dataset, the methodology is studied thoroughly. Assume that the data sample set is Ω,

and $\Omega = \{x_i \mid x_i = (x_{i1}, x_{i2}, \ldots, x_{id}), \quad i = 1, 2, \ldots, n\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ is a d dimensional vector and n is the sample size. There are two data objects, x_i and x_j. The distance and similarity coefficients are usually used to measure the similarity between these two data objects.

The clustering algorithm uses the distance in the eigenspace as a metric to determine the similarity between two data objects x_i and x_j. By calculating the distance $d_{ij}(x_i, x_j)$ between them, some results could be defined. When x_i and x_j are more similar, the distance $d_{ij}(x_i, x_j)$ is smaller. when x_i and x_j are less similar, the distance $d_{ij}(x_i, x_j)$ is larger. There are many approaches to measure the similarity of the distance between two data x_i and x_j, and the following methods are commonly used. The distance $d_{ij}(x_i, x_j)$ is defined in the following equation.

$$d_{ij}(x_i, x_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^q \right)^{\frac{1}{q}}, q \geq 1, (i, j = 1, 2, \ldots, n) \tag{6.1}$$

First, the most commonly used measurement parameter is called “Minkowski Distance". When using the “Minkowski distance" method, the variables must be of the same magnitude. If they are not the same, the range of variation of the measurements can vary considerably. Therefore, when the units of the variables are different or when the measured values differ significantly, the data of the variables should be standardised. Hence, the distance is calculated by the standardised data.

Second, the “Manhattan Distance" was tent to utilize when q is equal to 1. It is

therefore simplify the distance equation that shows as follows:

$$d_{ij}\left(x_o, x_j\right) = \sum_{k=1}^{d} \left|x_{ik} - x_{jk}\right|, (i, j = 1, 2, \ldots, n) \tag{6.2}$$

Third, the “Euclidean Distance" is as follows when q is equal to 2. It is the most widely used calculated distance in cluster analysis. The clustering results obtained by choosing the Euclidean distance do not change with the translation and rotation of the feature space, but it may change with the linear transformation or other transformations that distort the distance relationship.

$$d_{ij}\left(x_i, x_j\right) = \left(\sum_{k=1}^{d} \left(x_{ik} - x_{jk}\right)^2\right)^{\frac{1}{2}}, (i, j = 1, 2, \ldots, n) \tag{6.3}$$

6.4 K-means Clustering Experiments

6.4.1 K-means Clustering Features

As K-means often produces different results due to the differences in the initial clusters, it is necessary to try several times to find the best solution. The preclusting is shown in the following listing.

Listing 6.1: K-means Preclusting

```
start = time.perf_counter()
tsne = STNE(n_components = 2,init='random',random)
end = time,perf_counter()
duration = end-statr
print = (f'Time␣lasped(duration)sec.')
df = pd.DataFrame(tsne.embedding.)
df['labels']=y_pred
```

```
df1 = df[df['labels']==0]
df2 = df[df['labels']==1]
fig = plt.figure(figsize=(9,6))
plt.plot(df1[0],'bo',df2[0],df2[1],'r*')
plt.savefig('Kmeans result.png')
plt.show()
```

The same testing dataset as the DBSCAN and LOF algorithms in K-means was chosen, and used the data within the Syslog_0 file. By pre-processing and extracting the themes, we substituted the data into the K-means algorithm and obtained the results in Figure 6.3. Comparing with the Figure 4.5 and Figure 5.1, it can be seen clearly in the Figure 6.3 that there are two classifications, blue and red groups. Since the two feature types are not entangled, we can effectively classify the two groups of features. The division is not affected by the data size, and the algorithm runs in a stable state without any memory overflow and program crashes. The preliminary experiment is to see if the algorithm works and help the researcher better understand the differences between the three algorithms.

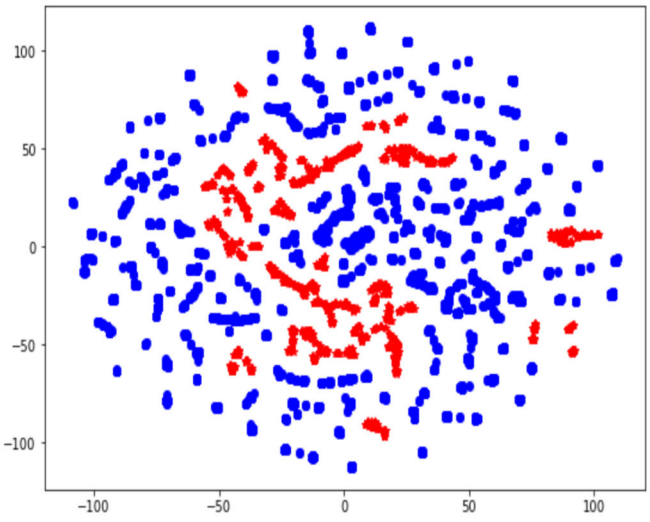


Figure 6.3: K-means Clustering Experimental Result

To determine the most suitable clustering algorithm for the current AIOps system, the remaining 19 log files were taken in the same way to better view all features and make the data distribution more consistent with the data segmentation requirements. The log files “2" and “15" were not taken due to the difficulties for processing the log files. For example, log file “2" is too large, and the log file“15" has error messages but cannot be edited. Thus, we consider this part of the data insignificant and has a negligible impact on the final results.

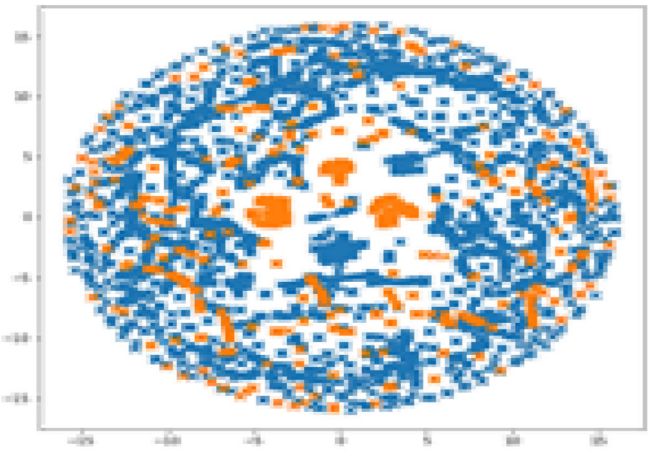
An essential reason for feature selection is that there are often dimensional disasters in practice caused by too many attributes. If the most important features can be selected to facilitate the construction of a model, then the dimensional catastrophe problem can be solved. In a statistical sense, feature selection has a similar purpose and application to that outlined previously for “T-SNE" dimensionality reduction. Another reason is that removing redundant features can significantly reduce the difficulty of the data analysis task, leaving critical information in the dataset, which is more beneficial for final model building and improving the accuracy and reliability of the results. One of the apparent concerns with the two graphs in Figure 6.4 is that the distribution of the data is not obvious owing to the vast number of characteristics that show.

In the process of refining the features, the researcher tried different types of feature combinations, but the results were not perfect showing because of the impact of too many redundant features on the clustering. To remove redundant features, the researcher calculated the relevance of features in the following three steps.

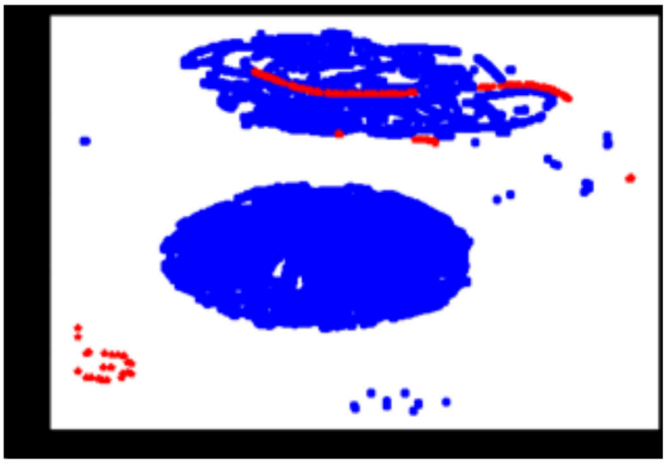
1. Calculate the feature relevance and discard the features with high similarity.

2. Discard the remaining features with many null values and discard those where all fields have the same value. The characteristic correlation is done by Figure 6.5.

3. Within the last seven remaining features, bring them into the algorithm to see who has the most effective grouping.



(a) Test Result1



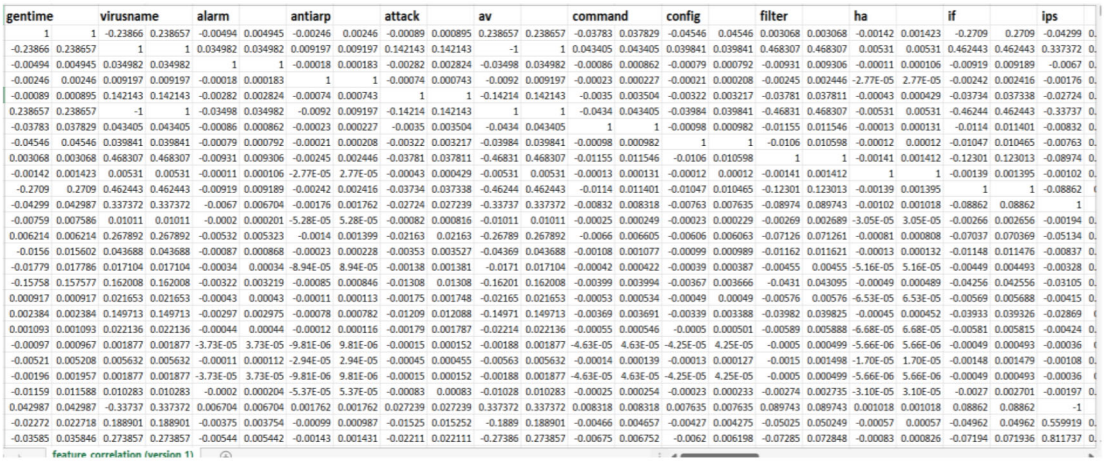
(b) Test Result2

Figure 6.4: Test Results with Different Features

After the characteristic correlation calculation, the column characteristics' absolute value and the result were obtained by the Pearson function: 0.14812. For better calculation and statistical observation, the value of 0.14812 has been set to 0.2.

6.4.2 Heat Maps Clustering Features

Heat maps are an intuitive representation of data and are helpful in spatial big data mining and knowledge discovery research. Determining the radius of influence of the data points, and plotting a heat map with a gradient in Figure 6.6. The correlation



gentime		virusname		alarm		antiarp		attack		av		command		config		filter		ha		if		ips	
1	1	-0.23866	0.238657	-0.00494	0.004945	-0.00246	0.00246	-0.00089	0.000895	0.238657	0.238657	-0.03783	0.037829	-0.04546	0.04546	0.003068	0.003068	-0.00142	0.001423	-0.2709	0.2709	-0.04299	0.
-0.23866	0.238657	1	1	0.034982	0.034982	0.009197	0.009197	0.142143	0.142143	-1	1	0.043405	0.043405	0.039841	0.039841	0.468307	0.468307	0.00531	0.00531	0.462443	0.462443	0.337372	0.
-0.00494	0.004945	0.034982	0.034982	1	1	-0.00018	0.000183	-0.00282	0.002824	-0.03498	0.034982	-0.00086	0.000862	-0.00079	0.000792	-0.00931	0.009306	-0.00011	0.000106	-0.00919	0.009189	-0.0067	0.
-0.00246	0.00246	0.009197	0.009197	-0.00018	0.000183	1	1	-0.00074	0.000743	-0.0092	0.009197	-0.00023	0.000227	-0.00021	0.000208	-0.00245	0.002446	-2.77E-05	2.77E-05	-0.00242	0.002416	-0.00176	0.
-0.00089	0.000895	0.142143	0.142143	-0.00282	0.002824	-0.00074	0.000743	1	1	-0.14214	0.142143	-0.0035	0.003504	-0.00322	0.003217	-0.03781	0.037811	-0.00043	0.000429	-0.03734	0.037338	-0.02724	0.
0.238657	0.238657	-1	1	-0.03498	0.034982	-0.0092	0.009197	-0.14214	0.142143	1	1	-0.0434	0.043405	-0.03984	0.039841	-0.46831	0.468307	-0.00531	0.00531	-0.46244	0.462443	-0.33737	0.
-0.03783	0.037829	0.043405	0.043405	-0.00086	0.000862	-0.00023	0.000227	-0.0035	0.003504	-0.0434	0.043405	1	1	-0.00098	0.000982	-0.01155	0.011546	-0.00013	0.000131	-0.0114	0.011401	-0.00832	0.
-0.04546	0.04546	0.039841	0.039841	-0.00079	0.000792	-0.00021	0.000208	-0.00322	0.003217	-0.03984	0.039841	-0.00098	0.000982	1	1	-0.0106	0.010598	-0.00012	0.00012	-0.01047	0.010465	-0.00763	0.
0.003068	0.003068	0.468307	0.468307	-0.00931	0.009306	-0.00245	0.002446	-0.03781	0.037811	-0.46831	0.468307	-0.01155	0.011546	-0.0106	0.010598	1	1	-0.00141	0.001412	-0.12301	0.123013	-0.08974	0.
-0.00142	0.001423	0.00531	0.00531	-0.00011	0.000106	-2.77E-05	2.77E-05	-0.00043	0.000429	-0.00531	0.00531	-0.00013	0.000131	-0.00012	0.00012	-0.00141	0.001412	1	1	-0.00139	0.001395	-0.00102	0.
-0.2709	0.2709	0.462443	0.462443	-0.00919	0.009189	-0.00242	0.002416	-0.03734	0.037338	-0.46244	0.462443	-0.0114	0.011401	-0.01047	0.010465	-0.12301	0.123013	-0.00139	0.001395	1	1	-0.08862	0.
-0.04299	0.042987	0.337372	0.337372	-0.0067	0.006704	-0.00176	0.001762	-0.02724	0.027239	-0.33737	0.337372	-0.00832	0.008318	-0.00763	0.007635	-0.08974	0.089743	-0.00102	0.001018	-0.08862	0.08862	1	
-0.00759	0.007586	0.01011	0.01011	-0.0002	0.000201	-5.28E-05	5.28E-05	-0.00082	0.000816	-0.01011	0.01011	-0.00025	0.000249	-0.00023	0.000229	-0.00269	0.002689	-3.05E-05	3.05E-05	-0.00266	0.002656	-0.00194	0.
0.006214	0.006214	0.267892	0.267892	-0.00532	0.005323	-0.0014	0.001399	-0.02163	0.02163	-0.26789	0.267892	-0.0066	0.006605	-0.00606	0.006063	-0.07126	0.071261	-0.00081	0.000808	-0.07037	0.070369	-0.05134	0.
-0.0156	0.015602	0.043688	0.043688	-0.00087	0.000868	-0.00023	0.000228	-0.00353	0.003527	-0.04369	0.043688	-0.00108	0.001077	-0.00099	0.000989	-0.01162	0.011621	-0.00013	0.000132	-0.01148	0.011476	-0.00837	0.
-0.01779	0.017786	0.017104	0.017104	-0.00034	0.00034	-8.94E-05	8.94E-05	-0.00138	0.001381	-0.0171	0.017104	-0.00042	0.000422	-0.00039	0.000387	-0.00455	0.00455	-5.16E-05	5.16E-05	-0.00449	0.004493	-0.00328	0.
-0.15758	0.157577	0.162008	0.162008	-0.00322	0.003219	-0.00085	0.000846	-0.01308	0.01308	-0.16201	0.162008	-0.00399	0.003994	-0.00367	0.003666	-0.0431	0.043095	-0.00049	0.000489	-0.04256	0.042556	-0.03105	0.
0.000917	0.000917	0.021653	0.021653	-0.00043	0.00043	-0.00011	0.000113	-0.00175	0.001748	-0.02165	0.021653	-0.00053	0.000534	-0.00049	0.00049	-0.00576	0.00576	-6.53E-05	6.53E-05	-0.00569	0.005688	-0.00415	0.
0.002384	0.002384	0.149713	0.149713	-0.00297	0.002975	-0.00078	0.000782	-0.01209	0.012088	-0.14971	0.149713	-0.00369	0.003691	-0.00339	0.003388	-0.03982	0.039825	-0.00045	0.000452	-0.03933	0.039326	-0.02869	0.
0.001093	0.001093	0.022136	0.022136	-0.00044	0.00044	-0.00012	0.000116	-0.00179	0.001787	-0.02214	0.022136	-0.00055	0.000546	-0.0005	0.000501	-0.00589	0.005888	-6.68E-05	6.68E-05	-0.00581	0.005815	-0.00424	0.
-0.00097	0.000967	0.001877	0.001877	-3.73E-05	3.73E-05	-9.81E-06	9.81E-06	-0.00015	0.000152	-0.00188	0.001877	-4.63E-05	4.63E-05	-4.25E-05	4.25E-05	-0.0005	0.000499	-5.66E-06	5.66E-06	-0.00049	0.000493	-0.00036	0.
-0.00521	0.005208	0.005632	0.005632	-0.00011	0.000112	-2.94E-05	2.94E-05	-0.00045	0.000455	-0.00563	0.005632	-0.00014	0.000139	-0.00013	0.000127	-0.0015	0.001498	-1.70E-05	1.70E-05	-0.00148	0.001479	-0.00108	0.
-0.00196	0.001957	0.001877	0.001877	-3.73E-05	3.73E-05	-9.81E-06	9.81E-06	-0.00015	0.000152	-0.00188	0.001877	-4.63E-05	4.63E-05	-4.25E-05	4.25E-05	-0.0005	0.000499	-5.66E-06	5.66E-06	-0.00049	0.000493	-0.00036	0.
-0.01159	0.011588	0.010283	0.010283	-0.0002	0.000204	-5.37E-05	5.37E-05	-0.00083	0.00083	-0.01028	0.010283	-0.00025	0.000254	-0.00023	0.000233	-0.00274	0.002735	-3.10E-05	3.10E-05	-0.0027	0.002701	-0.00197	0.
0.042987	0.042987	-0.33737	0.337372	0.006704	0.006704	0.001762	0.001762	0.027239	0.027239	0.337372	0.337372	0.008318	0.008318	0.007635	0.007635	0.089743	0.089743	0.001018	0.001018	0.08862	0.08862	-1	
-0.02272	0.022718	0.188901	0.188901	-0.00375	0.003754	-0.00099	0.000987	-0.01525	0.015252	-0.1889	0.188901	-0.00466	0.004657	-0.00427	0.004275	-0.05025	0.050249	-0.00057	0.00057	-0.04962	0.04962	0.559919	0.
-0.03585	0.035846	0.273857	0.273857	-0.00544	0.005442	-0.00143	0.001431	-0.02211	0.022111	-0.27386	0.273857	-0.00675	0.006752	-0.0062	0.006198	-0.07285	0.072848	-0.00083	0.000826	-0.07194	0.071936	0.811737	0.

Figure 6.5: Characteristic Correlation

threshold of 0.2 was used to look at the correlation of the 20 log data features using heat map analysis algorithm. Any threshold more significant than 0.2 is shown in a white square in the plot, and only one strong correlation is left.

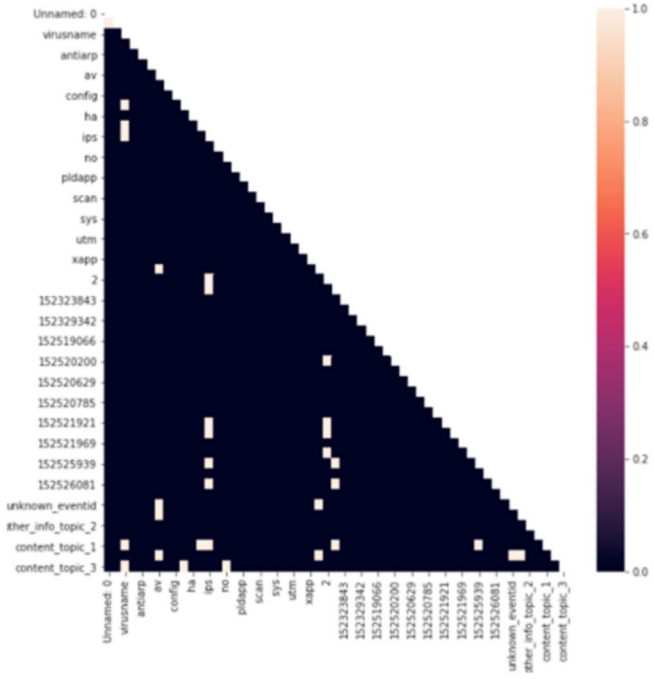


Figure 6.6: Features in Heat Maps

A graph of the effect of features is generated based on the results, with the horizontal axis being the features and the vertical axis being the mean of the features in Figure 6.7.

In Figure 6.7, a blue line represents a “Group1", and a green line represents another “Group2". Here we are looking for the difference between the two groups, so we take the mean of each feature to compare and then look at the feature with the most significant difference from the graph as our choice. As it shows in Figure 6.7, there are seven highlighted red circles. The seven selected features are: “virus name", “attack", “pldrun", “sensor", “2", “other_info_topic_2", “content_topic_2 ". The three features that carried the most information were filtered to obtain “virus name",“2", “Content_topic_2".

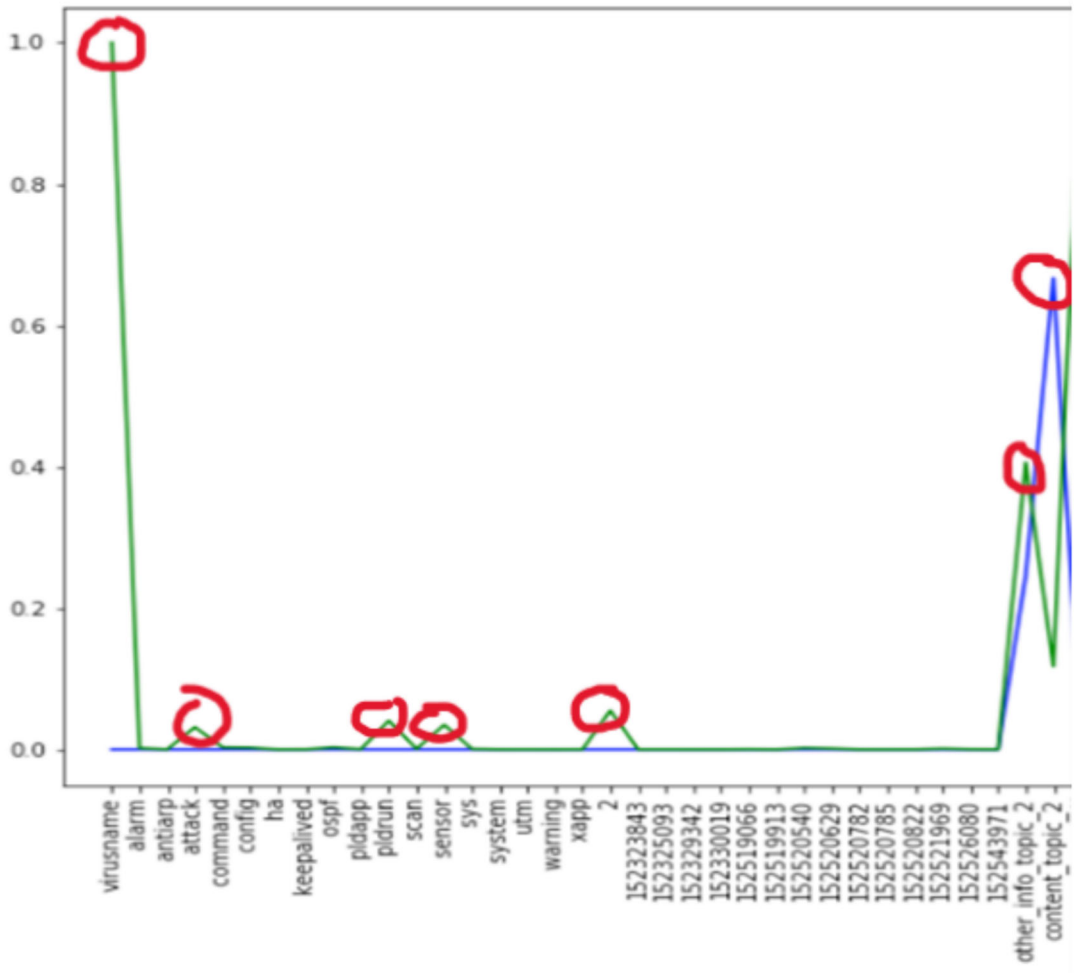


Figure 6.7: Comparing Selected Features

The comparison of the most important features for “Group1" and “Group2" are shown in Table 6.1. It can be seen that “Mean" represents the topic in the topic. From the

statistical LDA topic model, it can be seen that “block", “deny", and “attack" account for a large proportion of the keywords, suggesting that this topic may be related to attacks and firewall defence activities. In the results for the “Group2", the feature “topic_2" is much higher than in “Group1". The feature “Content_topic_2" is transformed to “1", meaning the event level is 2. Otherwise, the event level is 0. The percentage of events with event-level 2 in the “Group2" is 8.4%, that is higher than 1.5% in “Group1". The event level of the “Group2" is generally higher. The reason for identifying “2" as the event-level feature is that it does not correlate well with other numerical features but varies significantly between clusters. And the reason for discarding other event-level “0" and “3" is because they are correlated with different features. But if only one event-level feature is kept, the feature “2" carries the most information. In terms of virus name, a virus name of 0 means not-virus, and 1 means something else. There is a lot of virus-related information in “Group2", while 68.3% of the logs in the “Group1" are not a virus, which is relatively more minor than “Group2". This means that “Group2" can be seen as a group with a higher risk than “Group1" . This means that “Group2" can be seen as a group with multiple “Group2" logs in automated operations and maintenance.

Group1	Group2
2 93.3%	2 6.7%
Content_topic_2 Mean:16% 2:98.5% 1: 1.5%	Content_topic_2 Mean:85% 2:91.5% 1:8.4%
Virus name 68.3%	Virus name 100%

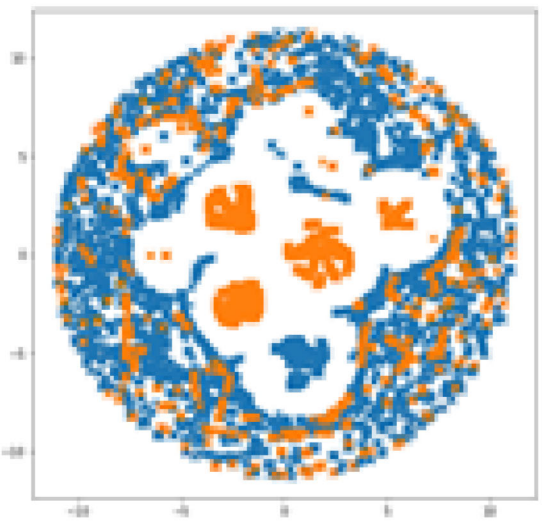
Table 6.1: Three Features Captured in K-means

Finally, to verify the impact of the remaining features on the results, we also tested different combinations of features. In Figure 6.8, the three images are composed of words with different characteristics, and the clustering effect is very similar in all three

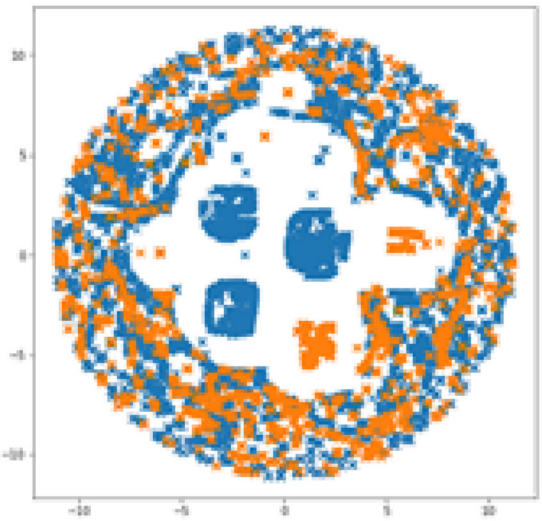
images. This proves that the clustering effect of these combinations of features is inferior and does not meet our requirements for clustering. The suitable characteristic is depicted in Figure 6.9. After determining the feature, the clustering on the training set was examined first, and it can be seen that the images appear to be grouped more clearly with the other feature combinations, whereas the test set has miscellaneous points written on it but still has clear grouping based on the overall view. Figure 6.10 depicts the keywords of the second topic following the LDA grouping, which are utilized to some extent to grasp the meaning of the grouping. Due to the large amount of data, the long running time and the crash caused by the lack of memory, 1,000,000 rows of data were selected from the 18 logs, and 70% of the data were used as the training set of 700,000 rows and the other 300,000 rows were used as the 30% test set for validation.

6.5 Summary

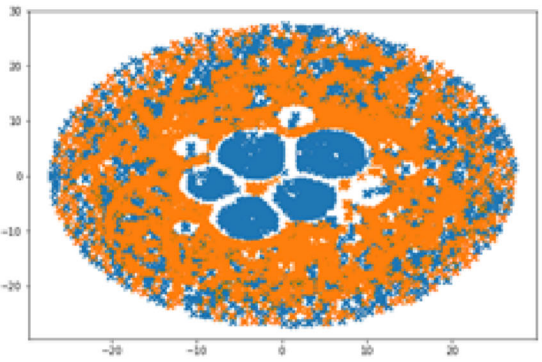
In the chapter on K-means, the researcher introduces the background of the algorithm, the principles and the basic formulas. In the preliminary experiments, K-means showed promising results in effectively classifying the data and adapting to extensive data clustering needs. In the final experiments, K-means was chosen as the final method, with TFIDF, LDA and K-Means as the tenant, to cluster the one million rows of valid data extracted. Then, through correlation analysis of the features, redundant features were removed, and the remaining features were tested separately for clustering, resulting in a valid clustering map that ultimately helped the researcher clarify the outliers in this log.



(a) Clustering Results “attack",“ sensor" and “pldrun"

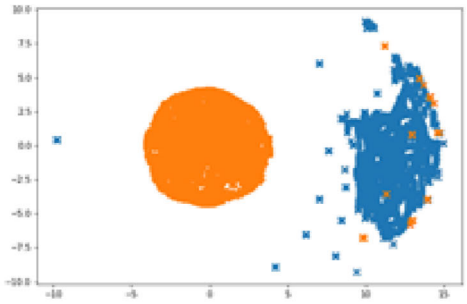


(b) Clustering Results “sensor", “pldapp" and “other_info_topic_2"

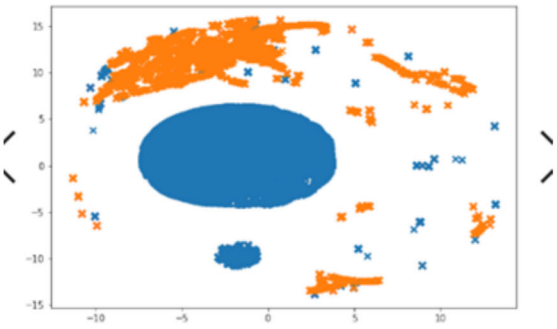


(c) Clustering Results “content_topic_2", “other_info_topic_2" and “pldrun"

Figure 6.8: Clustering Result with Different Features



(a) Train_Result



(b) Test_Result

Figure 6.9: Train and Test Result

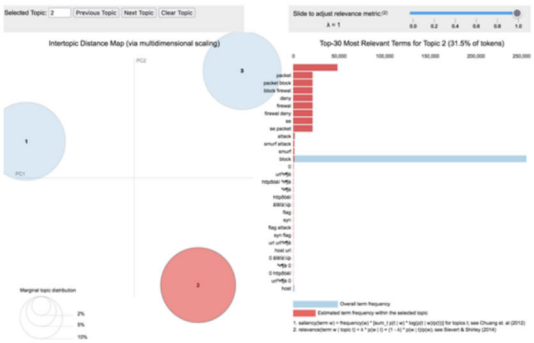


Figure 6.10: Keyword Distribution of “content_topic_2"

Chapter 7

AIOps System Model Design based on Log Anomaly Detection

The previous chapters give a brief introduction to clustering algorithms and existing solutions for log anomaly detection methods. This is followed by a complete description of the proposed techniques, including log data pre-processing and log anomaly detection, based on the K-means algorithm for detecting anomalous data. This chapter presents an initial prototype design of the log monitoring solution, and the researcher as analysed the AIOps anomaly detection requirements and designed a model structure based on log anomaly detection.

7.1 System Requirements Analysis

Firstly, in order to implement a complete log anomaly detection solution, the designed log anomaly detection system needs to contain the following functions.

1. Log collection: to monitor the log data generated on each server of the client system in real-time, cache the generated log streams centrally and transferred them uniformly.

2. Log storage: the collected log data is stored centrally in either files or database tables on a permanent storage device.

3. Model training: the stored log data is trained according to the theory to generate a set of behavioural patterns updated periodically.

4. Log stream analysis: real-time analysis of log streams based on behavioural patterns, abnormal values calculation, and system status judgment by comparison with abnormal thresholds.

5. Performance requirements: the ability to accurately predict possible system anomalies in a timely and correct manner and operate stably under varying load conditions are required. To specific, the system can quickly query indicators and view and configure information about each indicator.The system can analyse the cause of failures and make relevant judgements in a timely and effective manner. Finally, the system can provide a comprehensive analysis of O&M in a timely and effective manner.

The researcher has prioritised the design of the overall system framework by combining the above requirements. There are two sub-systems in the overall AIOps monitoring system: the management system and the diagnostic system. The front-end and back-end data collected are aggregated and pre-processed. In the diagnostic system, fault analysis consists of various algorithms to cope with different data models. The AIOps monitoring system framework can be seen in Figure 7.1.

7.2 AIOps System Model Design

According to the framework of the AIOps monitoring system, the log data should be collected in the first stage. The log collection module allows monitoring and collecting

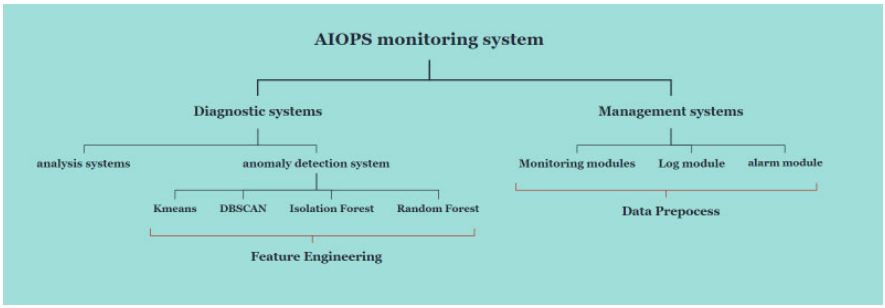


Figure 7.1: AIOps Monitoring System Framework

logs generated by the monitored system in real-time. That collects logs under conditions that do not interfere with the operation of the monitored system under any circumstances. The system architecture can be shown in Figure 7.4. Logstash is often used with ElasticSearch and Kibana to form a complete log collection, query and presentation system (Aeri & Tukadiya, 2015). An excellent open-source log management solution can be considered later on.

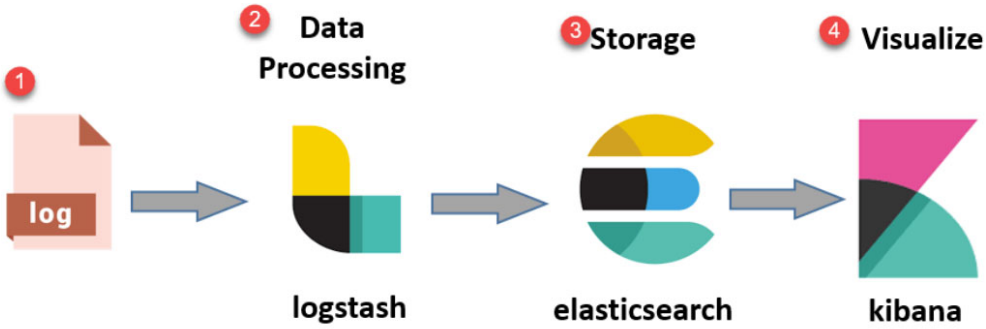


Figure 7.2: LogStash Processing System

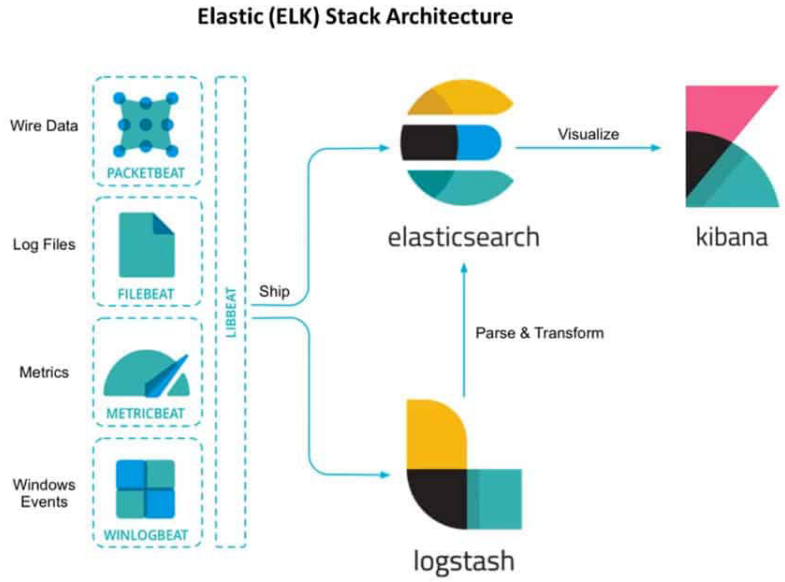


Figure 7.3: LogStash Architecture

Figure 7.2 depicts the whole processing pipeline, with Logstash retrieving data from numerous data sources, such as log files, using the input plugin, filtering and processing the data, and finally sending it to Elasticsearch for display through Kibana.

Figure 7.3 depicts the extension of Logstash data collection nodes to several computers, collecting various sorts of data, parsing the data, delivering it to an Elasticsearch server for storage, and lastly, querying and creating log reports in Kibana.

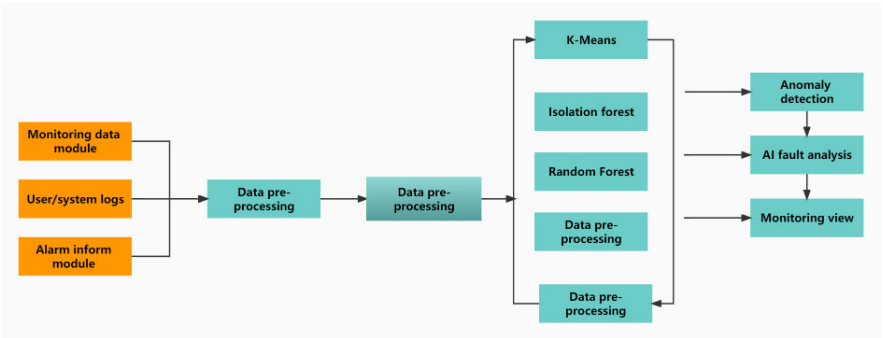


Figure 7.4: AIOps System Architecture

After that, the log storage module is responsible for storing log data and has a straightforward but essential function. The stored logs are used as a training set for log

analysis and therefore need to be easily viewed and quickly accessed. The log storage module usually consists of one or several storage servers, which maintain the file system on which the logs are stored.

The log analysis module finally implements the log anomaly detection method, including log pre-processing, detection model training and final anomaly detection. As this module is the main objective of this experiment, the researcher makes a summary diagram of the anomaly detection and model training based on the steps of this experiment. It considers that the log analysis module receives different data inputs from alarm information, business logs, detection information. The data stream from the log acquisition module is used to perform real-time anomaly detection and send the results to the monitoring attempts. The log data from the log storage module is permanently stored and is used to train the anomaly detection model. As the stored log data constantly changes, the detection model also needs to be updated periodically.

The training or updating process for the behavioural model shows in Figure 7.5. The logs are read from the log storage module, then the logs are regularised, including merging redundant rows, removing redundant characters, transforming log levels and de-parameterising log content. Then the clustering algorithm is invoked to categorise the logs into different types at the appropriate granularity. Next, the model training algorithm is invoked to convert the log data into text format and generate a behavioural topic model to obtain a topic model of the data. The next step is to call the model training algorithm to convert the log data into text format and generate a behavioural topic model to obtain a topic model of the data.

Finally, the logging anomaly detection module is designed to determine any anomalies in the log stream in Figure 7.6. When the system reads the log stream and detects it, it first extracts the characteristics of the indicators using the feature extraction function. It passes them into the trained log anomaly detection model.

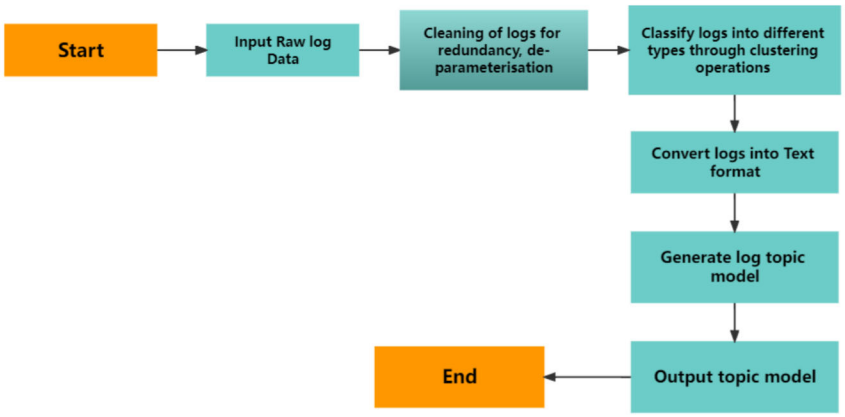


Figure 7.5: The Process of Log Analysis Module

7.3 Summary

The prototype implementation of a log monitoring system based on log anomaly detection is described in-depth in this chapter. The system requirements are first examined, then the overall framework of the system and its modules are described. Finally, the log collection module and the storage and analysis module of the monitoring system are introduced in turn, with the functional framework diagram and flowchart used to support the description.

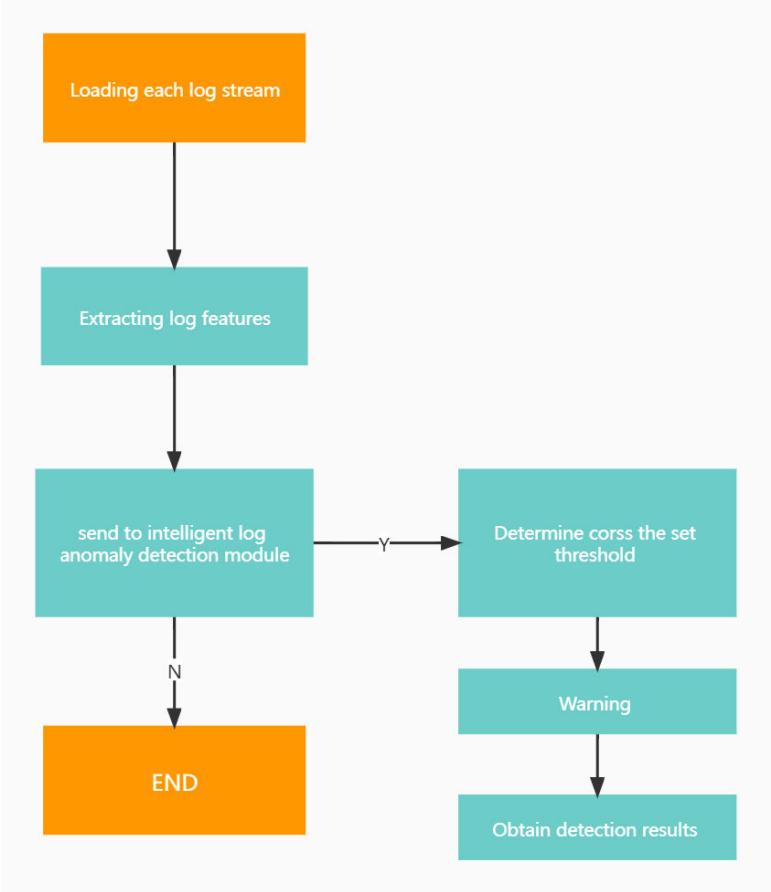


Figure 7.6: The Process of Logging Anomaly Module

Chapter 8

Conclusion

This thesis builds a prototype model for AIOps log anomaly detection, and a compelling analogy to many kinds of unsupervised algorithms is made. It also covers the same research topics in this thesis and accomplishes consistent research contributions. In this chapter, we summarize the contribution that we have made, including clustering our dataset using unsupervised machine-learning algorithms, the entire evolution of AIOps, the difficulties with the first constructed model, and the future work direction.

8.1 Summary of Contribution

The research direction of AIOps is relatively open, and log monitoring has always been an essential part of the research direction of AIOps. By collecting and analyzing the log data, the status of system can be analyzed, and abnormal events can be predicted. The thesis proposes an anomaly detection method to log data based on LDA topic refinement by analyzing the current research and the problems faced. The proposed methodology and the work accomplished in this thesis are reflected in the following aspects.

First, a log pre-processing method based on log normalisation and clustering is proposed. A series of regularisation methods formatted the logs and then clustered them

using a K-means based clustering method with the main direction of density distance.

Second, the LDA topic model is proposed to find log outliers in the face of the inability to interpret logs manually in a regular way, using the LDA topic model to refine the topic of the location content. The algorithm and the principle of LDA topics are introduced. The data is initially cleaned to establish data selection. The keyword extraction algorithm of TF-IDF is combined with LDA topics to analyse the association between topics and themes while predicting possible system anomalies by judging the magnitude of the correlation coefficient. The effectiveness of the log detection method proposed in this thesis is demonstrated by comparing the test results with several other algorithms on the experimental dataset.

Finally, a preliminary design of the AIOps-based log detection framework has been presented. A log monitoring system was designed and implemented with a log collection module, storage, and analysis modules in conjunction with existing commercial solutions and collected materials.

There are also some constraints in the thesis. Since the data set obtained is limited by the unsupervised learning algorithm and the inability to interpret the logs correctly, so the accuracy and recall statistics are omitted. The initial collection of a large dataset was not conducive to data analysis, and the long run of the algorithm resulted in a lack of system memory, which led to the initial experiments being discarded in the proposed algorithm. As a sequence, less reference for subsequent experiments. Due to the constraints, the experiment only roughly classifies the historical log anomaly detection problem into statistics without an accurate rate, F1 Score.

8.2 Future Directions of Research

There is plenty of work for researcher to do in the near future. Since the data is unlabeled in raw form, finding the proper methods for parsing is complex and requires a lot of

research and study of existing parsing methods. The choice of the correct dimensionality reduction algorithm is a challenge for this project, as it is crucial in the direction of feature engineering visualization in the face of multi-feature, high-latitude data.

In future, the researcher will continue to improve the performance of the models, including anomaly detection models and fault analysis. The historical log anomaly detection should achieve accuracy, for instance, anomaly detection and analysis without relying on manual tagging data. The functionality of the AIOps framework should be extended, such as efficiency improvement, cost management, and other aspects of quality assurance.

References

Aeri, A. & Tukadiya, S. (2015). A comparative study of network based system log management tools. In *2015 international conference on computer communication and informatics (iccci)* (pp. 1–6).

Ahmad, H. & Dang, S. (2015). Performance evaluation of clustering algorithm using different dataset. *International Journal of Advance Research in Computer Science and Management Studies*, *8*.

Bao, L., Li, Q., Lu, P., Lu, J., Ruan, T. & Zhang, K. (2018). Execution anomaly detection in large-scale systems through console log analysis. *Journal of Systems and Software*, *143*, 172–186.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, *3*, 993–1022.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. (1999). Optics-of: Identifying local outliers. In *European conference on principles of data mining and knowledge discovery* (pp. 262–270).

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 acm sigmod international conference on management of data* (pp. 93–104).

Chandola, V., Banerjee, A. & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 1–58.

Cheng, Z., Zou, C. & Dong, J. (2019). Outlier detection using isolation forest and local outlier factor. In *Proceedings of the conference on research in adaptive and convergent systems* (pp. 161–168).

Chuvakin, A., Schmidt, K. & Phillips, C. (2012). *Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management*. Newnes.

Dabbura, I. (2020, Aug). *K-means clustering: Algorithm, applications, evaluation methods, and drawbacks.* Towards Data Science. Retrieved from `https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a`

Deng, D. (2020). Research on anomaly detection method based on dbscan clustering algorithm. In *2020 5th international conference on information science, computer technology and transportation (isctt)* (p. 439-442). doi: 10.1109/ISCTT51595.2020.00083

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, pp. 226–231).

Fronza, I., Sillitti, A., Succi, G., Terho, M. & Vlasenko, J. (2013). Failure prediction based on log files using random indexing and support vector machines. *Journal of Systems and Software*, *86*(1), 2–11.

Fu, Q., Lou, J.-G., Wang, Y. & Li, J. (2009). Execution anomaly detection in distributed systems through unstructured log analysis. In *2009 ninth ieee international conference on data mining* (pp. 149–158).

Gaikwad, R., Deshpande, S., Vaidya, R. & Bhate, M. (2021). A framework design for algorithmic it operations (aiops). *Design Engineering*, 2037–2044.

Goldstein, M. & Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, *11*(4), e0152173.

Han, L. (2011). Using a dynamic k-means algorithm to detect anomaly activities. In *2011 seventh international conference on computational intelligence and security* (pp. 1049–1052).

He, S., Zhu, J., He, P. & Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. In *2016 ieee 27th international symposium on software reliability engineering (issre)* (pp. 207–218).

Iturbe, M., Garitano, I., Zurutuza, U. & Uribeetxeberria, R. (2017). Towards large-scale, heterogeneous anomaly detection systems in industrial networks: A survey of current trends. *Security and Communication Networks*, *2017*.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, *31*(8), 651–666.

Khan, K., Rehman, S. U., Aziz, K., Fong, S. & Sarasvady, S. (2014). Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (icadiwt 2014)* (pp. 232–238).

Kiss, I., Genge, B., Haller, P. & Sebestyén, G. (2014). Data clustering-based anomaly detection in industrial control systems. In *2014 ieee 10th international conference on intelligent computer communication and processing (iccp)* (pp. 275–281).

Kulshrestha, R. (2020, Sep). *Latent dirichlet allocation(lda).* Towards Data Science. Retrieved from `https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2`

Likas, A., Vlassis, N. & Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern recognition*, *36*(2), 451–461.

Lim, C., Singh, N. & Yajnik, S. (2008). A log mining approach to failure analysis of enterprise telephony systems. In *2008 ieee international conference on dependable systems and networks with ftcs and dcc (dsn)* (pp. 398–403).

Linders, B. (2021, Jul). *Artificial intelligence for it operations: an overview.* InfoQ. Retrieved from `https://www.infoq.com/news/2021/07/AI-IT-operations/`

Liu, J. & Wang, Y.-Q. (2011). Web log data mining based on association rule. In *2011 eighth international conference on fuzzy systems and knowledge discovery (fskd)* (Vol. 3, pp. 1855–1859).

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Market guide for aiops platforms. (n.d.). Retrieved from https://www.gartner.com/en/documents/3971186/market-guide-for-aiops-platforms,author={Gartner_Inc},year={2021}

Münz, G., Li, S. & Carle, G. (2007). Traffic anomaly detection using k-means clustering. In *Gi/itg workshop mmbnet* (pp. 13–14).

Nedelkoski, S., Cardoso, J. & Kao, O. (2019). Anomaly detection and classification using distributed tracing and deep learning. In *2019 19th ieee/acm international symposium on cluster, cloud and grid computing (ccgrid)* (p. 241-250). doi: 10.1109/CCGRID.2019.00038

Pezzotti, N., Lelieveldt, B. P., Van Der Maaten, L., Höllt, T., Eisemann, E. & Vilanova, A. (2016). Approximated and user steerable tsne for progressive visual analytics. *IEEE transactions on visualization and computer graphics*, *23*(7), 1739–1752.

Prado, K. S. d. (2019, Jun). *How dbscan works and why should we use it?* Towards Data Science. Retrieved from https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*.

Sahoo, R. K., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., Ma, S., ... Sivasubramaniam, A. (2003). Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the ninth acm sigkdd international conference on knowledge discovery and data mining* (pp. 426–435).

Singh, R. (2019, Oct). *The most important elements of aiops - dzone ai.* DZone. Retrieved from https://dzone.com/articles/the-most-important-elements-of-aiops

Tianguo, X. (2021). *Enterprise aiops deployment white paper.* Retrieved from https://max.book118.com/html/2018/0804/7010054015001142.shtm

Vaarandi, R. (2003). A data clustering algorithm for mining patterns from event logs. In *Proceedings of the 3rd ieee workshop on ip operations management (ipom 2003) (ieee cat. no.03ex764)* (p. 119-126). doi: 10.1109/IPOM.2003.1251233

Visualizing dbscan clustering. (n.d.). Retrieved from https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Wang, H. & Zhang, H. (2020). Aiops prediction for hard drive failures based on stacking ensemble model. In *2020 10th annual computing and communication workshop and conference (ccwc)* (p. 0417-0423). doi: 10.1109/CCWC47524.2020.9031232

Wang, T., Wei, J., Zhang, W., Zhong, H. & Huang, T. (2014). Workload-aware anomaly detection for web applications. *Journal of Systems and Software*, *89*, 19–32.

What is aiops? (2021, Oct). moogsoft. Retrieved from https://www.moogsoft.com/resources/aiops/guide/everything-aiops/

What is a log file? (2021). Retrieved from `https://www.sumologic.com/glossary/log-file/`

Winding, R., Wright, T. & Chapple, M. (2006). System anomaly detection: Mining firewall logs. In *2006 securecomm and workshops* (pp. 1–5).

Xu, D. D. & Wu, S. B. (2014). An improved tfidf algorithm in text classification. In *Applied mechanics and materials* (Vol. 651, pp. 2258–2261).

Xu, W. (2010). *System problem detection by mining console logs*. University of California, Berkeley.

Xu, W., Huang, L., Fox, A., Patterson, D. & Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the acm sigops 22nd symposium on operating systems principles* (pp. 117–132).

Yamanishi, K. & Maruyama, Y. (2005). Dynamic syslog mining for network failure monitoring. In *Proceedings of the eleventh acm sigkdd international conference on knowledge discovery in data mining* (pp. 499–508).

Zhao, Y., jun Wei, M. & Wang, J. (2013). Realization of intrusion detection system based on the improved data mining technology. In *2013 8th international conference on computer science & education* (pp. 982–987).

Zhu, W., Webb, Z. T., Mao, K. & Romagnoli, J. (2019). A deep learning approach for process data visualization using t-distributed stochastic neighbor embedding. *Industrial & Engineering Chemistry Research*, *58*(22), 9564–9575.

Çelik, M., Dadaşer-Çelik, F. & Dokuz, A. (2011). Anomaly detection in temperature data using dbscan algorithm. In *2011 international symposium on innovations in intelligent systems and applications* (p. 91-95). doi: 10.1109/INISTA.2011.5946052