REAL-TIME ANOMALY DETECTION USING CLUSTERING IN BIG DATA TECHNOLOGIES

RIYAZ AHAMED ARIYALURAN HABEEB

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

REAL-TIME ANOMALY DETECTION USING CLUSTERING IN BIG DATA TECHNOLOGIES

RIYAZ AHAMED ARIYALURAN HABEEB

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Riyaz Ahamed Ariyaluran Habeeb

Registration/Matric No.: WHA140019

Name of Degree: Doctor of Philosophy

Title of Thesis: Real-Time Anomaly Detection using Clustering in Big Data Technologies

Field of Study: Computer Science

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:
Designation:

REAL-TIME ANOMALY DETECTION USING CLUSTERING IN BIG DATA TECHNOLOGIES

ABSTRACT

The advent of connected devices and omnipresence of Internet have paved way for intruders to attack networks, which leads to cyber-attack, financial loss, information theft and cyber war. Hence, network security analytics has become an important area of concern and has gained intensive attention among researchers, off late, specifically in the domain of anomaly detection in network, which is considered crucial for network security. However, critical reviews have identified that the existing approaches are inefficient in processing data to detect anomalies due to the amassment of massive volumes of data through the connected devices. Therefore, it is crucial to propose a framework that effectively handles real time big data processing and detect anomalies in networks. In this regard, this research attempted to address the issue of accuracy in anomalies detection in real time. To begin with, the existing state-of-the-art techniques related to anomaly detection, real-time big data technologies and machine learning algorithms have been critically reviewed to identify the problems. Subsequently, comparative analysis to further establish the problems has been carried out via utilization of various existing algorithms which were then validated using three openly available datasets. Based on the outcome of the analysis, this research proposed a novel framework namely real-time anomaly detection based on big data technologies (RTADBDT), along with supporting implementation algorithms. The framework comprises of BroIDS, Flume, Kafka, Spark Streaming, Spark MLlib, Matplot and HBase. The BroIDS processes the existing datasets and generates various log files such as HTTP which is used in this research while Flume component reads and tracks the incoming packet data blocks. Kafka comprises repository of messages, categorized into

different topics, with each category further divided into numerous partitions comprising of well-arranged and absolute sequence of messages. Meanwhile, Spark Streaming effectively provides illustrious abstraction known as DStream, signifying an uninterrupted stream of data whereas Spark MLlib leverages algorithmic optimizations of MLlib and applies them in the proposed algorithms. Ultimately, the processed data has been visualised by using Matplot and stored via HBase. The proposed framework was validated to substantiate its efficacy particularly in terms of accuracy, memory consumption and execution time by performing critical comparative analysis using internal, external and statistical techniques. The performance of the proposed framework was assessed using mathematical expressions derived in this research and also by conducting comparative analysis. All the analysis has proven that the proposed framework's technique has outperformed other existing techniques in terms of accuracy, memory consumption and execution time. The significance of this research can be attributed to wide spectrum in the body of knowledge, with the proposed framework serve as a backbone in real-time anomaly detection with increased accuracy, minimised memory consumption and shortened execution time. Furthermore, when implemented, this framework shall enable an organization to instantly detect anomaly in real-time while having potential for a more effective fault tolerance and scalability.

Keywords: Anomaly detection, Real-time big data processing, Clustering.

PENGESANAN ANOMALI MASA NYATA MENGGUNAKAN PENGKLUSTERAN DALAM TEKNOLOGI DATA RAYA

ABSTRAK

Kemunculan peranti bersambung dan penggunaan Internet telah membuka jalan bagi penceroboh untuk menyerang rangkaian, yang membawa kepada serangan siber, kerugian kewangan, kecurian maklumat dan perang siber. Oleh itu, analisis keselamatan rangkaian telah menjadi fokus utama dan telah mendapat perhatian intensif di kalangan para penyelidik kebelakangan ini, khususnya dalam pengesanan anomali dalam rangkaian, yang dianggap penting untuk keselamatan rangkaian. Walaubagaimanapun, ulasan-ulasan kritikal telah mengenal pasti bahawa pendekatan-pendekatan yang sedia ada tidak cekap dalam memproses data untuk mengesan anomali disebabkan pengumpulan data secara besar-besaran melalui peranti-peranti yang berhubung antara satu sama lain. Oleh itu, adalah penting untuk mencadangkan satu rangka kerja yang berkesan mengendalikan pemprosesan data besar dalam masa nyata serta mengesan anomali dalam rangkaian. Maka, kajian ini cuba menangani isu ketepatan pengesanan anomali dalam masa nyata. Sebagai permulaan, teknik sedia ada yang berkaitan dengan pengesanan anomali, teknologi data besar dalam masa nyata dan algoritma pembelajaran mesin telah dikaji secara kritikal untuk mengenal pasti masalah. Seterusnya, analisis perbandingan untuk menyelesaikan masalah telah dijalankan melalui penggunaan pelbagai algoritma sedia ada yang kemudiannya disahkan menggunakan tiga himpunan data yang diperolehi secara terbuka. Berdasarkan hasil analisis, penyelidikan ini mencadangkan rangka kerja baru iaitu pengesanan anomali masa nyata berdasarkan teknologi data besar (RTADBDT) dan algoritma untuk pelaksanaan model ke dalam rangka kerja. Kerangka ini terdiri daripada BroIDS, Flume, Kafka, Spark Streaming, Spark MLlib, Matplot dan HBase. BroIDS memproses himpunan

data yang sedia ada dan menjana pelbagai fail log seperti HTTP yang digunakan dalam penyelidikan ini manakala komponen Flume membaca dan menjejaki blok paket data yang masuk. Selain itu, Kafka terdiri daripada repositori mesej, dikategorikan ke dalam topik yang berbeza, dengan setiap kategori selanjutnya dibahagikan kepada banyak bahagian yang terdiri daripada urutan mesej yang disusun dengan sempurna dan mutlak. Sementara itu, Spark Streaming berkesan menyediakan abstraksi yang terkenal dikenali sebagai DStream, menandakan aliran data yang tidak terganggu manakala Spark MLlib menggunakan pengoptimuman algoritma MLlib dan menggunakannya dalam algoritma yang dicadangkan. Kemudian, data yang diproses telah digambarkan dengan menggunakan Matplot dan disimpan melalui HBase. Secara keseluruhan, rangka kerja dicadangkan dan disahkan untuk membuktikan keberkesanan khususnya dari segi ketepatan, penggunaan memori dan masa pelaksanaan dengan melakukan analisis perbandingan secara kritikal menggunakan teknik dalaman, luaran dan statistik. Prestasi rangka kerja yang dicadangkan dinilai pula dengan menggunakan ungkapan matematik yang dihasilkan melalui kajian ini disamping melakukan analisis perbandingan. Semua analisis telah membuktikan bahawa teknik rangka kerja yang dicadangkan telah mengatasi teknik sedia ada yang lain dari segi ketepatan, penggunaan ingatan dan masa pelaksanaan. Kepentingan kajian ini boleh dikaitkan dengan spektrum yang luas dalam bidang pengetahuan relevan, dengan kerangka yang dicadangkan berfungsi sebagai tulang belakang dalam pengesanan anomali masa nyata dengan peningkatan ketepatan, penggunaan memori dan masa pelaksanaan yang lebih rendah. Lebih-lebih lagi, apabila dilaksanakan, rangka kerja ini akan membolehkan sesebuah organisasi untuk mengesan anomali dengan serta-merta dalam masa nyata sementara berpotensi untuk lebih keberkesanan dalam toleransi sesar dan pengubahan skala.

Kunci kata: Pengesanan anomali, Pemprosesan data raya masa nyata, Pengklusteran.

ACKNOWLEDGEMENTS

All praise to the Almighty Allah, the All-Compassionate, the All-Merciful, for His countless and endless blessings which enable me to complete this thesis. Throughout the course of this research, I was fortunate enough to meet wonderful people and receive a great deal of support from them while pursuing my PhD at University of Malaya, Malaysia. Without them, it would not be possible to finish my dissertation.

I will start with deepest gratitude to my supervisors, Prof. Dr. Abdullah Gani and Dr. Fariza Nasaruddin for their commitment and continuous guidance from the early stages to the completion of this thesis. Meetings and discussions with them enable me to identify not only my thesis area, but also my strengths which I would hold on to the next chapter of my career.

I would also like to express my profound appreciation to Dr. Anjum Naveed, Dr. Ejaz Ahmed, Dr. Ibrahim Abaker, Dr. Hasan Jamil and Mr. AbdelMuttlib Ibrahim for their supports and encouragement during my thesis completion. I am also deeply indebted and grateful to Mr. Ahamed Rasmi, Mr. Mohamed Ahzam Amanullah and Mr. Abdul Salam Nainar for their assistance.

No words would be able to convey my true feelings and gratitude to my beloved family whom has been a source of inspiration for me. This appreciation is especially to my dear parents who always pray for me and provides me comfort during my difficult times. Also, words cannot express how grateful I am to my dearest brother Imtiyas for his great support. He had always been a wonderful advisor for me.

To my beloved wife, Zurul Aisya, who not only very caring and tolerance, but also instrumental in assisting me with the thesis. I would not have been able to go through this without her support. Special hugs and kisses to my son Rais, who not only gives me

inspiration but also motivation to finish this thesis so that I can spend more time to see his adorable antics. It is also not an exaggeration to be very grateful and indebted to my siblings' in-law for their tremendous support and prayers.

Finally, thank you to all the members and support staff at the Centre for Mobile Cloud Computing, Wisma R & D and FSKTM for lending their support and resources.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ACM	Association for Computing Machinery
API	Application Programming Interface
ARI	Adjusted Rand Index
AWS	Amazon Web Services
BN	Bayesian Networks
CANN	Cluster Center and Nearest Neighbour
DBSCAN	Density-based Spatial Clustering of Application with Noise
DDOS	Distributed Denial-of-Service
DNS	Domain Name System
DOS	Denial-of-Service
ETL	Extract, Transform, Load
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Application with Noise
HDFS	Hadoop Distributed File System
HTM	Hierarchical Temporal Memory
HTTP	HyperText Transfer Protocol
IBM	International Business Machines
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IOT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KCL	Kinesis Client Library

KNN	K-Nearest Neighbours
LOF	Local Outlier Factor
LTE	Long-Term Evolution
MAC	Media Access Control
MIME	Multi-Purpose Internet Mail Extensions
ML	Machine Learning
NMI	Normalized Mutual Info
NN	Nearest Neighbours
OC-SVM	One Class Support Vector Machine
PAAS	Platform as a service
PCA	Principal Component Analysis
RAM	Random Access Memory
RDD	Resilient Distributed Dataset
RF	Random Forest
RI	Rand Index
RTADBDT	Real-Time Anomaly Detection based on Big Data Technologies
R2L	Root to Local attacks
SAMOA	Scalable Advanced Massive Online Analysis
SIEM	Security Information and Event Management
SMTP	Simple Mail Transfer Protocol
SOM	Self-Organizing Map
SSWLOFCC	Streaming Sliding Window LOF Coreset Clustering
SVM	Support Vector Machine
TCP/UDP	Transmission Control Protocol/User Datagram Protocol
U2R	User to Root attack

WEKA	Waikato Environment for Knowledge Analysis
XML	eXtensible Markup Language
YARN	Yet Another Resource Negotiator

CHAPTER 1: INTRODUCTION

Every year, the usage of connected devices increases tremendously, which contributes to the growth of real-time network data with high velocity and huge volume. Conversely, threats on networks in the form of intrusions become inevitable, which needs to be discovered in real-time. As the first step of defence to detect threat, it becomes crucial to identify anomaly in the network data.

This chapter presents an overview of the research carried out in this thesis. This research has encompassed the fundamentals of anomaly detection and real-time big data technologies, to enable the readers to understand the domain of our research. The chapter has been organised as follows: Section 1.1 introduces anomaly detection, real-time big data technologies and machine learning. This followed by the motivation of the research in section 1.2. The research gap presented in section 1.3 which details the issues associated with threat detection, utilization of resources, which leads to the synopsis of problem statement. The aim and objectives of this research are presented in section 1.4, whereas the overview of our research methodology is presented in section 1.5. Lastly, the section 1.6 summarizes the organization of the chapters in this research.

1.1 Domain Background

In this section, an overview of anomaly detection, real-time big data technologies and machine learning are presented to offer the fundamental knowledge on the research domain.

Figure 1.1 illustrates bottom up sequence of real-time big data processing for anomaly detection, where various smart devices are communicated via network technologies. Such devices generate a lot of sensor data, which are stored in cloud and other storages devices. These stored datasets collected from sensors devices are processed with big data processing technologies, such as, Hadoop, Spark, Apache storm, and the results are used for analysis

and anomaly detections using machine learning algorithms.

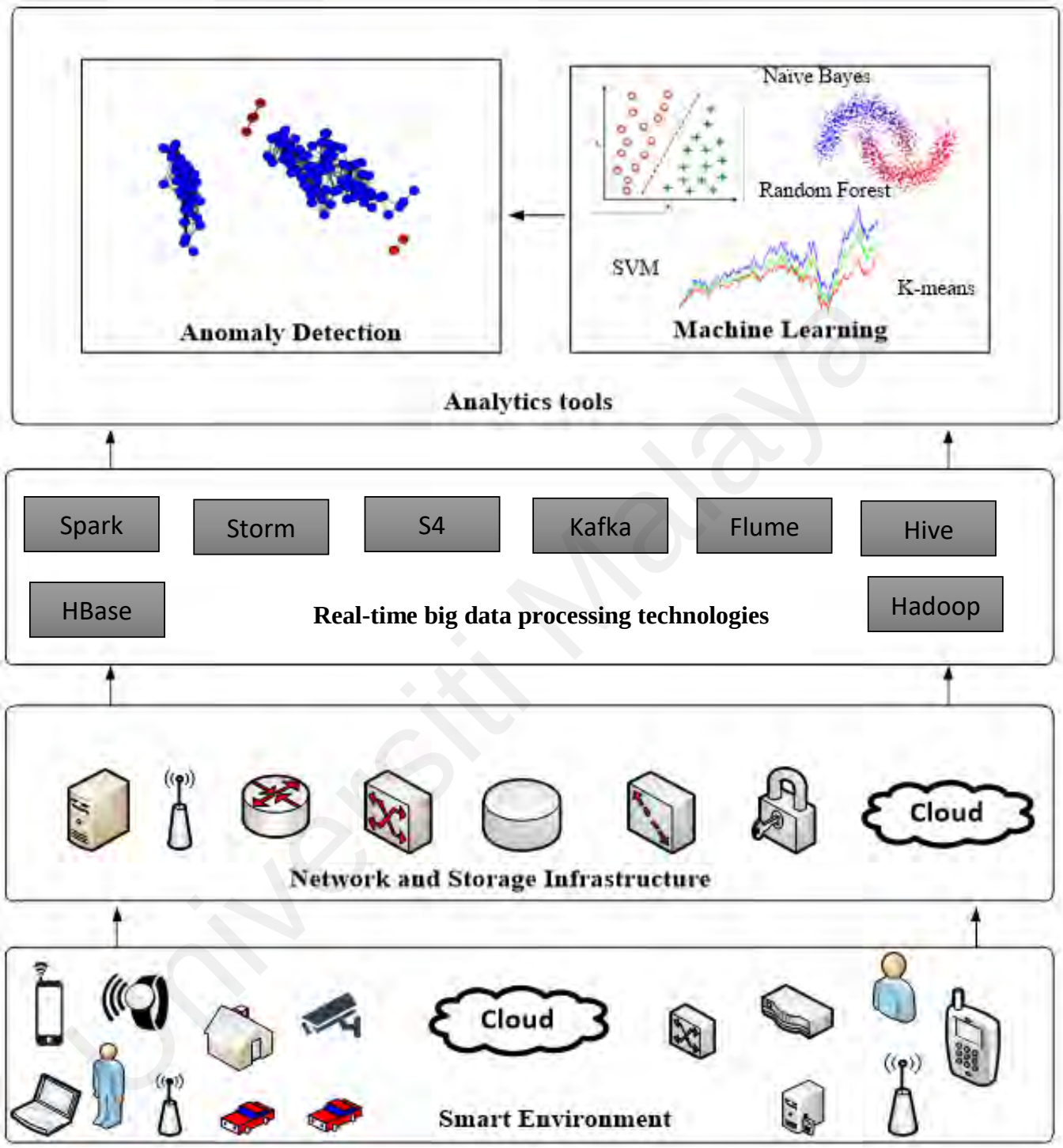

Figure 1.1: The flow control of different stages in big data processing and anomaly detection

1.1.1 Anomaly detection

Anomaly detection is an important problem that has been researched within diverse research areas and application domains (Buczak & Guven, 2016) (Chandola, Banerjee, & Kumar, 2009). Anomaly detection refers to the process of finding patterns in data that do not conform to expected behaviour. These non-conforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different applications domains. Anomaly detections find extensive use in a wide variety of applications such as, networking monitoring, healthcare, smart devices, smart cities, Internet of things, fraud detection, cloud, and much more. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination (Chandola et al., 2009). In particular, network monitoring helps to detect threat from various network infrastructure elements, such as MAC spoofing, IP Spoofing, TCP/UDP fanout, Duplicate IP, Duplicate MAC, Virus detection, bandwidth anomaly detection and connection rate detection. Additionally, anomaly detection helps to track the profiles of normal day to day activities of every system, application, or network (Buczak & Guven, 2016).

1.1.2 Real-time Big data processing

Real-time big data processing is critical than any other processing application, because, it is essential for the uninterrupted monitoring of events, messages, processes in the network infrastructure (Cloud Strategy Partners, 2015). Furthermore, fast data is generated for network monitoring from hardware and software. For example, the log file that can be rapidly changing in-memory data set, however, in fast data, the data dynamically changes in certain time intervals varied between seconds and milliseconds. The huge quantity of data that arrive continuously to the pipeline can be in any format such as, structured, unstructured, and semi-structured. These data contain the detailed information about the

messages and events. Streamed data are positioned in the big data analytics for p(Caliński & Harabasz, 1974)rocessing, and then big data analytics will help to make the analysis and decision for further process. Adoption of streaming architecture will guarantee the efficient and seamless communication between the sensing devices and network (Hashem et al., 2016). A Large amount of real-time data can be processed with the following tools: Storm, Splunk, S4, SAP Hana, Spark (Yaqoob et al., 2016). With connected devices continuously collecting, processing, and storing massive amounts of data, it is evident that we are living in the era of big data. (Hashem et al., 2015) have defined a set of techniques and technologies that require new forms of integration to uncover large hidden values from the big data that are diverse, complex and of a massive scale. i. Furthermore, collecting and processing big data provide new opportunities to use the machine learning algorithms along with dynamic statistical analysis. These contribute to a more reliable real-time solution to network anomaly detection problems (Prosak, Gangopadhyay, & Garg, 2019).

1.1.3 Machine Learning

The main objective of the machine learning is to allow a system to learn from the past or present and use the knowledge to make predictions or decisions regarding unknown future events(Landset, Khoshgoftaar, Richter, & Hasanin, 2015). Machine learning can be applied to different industries, like banking, autonomous car, manufacturing, retail industry, marketing, networking, and general science, including chemistry, physics, medicine, bioscience, pharmaceutical, insurance, energy, and sustainability.

Several machine learning algorithms have been proposed and used for mining meaningful information from the data through preparation and validation using categorized datasets. These algorithms are classified into two major categories, such as supervised and unsupervised. In real-time applications, the machine learning algorithm needs to analyse a continuous sequence of data occurring in real-time. When compared to batch processing,

the entire dataset is not available in runtime. Furthermore, real-time application requires to process data in sequential form as they arrive, and make decisions online (Ahmad, Lavin, Purdy, & Agha, 2017). Manipulating and modifying the existing machine learning system will satisfy the market needs to increase the conservation of energy and increase the computational cost (Al-Jarrah, Yoo, Muhaidat, Karagiannidis, & Taha, 2015).

Traffic among machines has become an essential portion of today's network environment and will escalate even more in the near future. It is expected to produce exceptional traffic patterns that will challenge network administrators to learn and track the threat in the network. Existing machine learning techniques are incompatible for addressing big data classification problems, and are incapable of handling unstructured data, which are essential to produce high accuracy for high-velocity data, and ineffective for multiple learning tasks and also for computational efficiency. Computational complexity has exponentially risen in high dimensional data that fail to fulfil current needs. In this context, various machine learning algorithms, such as Nearest Neighbours, Bayesian Networks, Support Vector Machines, Decision Trees, Random Forest, Ant Colony Optimization, Fuzzy logic, Principal Component Analysis are discussed in this research.

The above sections had discussed the importance of anomaly detection, real-time big data processing, machine learning and noted most used algorithms for anomaly detection.

1.2 Research motivation

This section describes the motivation for anomaly detection through real-time big data processing technologies.

According to report by, Cisco ("Cisco VNI Forecast and Methodology, 2015-2020," 2016) forecasted that, 2.3 Zettabytes of Internet protocol(IP) traffic would go across the Internet in 2020, which will be 879 Exabyte more from 2015. This leads to a lapse in the existing security analytics to detect the threats in real time. Furthermore, Cisco also

reported that 71 percent of total IP traffic in 2020 is expected to be generated from non-PC devices (smart devices) such as, tablets, smart watches, smartphones, smart bands, video game consoles, television set-top devices, smart key chains, smart bulbs, smart security cameras, smart TVs, and smart locks (Kerner, 2016). This leads to a huge volume of data to be analysed in real time with high velocity and more varieties, which fulfil the characteristics of big data, which are volume, velocity and variety.

Meanwhile, non-PC devices pose a huge security threat, if they are not monitored in real time. In October 2016, large organizations such as, CNN, Twitter, Reddit, The Guardian, and Netflix in US and Europe were massively attacked via smart home devices (Woolf, 2016). In addition, new threats are expected to emerge in 2021 as hackers find new ways to attack smart devices and protocols (Jones, 2016).

According to (Gartner, 2018) the expenditures towards worldwide security spending might reach an amount of 124 Billion dollar by the end of 2019. Universally, organizations are expected to be extra conscious of the security risk, due to the inefficient and inadequate protection against attacks by the existing technologies. The U.S. Federal Cybersecurity Market Forecast has been estimated to reach 22 Billion dollar by 2022, which will constitute a steady compound annual growth rate of 4.4 percentage (Ohio, 2017). Further, this will drive the new norms of machine learning solutions to replace Security Information and Event Management (SIEM) of traditional anti-virus within the next five years (helpnetsecurity, 2017). It has also been highlighted that in new generation applications, data stream processing has emerged as one of the potential research areas, in which, data continuously flows into the processing site.

1.3 Statement of the problem

This section presents the statement of the problem associated with the domain of this research. Detecting anomalies in real-time is important to ensure network security. Practically, it is difficult to detect threats in real-time due to the limitations of existing approaches. The limitations can be attributed to many factors, which can be regarded as 'causes' of problem, which eventually restrict the efficacy of real-time anomaly detection that can be termed as 'effect'. Figure 1.2 illustrates the causes and events and their association towards statement of the problem. It is called causal-effect relationship approach to formulate the research problem.

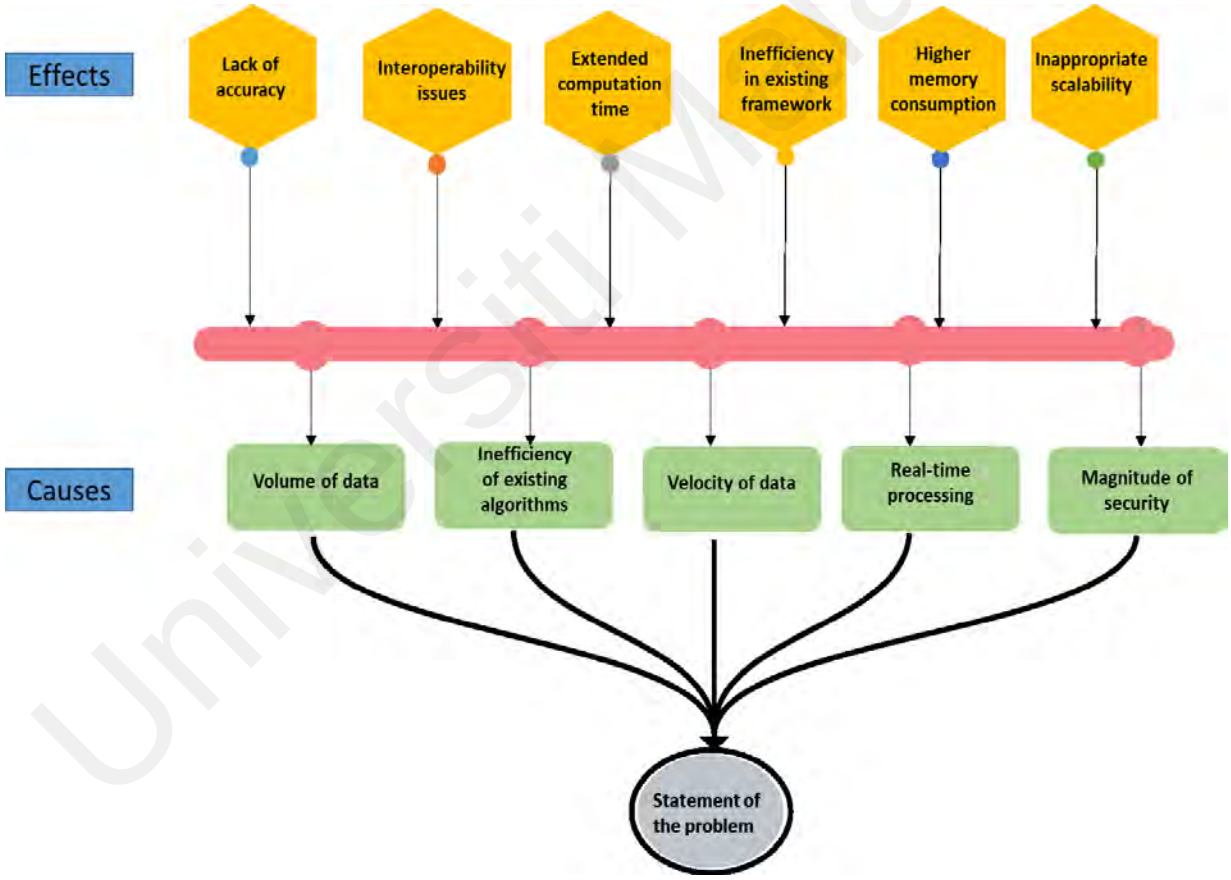

Figure 1.2: Approach to statement of the problem

Real-time analysis of anomalies in an incoming data streams is challenged by the magnitude of threat, the volume and velocity of the data. Given these challenges, current

techniques for accurate anomaly detection are often characterised by exorbitant execution time and a bulk memory usage which make any detections insufficient with the context of real-time intervention. Additionally, storing, processing and analysing the incoming data, able to explain the threat and outcome would act as a feedback of the future detections, although offer no value in on-the-spot remedial actions. This is equally challenged by storage, fault tolerance and scalability issues. Apart from these, the existing approaches suffer from interoperability problems which deter the efficiency of existing framework.

The massive amounts of data generated in real time have inhibited the performances of network analysts in terms of scaling the abundant volume of data. Hence it is crucial to produce network security analytics performance reports in the real-time, and not just to be generated from existing monthly and weekly log data. Furthermore, increasing numbers of new types of threat have become common in networks, every day, but the existing monitoring tools have become obsolete to detect those threat due to the huge volume, velocity, variety, and veracity of data received for analysis. It is crucial to immediately process the data collected to detect any potential threat in the network, however, the existing traditional monitoring tools are incapable of handling big data, and therefore struggle to continuously monitor network infrastructure and detect the anomaly behaviour and threats (Raguseo, 2018).

In conclusion, existing anomaly detection techniques for real-time analysis is still at the premature stage with numerous of flaws.

1.4 Statement of Objectives

This research aimed to enhance framework for anomaly detection by means of real-time big data technologies and machine learning algorithm. The research objectives listed below provide the direction of this research:

1. To review the state-of-the-art anomaly detection techniques and real-time big data technologies with respect to the performance issues of real-time anomaly detection.
2. To investigate various anomaly detection algorithms to evaluate accuracy, execution time, and memory consumption in real-time anomaly detection
3. To propose and implement a new framework for improving accuracy, minimizing memory consumption and shorten execution time.
4. To evaluate the performance of the proposed framework by comparing and validating it with other existing techniques

The principal objective of this research is developing a real-time big data processing for anomaly detection using clustering algorithm, mainly aimed at enhancing the detection accuracy and minimizing the execution time and memory consumption.

1.5 Proposed methodology

This research comprises of four stages to enable the accomplishment of the research objectives (see section 1.4).

The four objectives listed above will be achieved through the four stages. The first stage reviews the past researches in the three domains such as, anomaly detection, real-time big data technologies and machine learning, this will help us to ascertain the research gap and identify the potential problems. This study will meticulously analyse a number of present technologies and adoption of real-time big data applications. These existing technologies will be classified as corresponding solutions. Our initial investigation has revealed that in the area of real time anomaly detection there are still a number of issues remain unaddressed (see Chapter 2 for brief discussion about these challenges).

The next stage of research investigates the research problem by means of simulation, which leads to the development of an application for real-time anomaly detection. Further-

more, the accuracy of anomaly detection by the existing applications will be tested along with performance factors such execution time and memory consumption, which will lead to the identification of the research problem.

In the third stage a composite algorithms based on clustering will be developed and integrated with big data technologies for improving the detection of anomalies.

The fourth stage focuses on the validation of the proposed framework and will be equated with current applications. Additionally, a various statistical analysis will be carried out to establish the outcome. Table 1.1 outlines the proposed research methodology.

Table 1.1: Proposed research methodology

OBJECTIVE	To review the state-of-the-art anomaly detection techniques and real-time big data technologies with respect to the performance issues of real-time anomaly detection.	To investigate various anomaly detection algorithms to evaluate accuracy, execution time, and memory consumption in real-time anomaly detection	To propose and implement a new framework for improving accuracy, minimizing memory consumption and shorten execution time.	To evaluate the performance of a proposed framework by comparing and validating it with other existing techniques.
ACTIVITIES	• Analyzing & synthesizing the merits and flaws of present techniques • Conducting a qualitative comparison. • Grouping the relevant works in the form of taxonomy. • Discovering the open issues	• Design experiment setup • Selecting and validating suitable dataset for the experiment • Using three different clustering algorithms performing experiment to detect the accuracy of anomaly, memory usage and execution time for real time logs • Critically analysing the results of the experiment	• Designing composite clustering algorithm • Developing a big data framework for real-time anomaly detection with enhanced accuracy, and minimized memory consumption, execution time • Implementing the proposed algorithm.	• Discovering the performance measuring metrics • Analysing with the outcome of existing framework. • Use appropriate statistical analysis to signify the results. • Validating - comparing with existing real-time anomaly detection applications. • Validating mathematical expression and experiment results.
OUTPUT	**Identify potential problem**	**Problem establishment**	**Framework with clustering algorithm and real-time big data technologies**	**Verification and validation of Solution**

1.6 Outline of Thesis

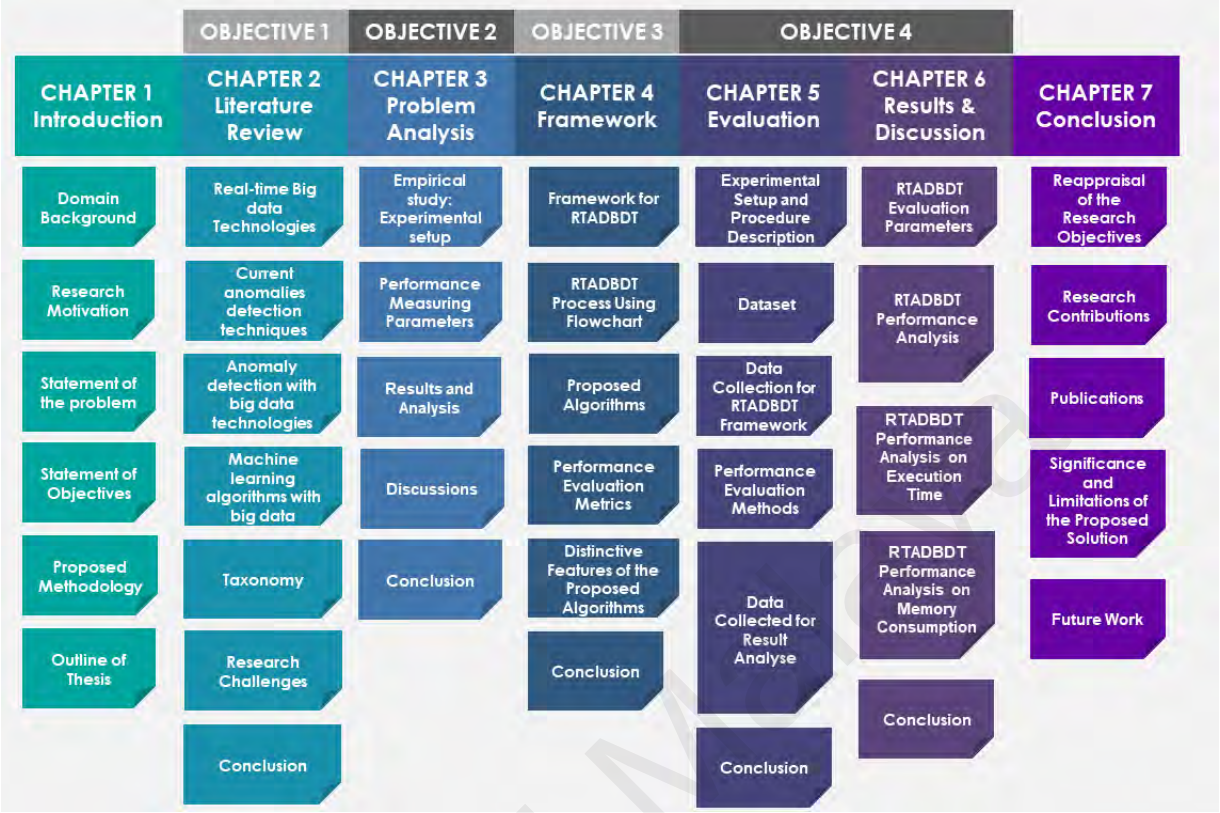

Figure 1.3: Graphical outline of thesis

The outline of this thesis is graphically presented in Figure 1.3. This thesis presents an in-depth investigation on real-time detection using big data technology, entitled "Real-time anomaly detection using clustering in big data technologies". Apart from this present chapter, this thesis comprises of 6 more chapters, each one presenting holistic discussion about the different parts of research.

Chapter 2 focuses on the review of existing literature related to detection of anomalies in real-time big data technologies. This review enlightens our knowledge about the existing anomaly detection techniques using big data technologies, which will subsequently help us to identify the crucial problems of the present applications, particularly in terms of accuracy of detection and significant performance aspects such as, utilization of memory and computational time. Apart from this, the chapter also emphasizes a number

of unsupervised clustering algorithm based anomaly detection applications; this study has also proposed a taxonomy to categorize the existing literature based on big data technology, anomaly detection, machine learning techniques, modes, data, and application. Ultimately, the research challenges and recommendations for future researchers have also been presented.

Chapter 3 presents the analytical report about the impact of clustering algorithms in anomaly detection specifically related to the precision of detection, utilization of memory, and computational time as against the traditional methods of real-time anomaly detection. The research problem, specifics of experimental setup, performance metrics, experimental bounds, datasets, and the outcome of evaluation are also presented in this chapter.

Subsequently, in **Chapter 4** , a novel framework for composite clustering algorithm for detection of anomaly using real-time big data technologies has been proposed. The proposed framework is aimed at addressing the pertaining issues such as inadequate detection accuracy rate, consumption of huge volume of memory and expansive computational cost.

These pseudo-code of clustering algorithms are presented in this chapter. Furthermore, the unique features of proposed hybrid algorithm have been presented. As the proposed framework is developed by hybrid big data technologies such as, Flume, Kakfa, Spark Streaming, Spark MLlib, and HBase. The hybridization facilitates to accomplish performance goals and enhance real-time anomaly detection.

The **Chapter 5** details the processing of the collected data for the evaluation of the proposed solutions. The tools used for validating the results have also been explained, followed by the details of performance parameters, experimental setup and the statistical methods used for validating the accuracy of the data collected from the model and proposed solutions.

The effectiveness of the proposed solutions has been presented in **Chapter 6** by

analyzing the collected results, which have been presented in Chapter 5. This chapter presents the various features of processing, accuracy rate anomaly detection in real-time, utilization of memory, and computational time. Furthermore, details about the evaluation of the mathematical expression with the experiment results have also been presented, along with comparative analysis of the performance parameter of the proposed framework as against the existing solution in terms of accuracy, memory consumption and execution time.

Chapter 7 presents the major contributions of the thesis by explaining the accomplishment of the research objectives. The findings of the research work has been summarized, and the importance of the proposed solutions has also been highlighted. The presentation of research limitations and directions for future works concludes this chapter and the thesis on the whole.

CHAPTER 2: REAL-TIME ANOMALY DETECTION BASED ON BIG DATA TECHNOLOGIES

This chapter aims to identify the most significant shortcomings of Real-Time Anomaly Detection based on Big Data Technologies (RTADBDT) and likewise, examined the problem that was highlighted in Chapter 1. To achieve this, this research has have investigated the recent research efforts focused on RTADBDT. This research has analysed several problems hindering the adoption of RTADBDT and proposed corresponding solutions by devising a taxonomy. The roots of RTADBDT and taxonomies as techniques, application, anomalies, modes, data, big data processing, and the record categories have been analysed. The similarities and differences among proposed solutions have been analysed in terms of their advantages and disadvantages. This research has also compared the literature based on objectives. Further, this chapter advocates that problems that stem from the intrinsic characteristics of RTADBDT, by identifying several new principles. Finally, several research challenges have been presented to be investigated in this research.

This chapter comprises of the following sub-sections: Section 2.1 carries the details of contemporary studies related to real-time big data technologies followed by a tabulation of comparison on the important features, advantages and disadvantages of those studies. Section 2.2 presents the limitations of the existing anomalies detection techniques and pros and cons of the most popular clustering algorithms. Details of the investigations about diverse big data technologies employed for detecting anomalies have been presented in section 2.3. Furthermore the significance of machine learning and big data technologies has been presented in section 2.4. Critical evaluation of real time big data processing for detecting anomalies by means of machine learning algorithms has been presented in section 2.5 along with their limitations. Taxonomy of real-time big data processing

technologies for anomaly detection has been presented in section 2.6. Next, the identified evaluation metrics for determining the precision of clustering and system performance of the real-time big data processing and anomaly detection have been presented in section 2.7. Moreover, the research challenges associated with real-time anomaly detection have been presented in section 2.8. Ultimately, the concluding observations have been presented in section 2.9.

2.1 Real-time Big Data Processing Technologies

Some of the contemporary real-time big data technologies employed for various anomaly detection have been critically analysed in this sub section. Generally, several real-time big data technologies have been employed to collect, pre-process, analyse, and store different types of data generated from networks, not limited to Spark, Hadoop, Storm, Samza, Flink, Kinesis, and S4. The technological working mechanism of these applications have been discussed in the taxonomy section. However, this subsection mainly compares the salient features, pros and cons of above mentioned technologies.

2.1.1 Spark

Apache Spark, a potent processing structure comprises of a user friendly tool to effectively analyse different types of data. (Rettig, Khayati, CudrMauroux, & Pirkowski, 2015) have proposed a new approach to evaluate online anomaly detection with two metrics, using entropy and pearson correlation. Moreover, big data streaming components, such as Kafka queues and Spark Streaming have been used to assure the generality and scalability issues. Nonetheless, complex processes are involved in handling data and also huge time is consumed for even periodic batch processing.

(Fang, Liu, & Lei, 2016) have employed a streaming algorithm in Spark Streaming framework identifies the click requests of users, and restructures interactions between the

users and browsers. They have validated their proposed model with HTTP traffic data from mobile cellular network. Spark Streaming has accomplished the low latency and real-time identification on main requests. The integrated window in the Spark Streaming retains a state depending on the incoming data from the data stream. In addition, Resilient distributed dataset (RDD) generates a slide interval of window period, and each RDD comprises 10 minutes of traffic data. Ultimately, the Spark engine of the streaming algorithms process the RDDs. It is notable that, the above study did not measure performance parameters such as execution time and usage of memory.

2.1.2 Storm

Apache Storm is a stream processing model, which centres on exceedingly low latency and has been regarded as the suitable choice for workloads that need optimal real-time processing. This model is capable of dealing with huge volume of data and yield less latency outcomes as against other existing models.

(Celebi, Kingravi, & Vela, 2013) have proposed a real-time data analytics framework based on Apache Storm to monitor smart home applications. This framework is capable of rapidly processing the sensed and historical data and streams, making use of IoT devices and embedded computing systems at the edge of the IoT network. Furthermore, Apache Storm has been widely utilised to examine spatio-temporal data streams, and it also facilitates centralized and global level environmental management. The Storm cluster comprises of master and worker nodes, where the former executes the Nimbus daemons, in charge for allocating tasks to the worker node and executes a supervisor daemons. Nevertheless, the need of manual configuration for every IoT device to the Apache Storm is the disadvantage that makes it difficult to adopt the framework; moreover the above framework had been evaluated only against one category of dataset.

Off late, (Ficco, Pietrantuono, & Russo, 2018) have investigated the problems associated

with aging manifestations in the apache Storm. They have identified that the Storm generates persistent anomalous behaviourism that inhibits some topologies from functioning constantly as the result of internal resource management tool, which is inclined by the garbage collector and the memory assigned to worker processes.

Consequently, Storm is subjected to several issues in the long run not limited to memory utilization, response latency, load balancing and other aging issues.

2.1.3 Flink

Apache Flink is a stream processing framework capable of dealing with batch tasks, where the batches are just regarded as data streams with limited restrictions, and therefore considers batch processing as a subset of stream processing.

(Qadah, Mock, Alevizos, & Fuchs, 2018) have proposed a distributed online prediction system based on Apache Flink framework for analysing user-defined patterns over numerous substantial streams of movement activities. In addition, this approach is based on conjoining probabilistic event pattern prediction models on multiple predictor nodes with a distributed online learning protocol for constantly learning the parameters of a global prediction model and effectively sharing them among the predictors. Preliminary investigations reveal that distributed online predicition model perform effectively with Apache Flink; however, the outcomes of Apache Flink have to be evaluated with other existing applications and it is essential to consider other performance metrics for evaluation purposes.

(Rivetti, Busnel, & Gal, 2017) have proposed a real-time event based anomaly detection application for manufacturing sector using Apache Flink. The proposed application delivers job task instances into task manager processes, each holding configurable number of slots. Given n slots in a task manager, each has access to a $n\text{-}th$ of the task manager available memory, it mught possess any number of threads and tasks instances.

Precisely, it is too early to consider Flink for machine learning adoption. This is due

to very negligible algorithms are supported by flinkml, and no commercial software has utilise it for their platform. Additionally, only few open source community are working on the flinkml resulting in lack of support for the adopting Flink for machine learning applications.

2.1.4 Kinesis

Kinesis is a part of Amazon's service that deals with real-time processing of streaming data on the cloud. This module is intensely incorporated with other Amazon services by means of connectors, such as S3, Redshift, and DynamoDB, to constitute a complete Big Data architecture. Kinesis comprises of Kinesis Client Library (KCL), which enables to develop applications and employ stream data for dashboards, alerts, or even dynamic pricing (Jayanthi, Sumathi, & Sriperumbudur, 2016).

In this regard, Dong Yuan et.al. (2015) have proposed a framework by incorporating cloud services with a network of IoT devices. They have utilised amazon kinesis for receiving data from all the mobile access points. Kinesis has also been used to process those access point data and stored in the Amazon S3 for further analysis. However, the proposed framework has not been evaluated on any performance parameters to prove its efficiency (Yuan, Jin, Grundy, & Yang, 2015).

Additionally (Srikanth & Reddy, 2016) have highlighted latency issues and inefficiency of the framework in handling stragglers as major problems of Amazon Kinesis. Data arriving at inconsistent time interval is called stragglers. In this case, Amazon Kinesis cannot handle the stragglers issues which is considered one of the challenge in real-time analytics.

2.1.5 Samza

A distributed stream processing framework, Apache Samza is compactly integrated with the Apache Kafka messaging system. (Kleppmann & Kreps, 2015) have examined the design of Kafka and Samza and identified that it facilitates the development of composite applications by composing a small number of simple primitives - replicated logs and stream operators.

(Noghabi et al., 2017) have proposed a Samza based distributed system that processes real-time streams, along with pre-processing of entire data streams. Furthermore, they have evaluated the recovery time of the system from failures. In addition, the proposed system supports very large scale of data despite limited memory, by combining local disk storage, an efficient changelog and caching. Nevertheless, in terms of real-time detection, Samza faces reliability challenges.

Precisely, Samza is advantageous due to its low latency and recoverability as against other real-time technologies. Nevertheless, when machine learning is embedded into Samza architecture it experiences a huge drop in the processing time and gets hindered with portability issues. Furthermore, lack of interactive mode and higher memory consumption for processing makes it less efficient to be applied with real-time machine learning algorithms.

2.1.6 Apache S4

In case of processing event streams, the Java based Apache S4 is regarded as a general purpose, scalable, distributed platform. The S4 comprises of a smallest component known as processing element (PE), which operates on a data subset, or a partition of the entire data, depending on the design. Furthermore, the Apache S4 has been designed by combining MapReduce and the Actors model. All the nodes in the cluster are identical and lacks centralized control. On the other hand, load balancing is one of the major challenges with

S4 (Neumeyer, Robbins, Nair, & Kesari, 2010).

(Xhafa, Naranjo, & Caballé, 2015) have proposed and evaluated S4 for big data stream processing using global flight monitoring system. They have used cluster environment using 50-70 znodes under zookeper services. They have achieved very fast process time per flight. However, heterogeneity of computing znodes in the Apache S4 clusters were not suitable for processing due to high incoming data rate for real-time processing. Hence, windowing which is an important function to integrate machine learning for any analysis is not possible in Apache S4. Consequently, Apache S4 cannot be incorporated in any machine learning and big data framework.

2.1.7 Hadoop

(Cui & He, 2016) have analysed many other anomaly detection models, where machine learning was most widely used, whereas, growing number of network traffic restrict the systems, since they need to perform complex calculation. In addition, the authors have proposed a model to yield better performance in detecting anomaly using Hadoop, HDFS, MapReduce, cloud, and machine learning algorithms. In fact, real time input data streaming were not addressed.

At the same time, another study used big data technologies, such as Hadoop, Hive and Mahout to implement scalable quasi-real-time intrusion detection system to detect peer-to-peer botnet using random forests (Singh, Guntuku, Thakur, & Hota, 2014). In which, Hive platform provides the distributed environment for sniffing and processing network traces and extraction of dynamic network features. However, the challenges related to memory consumption and execution time were not addressed for performance evaluation.

Of late, (Bao et al., 2018) have proposed anomaly detection algorithms that trace sequence data and uses a probabilistic suffix tree to detect problems from console logs. It

used Hadoop ecosystem to process and analyse the console log data; however, the proposed algorithm involved batch processing and not on real-time. In addition, performance metrics such as, memory consumption and execution time were not addressed. Mostly, Hadoop was developed to support the batch processing applications. In some of the real-time applications, Hadoop are used to do batch processing separately on their architecture. Moreover, embedding two different technologies for real-time and batch processing in the same architecture raise some issues such as integration, higher memory consumption and adoptability issues. Therefore, Hadoop is not suitable for a framework in this study as it is not capable to assist both batch and real-time processing in our work. Table 2.1 shows the comparison of various recent real-time big data processing technologies features. Similarly, Table 2.2 detailed the advantages and disadvantages of real-time big data processing technologies.

Table 2.1: Comparison of the features of recent real-time big data processing technologies

Real-time Big Data technologies	Programing Languages	Abstraction	Execution Model	Associated ML tools	Machine Learning Compatibility	In memory processing	Fault tolerance	Resource Management
Spark	Scala, Python, Java, R	RDD	Batch, streaming	MLlib, Mahout, H2O	Classification, Regression, Clustering, Decision Trees.	Yes	Yes	YARN, Mesos, Stand-alone
Storm	Java, Ruby	Spouts, Bolts, Topologies	Streaming	SAMOA	Not supported for machine learning.	Yes	Yes	YARN, stand-alone
Flink	Java and Scala	Dataset	Batch, streaming	Flink-ML, SAMOA	Multiple linear regression, SVM, K-Nearest Neighbours, MinMax Scaler.	Yes	Yes	YARN, Mesos, stand-alone
Kinesis	Java and C++	Kinesis Client Library	Batch, streaming	SparkML, MxNet/TensorFlow	Linear Learner, Factorization Machines, Neural Topic Modelling Principal Component Analysis (PCA), DeepAR forecasting.	No	Yes	Identity and Access Management (IAM)
Samza	Java	topics, partitions, brokers, producer, consumers	Streaming	Mahout	Decision Tree, Adaptive model rule, Collaborative filtering, Association rule learning.	No	Yes	YARN
S4	Java	Processing Elements	Streaming	Mahout	Decision Tree, Adaptive model rule, Collaborative filtering, Association rule learning.	No	Yes	YARN,
Hadoop	Java, Python	MapReduce	Batch	Mahout	Logistic regression, Naive Bayes, Random Forest K-means, fuzzy, spectral clustering, principal component analysis, Chi squared, Association rule learning.	No	Yes	YARN

Table 2.2: Advantages and disadvantages of existing real-time big data processing technologies.

Technologies	Advantages	Disadvantages
Spark	• Scalable • High throughput • Both batch and streaming	• Share memory with different applications. • No separate file management. • Iterative processing.
Storm	• Easy to use • Scalable • Fault-tolerant	• No support for batch processing. • Less reliability, • No automatic load balancing
Flink	• Cross-platform • Highly Flexible • Both batch and streaming.	• Less APIs • Bit Slower • Only one data processing core component.
Kinesis	• Durability • Elasticity • Easier aggregation of data	• High Cost • Complicated process. • Scalability Issue.
Samza	• fault-tolerant • High-level abstractions • Delivery guarantee	• Less reliability • Less accuracy of recovery
S4	• Scalable • Fault-tolerant • Pluggable platform	• Lack of the dynamic load balancing • No centralized administration • Complex configuration
Hadoop	• Easy integration • Scalable • Batch processing	• No Caching • Slow processing speed • No Real-time Data Processing

In the above section real-time big data technologies used for various applications have been discussed. These technologies when incorporated with the machine learning algorithms have their own pros and cons in terms of: performance, inadequacy in detection, longer execution time, and higher memory consumption. Among the various real-time big data technologies discussed above, the Apache Spark and Flink have more potential to be included for real-time anomaly detection using big data technologies when compared to others. However, Spark Streaming is more convenient with numerous libraries to be embedded with the machine learning as compared to the Flink. Furthermore, Spark streaming also supports the unsupervised algorithm in their architecture.

2.2 Current anomalies detection techniques

This sub-section critically analyses various existing techniques used for detecting anomalies. A hybrid Support Vector Machine (SVM) and ant colony network were designed to produce a high performance intrusion detection system, which contains parallel phases to detect the anomaly in real time (Feng, Zhang, Hu, & Huang, 2014). Combining SVM with ant colony achieves better performance in accuracy rate and faster running time than other traditional methods. Nonetheless, efficiency of the algorithms needs to be compared with existing techniques to show the major different between the proposed designs.

In addition, pcStream algorithm based framework has been used to evaluate three different types of attacks and to detect the data leakage, malware, and device thefts. It is a stream clustering algorithm, mainly applied for dynamically detecting and managing temporal contexts. Choosing parameter can be demanding for very extensive training time. The framework uses non-exhaustive grid search over this parameter to establish network parameter (Mirsky, Shabtai, Shapira, Elovici, & Rokach, 2017). Nevertheless, there is a need to determine the proficiency of the proposed framework to be in line with current approach of anomaly detection.

Meanwhile, a robust random cut forest technique has been used to perform dynamic data stream over anomalies detection, which treats different dimensions independently (Guha, Mishra, Roy, & Schrijvers, 2016). Its helps to preserve pairwise distance, which will be important for computation and anomaly detection. Consequently, the analytical result shows that the algorithm promises to fight against the false alarm. However, dataset size is the one of the constraints for robust random cut forest technique.

In one class support vector machine (OC-SVM) high unbalanced problems of classification is used to find the positive data for detection of DDOS attack in application

layer. OC-SVM abstract sorts' data from users' session and cluster these data to build user behavioural model (She, Wen, Lin, & Zheng, 2017). However, low acceptance level for feature selection is the limitation of OC-SVM

Another one-class support vector machine algorithm (Maglaras & Jiang, 2014) had been trained to trace the offline network and also detect anomalies in the real time. OC-SVM is an addition to the support vector algorithms to the occasion of unlabelled data, exclusively for detection of outliers. It assists the map input data into a high dimensional feature space and continual finding of maximal margin hyperplane, which utmost splits training data from the original source. Yet, the dataset used for evaluation by the authors were smaller in size when compared to the openly available larger datasets.

Furthermore, Cluster center and nearest neighbour (CANN) algorithm is a model proposed by (W.-C. Lin, Ke, & Tsai, 2015), which is capable of identifying both, similar and dissimilar classes for a given dataset. Moreover, it has increased the effectiveness and efficiency of anomaly detection. CANN comprises of the following three stages: (i) clustering technique to extract cluster center, (ii) measure and sum the distance between all the data of the given dataset and, (iii) cluster centers. Nevertheless, low acceptance level of feature demonstration with regards to a better quality in the pattern detection is a significant pitfall.

Besides, expectation maximization algorithm had been utilized to model an anomaly detection, using structural time series for the industrial Ethernet traffic system. The system decomposes the traffic into four components, based on a model, which has definite meaning for detection (Lai, Liu, Song, Wang, & Gao, 2016). This model helps to improve the performance of measuring abnormal and low false alarm rate, although, updating the parameter is a complex process.

In addition, a hybrid one-class support vector machine and deep belief nets model has

been proposed for anomaly detection with high dimensional and large-scale dataset (Erfani, Rajasegarar, Karunasekera, & Leckie, 2016). This architecture has significant detection rate and eases the computational complexity of training and testing the model. It also helps to approach the complexity and scalability problems of one SVM. Despite the ability to provide detection in high dimension, the architecture is restricted in non-convex loss function.

Online local adaptive multivariate smoothing model had been proposed to engage the high false alarm rate in network intrusion detection system (Grill, Pevný, & Rehak, 2017). In which, HTTP proxy logs dataset are collected from networks of different companies for evaluation. In this technique, true anomaly score are improved concurrently detecting the time and space. However, inefficiency in selecting more accurate parameters to set for algorithm is a challenge.

Self-organizing maps (SOM) (C. Yin, Zhang, & Kim, 2017) is one of the well-known clustering algorithms, which has been used to detect the anomalous threat in mobile devices. The process consists of three types of evaluation benchmarks, which are accuracy rate, precision rate, and recall rate. The result proves that improved SOM can produce higher accuracy rate for openly available KDD Cup99 datasets. At the same time, it consumes longer time to find the initial weight vector and the proposed setup should be evaluated with mobile devices.

Moreover, enhanced support vector (Ramamoorthi, Subbulakshmi, & Shalinie, 2011) is famous for detecting anomaly in real time applications. It provides the better classification accuracy for incoming flows, as an attack or normal flow. This model has increased the classification accuracy using weight assignment for real-time instances. Nonetheless, the complexity in selecting the parameter for this algorithm is a bottleneck.

The hidden semi-markov model (Bang, Cho, & Kang, 2017) is used to detect anomaly

in wireless sensor actuator network based on the Long-Term Evolution (LTE) signalling traffic. This model improves the detection sensitivity result, and shows more attack alarms with false positive and true negative ration. Despite that, longer time duration and unknown size for training dataset makes it complex to perform.

Lastly, Hierarchical temporal memory (Lavin & Ahmad, 2015) based model is used to test and measure open source data for anomaly detection on streaming data. In this model, Numenta anomaly benchmark broad and solid assessment tools are used for real-world anomaly detection. Regardless, the efficiency of the proposed techniques were not provided on the basis of performance and false detection.

In the above section various anomaly detection techniques from different domains had been discussed. All these techniques have their own strengths and challenges in terms of anomaly detection. Especially, clustering and support vector machine techniques provide more promising accuracy, compared to other techniques. However consuming longer time duration for training model and inefficiency in choosing the specific parameters are the challenges of the models.

2.2.1 Clustering Algorithms

This sub-section critically analyses various clustering algorithms used for real-time anomaly detection.

Unsupervised algorithm focuses on the data that does not contain any labelling information and it does not need a separate training and testing phase (Habeeb et al., 2018). Many unsupervised machine learning algorithms have been used for anomaly detection (Ahmad et al., 2017). Unquestionably, clustering helps to classify the patterns into groups and further it supports to obtain insight, classification and compressing of data (Celebi et al., 2013).Subsequently,less computational point in clustering algorithms have outperformed other unsupervised machine learning algorithms (Muller et al., 2018).

Particularly in clustering, the K-means algorithm is most widely used due to various reasons namely, simplicity, adaptability, time and storage complexity, invariant to data ordering and guaranteed coverage. Furthermore, K-means has been used to initialize other clustering algorithms, such as Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) and Spectral clustering (Celebi et al., 2013)(Chen, Song, Bai, Lin, & Chang, 2011).

The K-means clustering algorithm congregates the objects as K disjoint clusters depending on their characteristics (Mnz, Li, & Carle, 2007). The objects that possess identical characteristics are amassed in the same cluster. Münz, G., et al. (Mnz et al., 2007) proposed flow-based anomaly detection scheme based on the K-means clustering algorithm. The fundamental features of the clusters are used as patterns to facilitate operationally effective detection of anomalies based on distance in new monitoring data. Even though, it provides promising result, the system failed to address the challenges of memory consumption and execution time of the proposed scheme. Another study, has also proposed a multi-level hybrid intrusion detection model using support vector machine and modified K-means algorithms (Al-Yaseen, Othman, & Nazri, 2017). This model has remarkably enhanced the performance of classifier and reduced the training time of classifier. However, this model also faced the challenge of memory overflow. At the same time (Celebi et al., 2013) have compared and evaluated various initialization methods developed for K-means algorithm. In performance evaluation, five effective quality criteria, such as initial Sum of Squared Error (SSE), final SSE, Normalized Rand, Number of iterations and CPU time were used to measure besides memory consumption.

Additionally, clustering membership of data instance (e.g. Small subset of eigenvectors of a graph Laplacian matrix) has been determined by implementing basic clustering techniques in Spectral Clustering. This is generally employed for addressing spectral

optimization issues. In addition the m-eigenvectors have been considered as an m-step iterative bi-clustering method, wherein, all consecutive iterations seek a bi-clustering in the space orthogonal to the first (m-1) bi-clustering space.

In some cases, Spectral Clustering have better performance when compared to other traditional clustering techniques, such as K-means and hierarchical clustering (Nian, Zhang, Tayal, Coleman, & Li, 2016). Moreover, spectral clustering has been used to develop a novel spectral ranking approach for anomaly detection, which produces an anomaly ranking based on the majority class or depending on two main patterns, additionally, the positive or negative smaller class order generates ranking reference. Furthermore the proposed algorithm was evaluated for Receiver Operating Characteristic curves. Nevertheless, memory consumption and execution time were not covered in evaluation metrics.

Isolation trees constitute the Isolation Forest, where, each instance is separated from one another. The tree structures of the learned groups produce anomaly records, this approach evades the computation of costly distance or density measures (Stripling, Baesens, Chizi, & vanden Broucke, 2018)(Puggini & McLoone, 2018). They have proposed a method based on isolation forest algorithm for streaming data using sliding window. In which, for each instance of sliding windows, a score was generated to detect the concept drift. Evaluation results prove to be effective, but no comparison was carried out with existing approaches, and performance metrics were not evaluated for computational complexity (Ding & Fei, 2013). Similarly, in another study, Isolation Forest was used for dimensionality reducing pre-processing step to enhance the interpretability of isolation forest model and improve the performance on anomaly detection (Puggini & McLoone, 2018).

In the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) the number of clusters are calculated automatically and it is able to handle clusters with different density and shapes (Abdullah & Chanderan, 2017). Furthermore,

noise and outliers were identified easily. In addition, (Abdullah & Chanderan, 2017) have developed a malware behaviour detection system based on HDBSCAN that groups malware samples. This system identifies anonymous malware, and also addresses the issues of asymmetric and anonymous malware from the data, and employs single-linkage clustering techniques. Precision and recall were used for evaluation metrics. While, none of the computation performance metrics were evaluated in that method.

Table 2.3 shows different anomalies detection techniques with their domains. All these algorithms have their own pros and cons. In terms of anomaly detection, a matured state batch data processing is an added advantage; which is not in the case of real time processing. Meanwhile, in terms of the accuracy of batch processing, higher percentage in detection rate is an advantage as against the real-time processing accuracy rate. Furthermore, in batch processing anomaly detection memory consumption is not high, but real-time anomaly detection consumes higher memory. In terms of execution time, batch processing anomaly detection takes shorter algorithm execution time when compared to real-time anomaly detection. This leads to this present research to propose a new composite clustering algorithm for anomaly detection to address all the existing challenges.

2.3 Anomaly detection with big data technologies

The non-stop gathering of streaming of traffic data by the network hints the big data problems that fulfil the basic criteria of big data, which are volume, variety, and velocity (Suthaharan, 2014). On the other hand, another study has highlighted the network monitoring for security problems, associated with the 4 V's of big data, such as variety, veracity, volume, and velocity. A large amount of network data were processed efficiently in real time using big data analytics and also early detection of the various network attacks. In a traditional database, interconnections were through using data synchronization techniques which are not required in the big data analytics (Camacho, Macia-Fernandez, Diaz-Verdejo,

Table 2.3: Existing anomalies detection techniques

Domain	Techniques	References
Smartphone	pcStream algorithm	(Mirsky et al., 2017)
	Self-organizing maps	(C. Yin et al., 2017)
Network & Wireless network	Hybrid of One-class Support Vector Machines (1SVM) & Deep Belief Nets(DBN)	(Erfani et al., 2016)
	Combines a Support Vector Machine with an Ant Colony Network	(Feng et al., 2014)
	One-Class Support Vector Machine.	(Maglaras & Jiang, 2014)
	Block-based One-Class Neighbor Machine and Recursive Kernel-based online	(T. Ahmed, Oreshkin, & Coates, 2007)
	Online Local Adaptive Multivariate Smoothing	(Grill et al., 2017)
	Random cut forest	(Guha et al., 2016)
	Cluster center and nearest neighbor	(W.-C. Lin et al., 2015)
	One Class Support Vector Machine	(She et al.)
	Enhanced Support Vector Machine.	(Ramamoorthi et al., 2011)
	Expectation Maximization (EM) algorithm	(Lai et al., 2016)
	Hidden Semi-Markov Model	(Bang et al., 2017)
	Hierarchical Temporal Memory	(Lavin & Ahmad, 2015)
Earth Science	Expectation Maximization – Clustering	(Q. Liu et al., 2017)
Oil Production	Bayesian Gaussian Markov Random Fields	(Idé, Khandelwal, & Kalagnanam, 2016)

& Garcia-Teodoro, 2014). Furthermore few other studies have addressed use cases related to big data and network security, which were malware beacon activity detection in a malware infected machine attached to an outside IP address to get instructions from the command and control centre. Nonetheless, such type connections are undetected in the huge legitimate traffic, especially when the malware traffic happens over port 80, where

HTTP web traffic occurs. Designing big data analytics for this scenario will help to identify what happens to the network, such as a number of bytes transferred over a certain period. Furthermore, it helps to compare old and new patterns from log files accessed for months via graph-theoretic analysis(D. Lin, 2013) (S. Liu, 2015).

Anomalies in data collected from a sensor network can indicate the data analysts which of the sensors are broken down or detecting events are fascinating (Chandola et al., 2009). Besides, collected sensor network might contain different types of data, such as binary, discrete, continuous, audio, video, which falls under characteristics of big data. Therefore, combining big data analytics to identify the threat detection has become crucial part in current environment. Table 2.4 shows the technologies used in different anomalous detection in big data technologies.

2.4 Machine learning algorithms with big data

Machine learning is the data that powers the models whereby the models get trained well using enormous amount of data to predict accuracy. The new era of big data is catapulting machine learning to the forefront of research and various industrial applications without leaving the network environment. The production of big data has forced us to rethink not just data processing frameworks, but implementations of machine learning algorithms to perform intelligent operations.

Today, the problem of big data collections is often solved through distributed storage systems, which are designed to carefully control access and management in a fault-tolerant manner. One solution to the problem of big data objects in machine learning is through parallelization of algorithms. Data parallelism, where the data is divided into more manageable pieces, and each subset is computed simultaneously or task parallelism, in which, the algorithm is divided into steps that can be performed concurrently. Furthermore, when developing big data processing engines various number of tools can be used for

Table 2.4: Existing anomaly detection and big data Technologies

Anomalous	Hadoop	Storm	Spark	Flume	Kafka	HBase	Others	References
Host misbehaviour	✓							(Gonçalves, Bota, & Correia, 2015)
Web server	✓							(Lee & Lee, 2011)
Threat Detection		✓		✓	✓	✓		(Lobato, Lopez, & Duarte, 2016)
Network & Hardware			✓		✓			(Rettig, Khayati, Cudré-Mauroux, & Piórkowski, 2015)
Power Consumption			✓				Hive	(X. Liu & Nielsen, 2016)
Hardware & resource usage		✓	✓		✓			(Solaimani et al., 2016)
Network	✓		✓					(Rathore, Ahmad, & Paul, 2016a)
	✓		✓					(Dromard, Roudière, & Owezarski, 2015)
			✓					(Casas, D'Alconzo, Zseby, & Mellia, 2016)
	✓	✓			✓			(S. Zhao, Chandrashekar, Lee, & Medhi, 2015)
Cloud log	✓							(Cui & He, 2016)
Malware in mobile			✓				MongoDB	(McNeil, Shetty, Guntu, & Barve, 2016)

machine learning algorithms for real-time analysis. Researchers have come up with few selection criteria for machine learning into big data processing, which are scalability, speed, coverage, usability, and extensibility of the algorithms that support well with big data

processing (Landset et al., 2015). The rapidly growing network data has moves the machine learning towards the distributed and real-time processing. This scenario is challenged by the inability to appropriately detect anomaly using traditional tool for machine learning. Integrating the hybrid machine learning algorithms for the big data processing will help to increase the percentage of producing accuracy in results and also reveal the hidden knowledge of the big data, which are associations, sequences, classification, forecasting, anomalies, and clustering among the data.

This research has discussed some of the existing machine learning algorithms related to big data processing challenges. (L. Zhou, Pan, Wang, & Vasilakos, 2017) have presented a three phased framework of machine learning on big data (MLBiD), which includes pre-processing, learning, and evaluation. Furthermore, it identifies the various challenges and opportunities in this domain for upcoming years. They have also proposed a taxonomy with supervised and unsupervised learning, reinforcement, and data availability. Moreover, they have summarized several research issues, which includes new big data machine learning architecture that seamlessly support the real-time processing with massive volume of heterogeneous data.

(Fernández, Carmona, del Jesus, & Herrera, 2016) have addressed various issues related to the data distribution and parallelization of the present algorithms and with fuzzy representation. Furthermore, challenges of different big data technologies were also discussed which includes Hadoop ecosystem (HDFS, HBASE, YARN,Map Reduce programming), Spark major concept resilient distributed datasets(RDD), FlinkML, including data pre-processing, supervised learning, and recommender systems.

Besides, (Suthaharan, 2014) has focused on various problems and challenges when combining big data and machine learning for network intrusion traffic. Due to time sensitive applications and prediction in network intrusion detection, it needs extremely

capable big data technologies to tackle the recent problems. As well as some of the major problems, such as network topology, communication and security, associated with big data were addressed.

There are various factors stopping the existing machine learning algorithms to incorporate for real-time application due to most of the existing machine learning algorithms were developed for batch processing application, and not for real-time. In addition, existing architecture struggle to provide operation for real-time pipelining for incoming data and windowing of micro batches to apply for any machine learning algorithms.

2.5 State-of-the-Art machine learning algorithm with real-time big data processing technologies for anomalies detection

This sub-section critically analyses real time big data processing for anomalous detection through machine learning algorithms and their limitation.

(McNeil, Shetty, Guntu, & Barve, 2016) have analysed the available tools to detect malware in the mobile devices; however, these tools failed to integrate group user profiling, which helps to automated behaviour driven dynamic analysis on targeted malware detection. Furthermore, they have proposed scalable real-time anomalies detection and notification of targeted malware in mobile devices (SCREDENT) architecture, to classify, detect, and predict targeted malware in real-time. Even so, evaluation of that proposed architecture failed to give the promising result. In Figure 2.1 present a timeline of the historical evolution and trends for anomaly detection techniques and big data technologies. Figure 2.1 highlights various algorithms and big data technologies which are being used in this research. The timeline shows the years in which the algorithms and technologies has been developed for anomaly detection and big data framework.

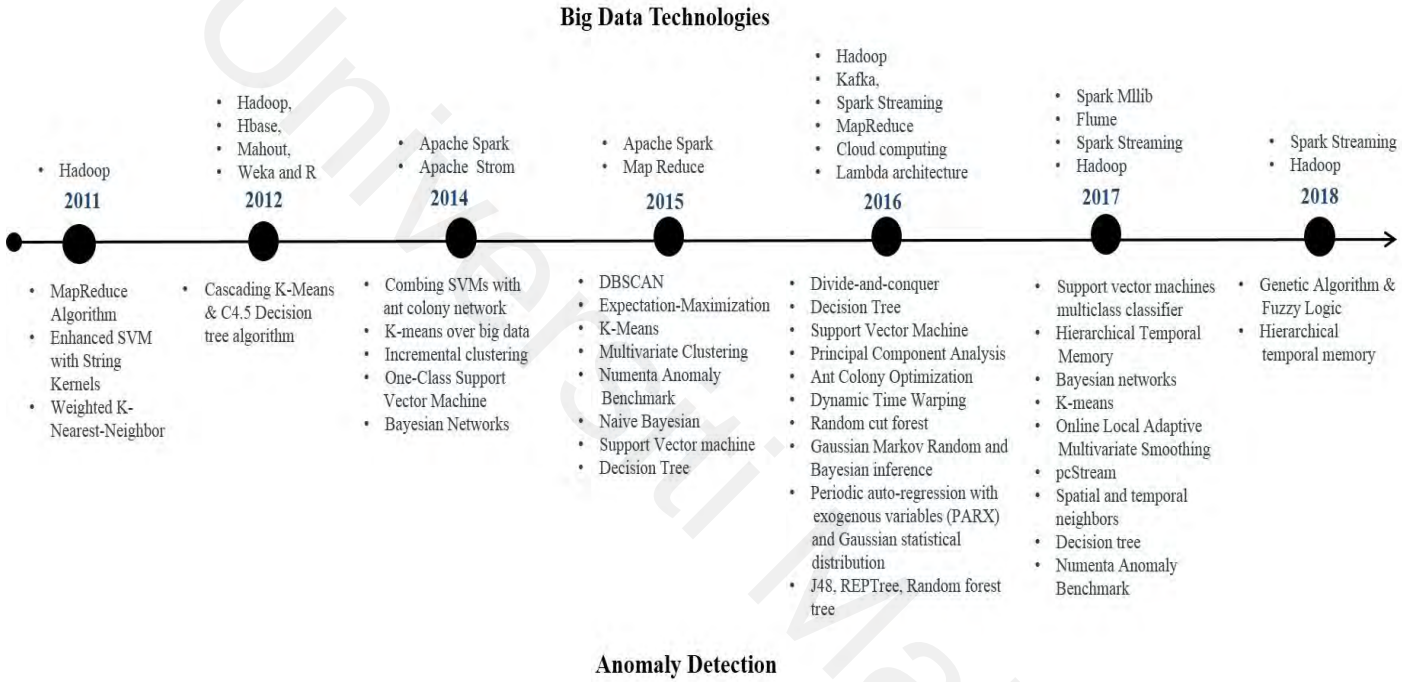

Figure 2.1: Historical evolution and trends of anomaly detection techniques and big data technologies

Moreover,(Lobato, Lopez, & Duarte, 2016) have reviewed existing security approach, such as security information and event management(SIEM) build, to handle the data gathering and process in single point. Apart from that, it produces huge amount of false alarm. In addition, they have proposed an architecture to detect threat using steam processing and machine learning in real time. This architecture combines the benefit of real-time streaming through batch processing over a past available dataset and reduction of human involvement to the system. The proposed system also helps to detect the known and zero-day attack for attack classification and anomaly. However, the proposed system is weak on the accuracy of the dataset used for the experiment, in spite of openly available dataset, such as KDD dataset.

Meanwhile,(Gonçalves, Bota, & Correia, 2015) have presented challenges in complex network infrastructure, which contains information of the number of devices stored in the vast logs file. Therefore, extracting meaningful information from that logs is demanding. The novel approach for assessing security logs of the various infrastructure devices to discover misbehaving hosts using machine learning and data mining techniques. The proposed approach has two phases. Firstly, executing a set of steps for defining and configuring the detection mechanism, and secondly executing the detection mechanism in runtime. Nevertheless, the experimental setup is based on batch processing, and efficiency of the output is not accurate enough and high human intervention is also needed to automate some of the process.

Another study reveals that out of many other anomaly detection models, the machine learning has been most widely used, whereas growing number of network traffic challenges the existing system, since it needs to perform complex calculation (Cui & He, 2016). In addition, a model was proposed to yield better performance in detecting anomaly using Hadoop, HDFS, Mapreduce, cloud and machine learning algorithms. Further, weka

interface was used in the model to evaluate accuracy and efficiency with naive bayes, decision tree and support vector machine algorithm. In fact implementation of cloud infrastructure and real time input data streaming were not addressed well.

Besides, (Rettig et al., 2015) have addressed challenges in detecting anomalies in the streaming data, mainly focusing on generality and scalability. They have proposed a new approach to evaluate online anomaly detection with two metrics, using entropy and pearson correlation. Moreover, big data streaming components, such as Kafka queues and Spark Streaming are used to assure the generality and scalability issues. Nonetheless, complex processes were limited to handle by the data and also long time duration for periodic batch processing.

Most of the current anomaly detection methods have been employed to handle the batch processing model, which needs to be setup manually and has to be trained to discover the threats. Nevertheless, these models have a low reliability and scalability, and used for non-real time detection (Wang et al., 2018). Furthermore, a study proposed hierarchical temporal memory model, which can predict the flow of data in real-time depending on the state of the previous learning. However, simulated dataset were used for evaluation as against an openly available attack dataset, and also, processing performance and efficiency has to improve for proposed model (Wang et al., 2018).

A study has proposed a novel framework for real-time network traffic anomaly detection based on machine learning algorithms to deal with large amount of real-time data in scalable manner and to be fault-tolerant (S. Zhao, Chandrashekar, Lee, & Medhi, 2015). The idea is to incorporate existing machine learning with big data processing framework in order to perform real-time processing and analyse the real-time network-flow. Using such technologies; the authors were able to get promising real-time network anomaly detection results. However, accuracy still needs to be improved, and visualization tools for

understanding lively behaviours have to be incorporated.

In recent years, user group profiling has been a major threat in malicious mobile attacks, in this context, a study has proposed a scalable solution for real time anomalies detection to predict targeted malware in real time (McNeil et al., 2016). Moreover, the proposed solution is based on notification of the target malware in mobile devices in order to minimize the number of parallel dynamic analysis by joining the behaviour-triggering probability approach and the user groups. The result shows proactive, adaptive alerts to individual users, however, it consumes a lot of computational time.

Furthermore, a real-time intrusion detection system has been also proposed based on apache Hadoop for ultra-high speed big data environment (Rathore, Ahmad, & Paul, 2016), which detects unknown network attacks using machine learning algorithm. The proposed intrusion detection system architecture has four layers, which are: (i) data gathering, (ii) pre-processing and load balancing, (iii) analysis, and (iv) decision making layer. The system presents nine parameters for classification for feature selection scheme in pre-processing stage. The results show that even using various classifiers, such as REPTree, J48, random forest tree, support vector and Naïve Bayes, only J48 and REPTree have performed better, in terms of accuracy at the cost of high memory consumption.

Likewise, (Juvonen, Sipola, & Hämäläinen, 2015) have proposed a framework for finding abnormal behaviour from http log. Additionally, it also compared random projection, principal component analysis and diffusion map algorithms for anomaly detection using high-dimensional data in real time. The memory usage and speed of all the three algorithms were tested. However, simulated logs were used instead of openly available datasets and scalability of the framework is unknown.

Similarly, a novel real-time anomaly detection framework has been proposed based on dynamic cloud resource scheduling (Solaimani, Iftekhar, Khan, Thuraisingham, &

Ingram, 2014). The idea is to monitor the virtual machine stream data performance, such as CPU load, memory usage and I/O. The study used a distributed framework like Apache Storm to deal with performance stream data and make decision without delay. The proposed framework has illustrated its effectiveness by offering a real-time complex analytic functionality over stream data. Additionally another study has evaluated various algorithms for anomalies detection in real-time based on execution time, CPU usage, and the number of anomalies. The objective is to monitor the streaming log data generated in the national educational network (Hasani, 2017). However, it did not focus on the scalability, and efficiency, moreover small sized dataset were used for analysis.

In addition, a new method for data driven Quality Management in industry processes has been proposed, which allows a multidimensional analysis of the anomalies and their real-time detection in the running system (Stojanovic, Dinic, Stojanovic, & Stojadinovic, 2016). The system proposed has been based on learning the normal behaviour of the system (based on past data) and detecting an anomalous behaviour in the real-time (by processing real-time data). The approach revolutionizes the way the quality control will be applied in complex processes with many nonlinearly correlated parameters. Nevertheless, accuracy and efficiency were not addressed in the method.

Besides the above, another study has also proposed a real-time threat detection system based on stream processing and machine learning algorithms, which helps to detect the known threats and classify them. Nevertheless, this approach has high latency with responses time, and the results were not analysed on the memory usage and duration (Lobato et al., 2016).

Lastly, (X. Liu & Nielsen, 2016) have revealed that the existing anomaly detection models for smart grid are mostly based on offline mode, and they also consume huge amount of energy. In addition, they have proposed a method to detect anomaly, using

in-memory distributed framework. The framework contains Spark Streaming and lambda system. Its major advantage is to support scalable live streaming for real-time detection. However, the framework took longer time duration to train the model. Consequently scheduling of real-time task was unknown.

All the above discussed approaches and their limitations demand reassessing the framework design to support the anomaly detection. Especially, an advance real time big data analytics for anomaly detection using machine learning will bring promising and improved performance and accuracy for anomaly detection.

Industries might face more challenges with emerging new set of technologies, such as cloud to the edge, data from IoT, smart devices, intelligent things, block chains, connected home, virtual reality, 5G, quantum computing, serverless and PaaS. However, sophisticated advance anomalous detection techniques using machine learning and big data should be adequate to handle those challenges. Table 2.5 reveals the big data processing technologies for anomaly detection using machine learning. Table 2.6 describes the number of commercial platforms, which have integrated machine learning, big data technologies into their anomalous detection.

Table 2.5: Overview of big data processing technologies for anomaly detection using machine learning

Domain	Finding	ML Techniques	Reference
Telecommunication and Mobile network	Extracting information from the security logs are trivial	Unsupervised – Clustering (Expectation- Maximization algorithm) and Supervised linear classification WEKA software used	(Gonçalves et al., 2015)
	Detecting anomalies in the streaming data	Pearson correlation pipeline	(Rettig et al., 2015)
	Existing framework do not integrate group user profiling which helps to perform targeted malware detection.	K-means clustering, Markov models	(McNeil et al., 2016)
Network traffic	Huge amount of traffic data delay the response time to detect the threats – Scalability and Accuracy - Memory consumption and searching complexity – boosting accuracy	Principal Component Analysis	(Lee & Lee, 2011)
	Current security approaches are single point, and generate huge amount of false alarm.	Decision trees algorithms, Artificial Neural Network, Support Vector Machine	(Lobato et al., 2016)
Smart grid	Detection models are based on offline. Huge amount of training data are required.	Periodic autoregression with an exogenous variable.	(X. Liu & Nielsen, 2016)
Cloud environment	Increasing network traffic data is a bottleneck for an existing system, which has to perform the complex calculation.	Naïve Bayes, Decision Tree and Support Vector Machine (Weka Interface)	(Cui & He, 2016; Shirdastian, Laroche, & Richard, 2017)

Table 2.6: Summary of commercial platform and solution for big data streaming analytics

Product name	Description	Architecture Components	Machine Learning	Source
Anodot	Anodot platform automatically selects the appropriate algorithm to exhibit the data pattern from available options and adjust it over time, based on real-time, mainly for anomaly detection	Hadoop, Spark, Hive, anomaly detection engine.	Yes	(Dror, 2017)
Numenta	Numenta platform can work with both, predictable and highly unpredictable platform. The algorithm works with continuously learning algorithm, so that data are automatically handled without human intervention.	No access to architecture component of the system.	Yes	(Lavin & Ahmad, 2015)
Microsoft azure stream analytics	Azure stream analytics is real-time analytic computations on streaming data to provide multiple solutions. It combines azure streaming analytics and apache storm on azure HDinsight, using PaaS solution.	Kafka, RabbitMQ, ActiveMQ Apache storm, azure stream analytics Hbase, HDFS.	Yes	(Branscombe, 2015)
WSO2 analytics	WSo2 analytics platform an one stop centre, which is capable to collect and analyse various IoT sensor data, which do real-time and batch threat analysis using a machine learning algorithm	Event receivers, Siddhi event, Apache spark, Caasandra, Hbase	No	("Introducing WSO2 Data Analytics Server," 2015)
Striim	Striim is end-to-end in-memory streaming platform used for infrastructure critical application.	Data lake, Kafka, NoSQL, Hadoop, Hbase.	Yes	(Wilkes, 2016)
Tibco streamBase	TIBCO streamBase is the event processing platform, which develop, host, execute, and integrate the predictive analytics in big time real time.	Hadoop, spark. Kafka, Flume, casssandra,Hbase.	No	("TIBCO StreamBase and the TIBCO Accelerator for Apache Spark," 2017)

2.6 Taxonomy of Real-time big data processing technologies for anomaly detection.

This section highlights and proposes a taxonomy for the anomaly detection, big data, and machine learning. A taxonomy of real-time big data processing for anomaly detection is classified into different categories, which are, techniques, application, anomalies, modes, data, big data processing, and the record categories. Figure 2.2 shows the classified taxonomy of real-time big data processing for anomaly detection, based on the set of parameters found in majority of the literature review.

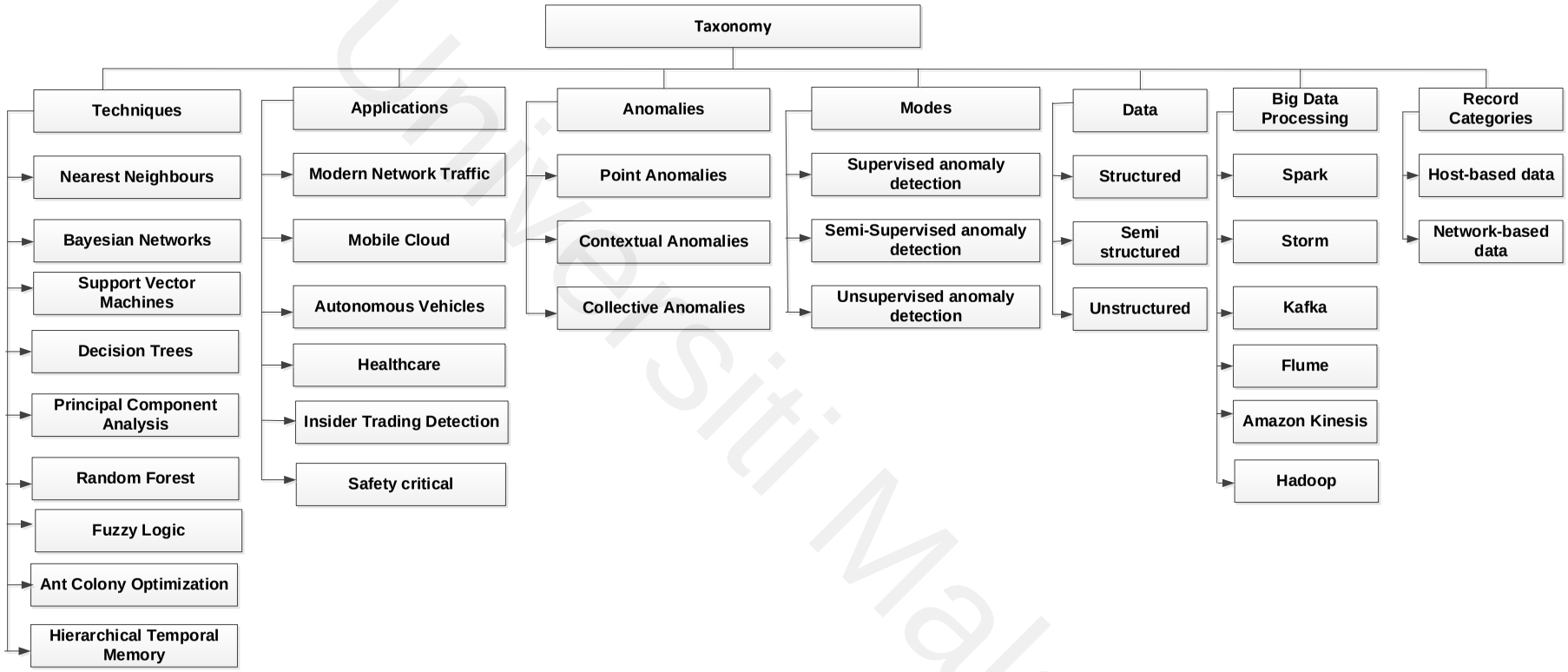

Figure 2.2: The process of real time big data processing technologies for anomaly detection

2.6.1 Techniques

Among a vast group of anomaly detection techniques accessible, this research has examined the following seven major techniques: Nearest Neighbours (Wauters & Vanhoucke, 2017), Bayesian network (Mascaro, Nicholso, & Korb, 2014), Support vector machine (Aburomman & Reaz, 2017), Decision trees (Muniyandi, Rajeswari, & Rajaram, 2012), Random forest (Farnaaz & Jabbar, 2016), Fuzzy logic (Hamamoto, Carvalho, Sampaio, Abrão, & Proença, 2017), Principal component analysis (Hamamoto et al., 2017).

2.6.1.1 Nearest Neighbours (NN)

Nearest Neighbours depends on the utilization of distance measures. NN method embraces the entire sampling set and incorporates the information in the set, and as well as the coveted grouping for respective item. The distance between each item in the sampling set must be processed for the purpose of classifying the points, wherein, the k closet passages in the sampling set is considered as the point in far distance. However, the shortcoming of the NN is the similarity measure, which leads to misclassification of points because of its inefficiency in accurately calculating distance between them, particularly while classifying small subset of the features (Su, 2011a). Historical data is employed to identify nearest neighbours at a given data point. The applications of the Nearest Neighbour method usually revolves around the similar domain of artificial intelligence methods, which are classification and prediction. The initial k-Nearest Neighbour has been deployed as a predictor and allowed to benchmark its accuracy and scalability. Nonetheless, the algorithm have outflows in concerning memory requirements and computation complexity in terms of anomaly detection (Wauters & Vanhoucke, 2017).

2.6.1.2 Bayesian Networks (BN)

Bayesian Networks have been broadly utilized for grouping issues. BN comprises of qualitative and quantities parts. The qualitative part of the system is spoken to by a coordinated non-cyclic graph, whose nodes denote the random factors in the problem domain and whose edges systematize significance relations between the factors they interface. The quantitative part of the model comprised of an arrangement of probability distributions on each node (Cozar, Puerta, & Gamez, 2017). Bayesian Networks can be used for detecting anomalies within vessel tracks. Dynamic and static network are produced by using Bayesian Networks learner on an Automated Identification System data, which has been supplemented with real-world data, which proved distinct and complementary strengths in identifying anomalies (Mascaro et al., 2014).

2.6.1.3 Support Vector Machine (SVM)

Support Vector Machine has been used to construct a decision boundary, which has the most extreme edge between the typical data set and the source (S. Yin, Zhu, & Jing, 2014). SVM is a classifier in light of finding a separating hyperplane in the feature space between two classes is such a path, to the point that, the distance between the hyperplane and the open data points of each class is increased. The approach depends on minimized classification threat as opposed to on optimal classification. Especially helpful when the number of features, m, is high and the number of data points, n, is low (m»n) (Buczak & Guven, 2016). Moreover, SVM is a robust classifier, used for many classification tasks such as, network intrusion detection. The main focus is extending two-class SVM classifiers to multi-class classifiers. This method is the most effective for classifying the samples from the NSL-KDD dataset, such a classifier is crucial in intrusion detection system (Aburomman & Reaz, 2017).

2.6.1.4 Decision tree

Decision tree is a tree-like structure that has leaves, which denote to groupings and divisions, which thusly state to the conjunctions of highlights that prompt those classification. The outstanding known techniques for naturally fabricating decision tress are the ID3 and C4.5 algorithms. The two algorithms formulate decision tress from an arrangement of training data utilizing the idea of data entropy (Buczak & Guven, 2016). In addition, a decision tree refines its decision boundaries on each cluster by acquiring knowledge from the subgroups within the cluster. The results from the decision trees in each cluster are exploited and a final conclusion is made decision trees (Muniyandi et al., 2012). Clustering helps to group the normal and abnormal data points in anomaly detection.

2.6.1.5 Random Forest (RF)

Random Forest is a collaborative classifier that is used to improve the accuracy. It consists of two stages such as, feature selection, and classification(Farnaaz & Jabbar, 2016). Random forest generates multiple decision trees from random subsets of data. Each of them capturing different regularities, since random subset of the instances are in the interest (Promod & Jacob, 2016). One of the major advantages of the Random forest is that it yields low classification errors when compared with other traditional classification algorithms. The Random forest can be used to detect four types of attacks like DOS, probe, U2R and R2L (Farnaaz & Jabbar, 2016). Likewise, RF has been used in botnet detection due to its high accuracy in prediction and ability to handle diverse bots, based on data characterized by a very large number and diverse types of descriptors (Singh et al., 2014) Yet, when working with large dataset and complex estimation procedures, the RF consumes lot of computational time (Genuer, Poggi, Tuleau-Malot, & Villa-Vialaneix, 2017).

2.6.1.6 Fuzzy Logic algorithm

Fuzzy Logic algorithm is capable of making rational decisions in an environment that is imprecise, uncertain, and incomplete. It uses time interval to detect an anomaly in the network. Moreover, exponentially weighted moving average techniques are applied to calculate threshold values, which represent more recent higher weight (Hamamoto, Carvalho, Sampaio, Abrão, & Proença Jr, 2018).

2.6.1.7 Principal Component Analysis

Principal Component Analysis creates a digital signature for traffic characterization of a network segment. It is a statistical procedure used to decrease those dimensional multivariate issues by examining those difference for every variable among all response measurement. Furthermore, the response information can be symbolised by a reduced fixed set of dimensions without much loss of information (Hamamoto et al., 2017). Besides, it analyses noteworthy information data from logs to find the important activity time intervals among the data set, and afterward diminish them, so this new set can professionally characterize the consistent conduct of a network segment (Fernandes, Rodrigues, & Proença, 2015).

2.6.1.8 Ant Colony Optimization

Ant Colony Optimization is inspired by the ability of ants to find the shortest path between their colonies to the food sources. Ant Colony Optimization is where a population of agents competing and globally asynchronous, cooperate with one another to find an optimal solution. Ant Colonization Optimization and Dynamic Time Wrapping methods have been used in the environment of pattern recognition and anomaly detection (Fernandes, Carvalho, Rodrigues, & Proença, 2016).

2.6.1.9 Hierarchical Temporal Memory (HTM)

Hierarchical Temporal Memory is capable of predicting the flow of data in real-time, based on the condition of the precedence in learning. Moreover, HTM has been widely employed for detecting anomalous in distributed real-time systems, mainly due its capability in yielding highly accurate detection. This techniques comprise of a solid framework which helps in better estimation, categorisation, and generating continuous time based data sequence, online. The HTM is also used to detect anomalous sequences on the vehicular controller area network bus, which sends alarm signal during abnormal conditions (Wang et al., 2018). Furthermore, HTM is employed to robustly detect anomalies on a variety of data streams, which mainly deals with noisy data and to minimize false positives (Ahmad et al., 2017).

2.6.2 Applications

In this sub-section major application scenarios in the area of real-time big data processing for anomalous detection have been discussed. The growth of IOT contributes for different number of applications with sensors that produce important data that changes over time (E. Ahmed et al., 2017). In fact, detecting anomaly in that data can help to overcome many challenges for organizations. Interestingly, challenges and limitations associated with some of the applications like network intrusion, healthcare, image processing, fraud detection, safety critical applications, and insider trading detection application have been discussed.

2.6.2.1 Modern network traffic scenario:

Present IoT infrastructures use various connected and mobile devices, and the machine-to-machine communication generates large scale of sensor data every second, and the generated data are stored in the cloud. These data are heterogeneous in nature with a variety of parameters such as, IP address, data transfer speed, volume, etc. (Xie & Chen,

2017). The heterogeneous data generated from these devices should be monitored and collected in real-time to be patterned for detecting abnormal behaviour.

2.6.2.2 Mobile Cloud:

Smartphones provide a wide variety of sensors integrated in one device, allowing monitoring, processing, measuring, locating the device anytime. Multimedia sensors like the microphone, dual-camera, and finger print sensors integrated in smartphones allow to employ them in a wide variety of applications (García, Tomás, Parra, & Lloret, 2018). All these data generated from sensors needs to be stored and processed in cloud due to resource shortage in the mobile device (Karim et al., 2017). Cloud computing technologies for mobile devices offer an innovative method of delivering IT services efficiently, like IT services that are available at all times and omnipresent (Ali, Shrestha, Soar, & Wamba, 2018).

Particularly, cloud computing helps mobile health services to provide access for electronic healthcare system. For example, remote data processing and monitoring, remote consultation, and digital multimedia data (Y. Zhao, Ni, & Zhou, 2017). Moreover, all these services contribute for the growth of big data in their services. In recent days, many of the cloud service providers like Amazon, Google cloud, Microsoft, IBM, Oracle and others major organizations are adopting big data technologies into their cloud platform for superior process, storage and analysis. All these data in the cloud needs to be monitored in real-time to detect any abnormal behaviour such as data theft, patient illness, origin or separating of new diseases from different demographic, cloud infrastructure such as memory usage, power consumption, cooling system.

2.6.2.3 Autonomous vehicles scenario:

These driverless vehicles have redefined the ecosystem of automotive industries, accompanied by enormous big data generated by connected devices in vehicles(Ger, 2017). Attackers have begun to target these connected vehicles data to hack the vehicles, which can help them to control the entire system, and disable the vehicle anytime. Now, most of the modern vehicles are equipped with a number of modules, such as in-vehicle networks (IVN) including, engine controls units (ECU), body control modules (BCM), smartphone integration module, which provide critical functionalities for control and safety of the vehicles (Symantec, 2016). These modules need to be analysed in real time to detect the anomalies in the vehicles, which includes sudden increase of speed, radar sensors detection, camera sensing, abnormal petrol consumption, sudden engine failure, inappropriateness in changing lanes, and inaccurate object detection.

2.6.2.4 Healthcare scenario:

Real-time anomaly detection helps in monitoring services of the patients to detect the anomalies in real-time and timely manner. This helps the hospital and caretakers to make a wise decision, especially as elderly people living alone are becoming a social problem for community and governments (Yasumoto, Yamaguchi, & Shigeno, 2016). The data can have several abnormal patient conditions or instrument errors, human errors, or focus on detecting disease outbreaks in a specific area. These records consist of various types of features such as, patient age, gender, height, sugar level, blood details, which need to be analysed with high accuracy (Chandola et al., 2009). In addition, growing numbers of compatible IoT devices help healthcare industry to collect and analyse massive data.

Furthermore, current medical technologies produce various types of multimedia data such as, high quality videos, image, graphics, and sound files. These multimedia data contain rich and complex information, which help in diagnosing and monitoring diseases.

Moreover, computing requirements of multimedia solutions for healthcare have led to the employment of cloud services for e-healthcare system (García et al., 2018). On the other hand, incorporating multimedia data into electronic health records face big data challenges and these data needs to be monitored in real-time to detect any abnormal behaviour in the system.

2.6.2.5 Insider Trading Detection:

In stock markets, data changes in milliseconds and anomaly detection techniques have been used to detect the insider trading early. People make illegal profits by leaking the inside information before the actual information is made available to the public. The information could be of pending merger, acquisition, a terrorist attack, judgment on the particular industry or any other relevant information that affect the stock prices of any specific industry. Insider trading can be detected by identifying anomalous trading activities in the market. It has to be detected in real time manner to prevent people from making illegal profits (Chandola et al., 2009).

2.6.2.6 Safety Critical Detection:

For safety-critical system, attackers begin targeting mobile connected vehicles, and vehicle-to-vehicle communications networks to enter into the controls units and body control modules, which provide critical data about the vehicles. Besides, these mobile devices store, process and access critical data from the cloud infrastructure. Here anomalies detecting techniques can help to monitor or notify the vehicle at what time it has been attacked, or just malfunctioning.

Conversely, all the above use case scenarios reveal that there are still challenges and difficulties in using existing anomaly detection techniques. Furthermore, every day increasing numbers of new types of threat are found in the connected devices. Current

monitoring technologies are challenged to detect the anomalies because of the growing volume, variety, and velocity of data received for the analysis

2.6.3 Anomalies

This sub-section briefly overviews three different categories of anomaly such as point, contextual, and collective anomalies in real time anomalous detection.

2.6.3.1 Point anomalies

Point anomalies happen when data points are observed upside or downside of normal points in the available dataset (Hayes & Capretz, 2015). For example, User-to-root (U2R) and remote-to-local (R2L) attacks are best known for point anomaly. In fact data point helps to point the unauthorized access from a remote and local access privileges (M. Ahmed, Mahmood, & Hu, 2016).

2.6.3.2 Contextual anomalies

Contextual anomalies discover association in datasets and detect differences in their external behaviour characteristics, and will label anomalous effects in the data. For example power consumption of an office building was found to be far greater during midday, and during a work day, as against night, and during weekends (Hayes & Capretz, 2015).

2.6.3.3 Collective anomalies

Collective anomalies is a group of related data instances, which act differently corresponding to the whole dataset, In fact, these set of data instances is termed a collected anomaly. For instance, DoS attack best suits the collective anomaly, these attack produces various connection request to web server, out of that, only a single request is reliable (M. Ahmed et al., 2016).

2.6.4 Anomaly Detection Modes

This section discusses different types of modes that are commonly used in anomaly detection.

2.6.4.1 Supervised anomaly detection

Supervised anomaly detection techniques detect anomaly based on generating a set of grouping rules that aid in predicting future data. An example of supervised anomaly detection is classification-based anomaly detection (Kakavand, Mustapha, Mustapha, Abdullah, & Riahi, 2015).

2.6.4.2 Semi-supervised anomaly detection

Semi-supervised anomaly detection is an approach that models only the normal records. The other records are labelled as outliers in the testing phase (Kakavand et al., 2015).

2.6.4.3 Unsupervised anomaly detection

Unsupervised anomaly detection focuses on the data that does not contain any labelling information and it does not need a separate training and testing phase. A general example is clustering based anomaly detection (Kakavand et al., 2015).

2.6.5 Data

This section details the different types of data used in anomaly detection.

2.6.5.1 Structured data

Structured data is tabular data found in spreadsheets or relational databases (Gandomi & Haider, 2015). Structured data can be processed more efficiently for anomaly detection compared to semi structured or unstructured data.

2.6.5.2 Semi structured data

Semi structured data fall between the structured and unstructured data groups. These data do not conform to strict standards. An example of semi structured data is XML (Gandomi & Haider, 2015). Compared to structured data, the semi structured data lacks proper formatting and is time consuming for anomaly detection.

2.6.5.3 Unstructured data

Unstructured data does not follow any strict format or sequences. Examples of unstructured data are social media contents, images, audio, IoT sensor data, and video (Gandomi & Haider, 2015). The processing of unstructured data for anomaly detection is time and memory consuming, and require lot of resources.

2.6.6 Big Data processing

The state-of-the-art technologies utilize different types of big data tools in a variety of domains, this section discuss those big data tools used for anomaly detection.

2.6.6.1 Spark

Spark is an open source big data tool used for data processing (Solaimani, Iftekhar, et al., 2014). Spark receives data from Kafka and processes it in real-time, using machine learning algorithms for anomaly detection.

2.6.6.2 Storm

Storm is a data application programming framework that is used to write applications for rapidly processing large amounts of data (Ranjan, 2014). Storm is a tool similar to Spark that processes data in real time, they both have their own pros and cons.

2.6.6.3 Kafka

Kafka is mainly utilized for constructing data pipelines in real-time. It is also useful for developing online streaming applications. This technology can be horizontally gauged, fool proof, and rapidly swift, hence it is mostly popularly used in number of sectors (Assunção, Calheiros, Bianchi, Netto, & Buyya, 2015).

2.6.6.4 Flume

Flume is a distributed service used for real-time data collection, temporary storage, and delivery of data to a target (Birjali, Beni-Hssane, & Erritali, 2017).

2.6.6.5 Amazon Kinesis

Amazon Kinesis is a distributed message queuing framework (Ranjan, 2014). It is capable of handling large data size and large pipelines, furthermore, the output generated by Kinesis can be applied to machine learning algorithms.

2.6.6.6 Hadoop

Hadoop is one of the most popular big data technology frameworks, which helps to solve this scalability problem by the Hadoop distributed file system (HDFS). It has been used to store large amounts of data across multiple nodes of commercial hardware (Landset et al., 2015).

2.6.7 Record categories

In this section different types of record categories that have been used in anomaly detection have been elaborated.

2.6.7.1 Host-based

Host-based data contains incoming and outgoing network traffic of an individual device on a network (Sahasrabuddhe, Naikade, Ramaswamy, Sadliwala, & Futane, 2017). The

host based data can be processed for anomaly detection using different machine learning algorithms in a single host.

2.6.7.2 Network-based

Network-based data contains network traffic to and from all devices in a network (Sahasrabuddhe et al., 2017). It can be processed for anomaly detection using different machine learning algorithms all the devices connected network. The network based data facilities detection of anomalies based on the pattern generated from the sensor data (L. Wang, 2017).

The above sections had thoroughly discussed, the various existing anomaly detection techniques, applications, categories and modes of anomaly detection, types of data, big data processing technologies, and record categories.

2.7 Evaluation metrics for clustering algorithm and system performance

In this section various evaluation metrics for determining cluster accuracy and system performance of the real-time big data processing and anomaly detection are discussed. In the case of cluster quality and accuracy, the evaluation metrics have been categorized into three main classes: internal and external measures, which are based on the internal or external information used for the cluster evaluation. The most popular internal and external techniques such as Silhouette coefficient, Sum of Squared Errors, Calinski and Harabaz for internal and normalized mutual info, adjusted rand score and F-measure for external as have been employed (Xu & Tian, 2015)(Kremer et al., 2011)(Amini, Wah, & Saboohi, 2014). Likewise, for the measurement of the clusters accuracy, statistical techniques such as confusion matrix, Precision, Recall, F1 score and Kappa, Matthew's coefficient have been utilized (Aghabozorgi, Shirkhorshidi, & Wah, 2015)(Kokate, Deshpande, Mahalle, & Patil, 2018). Selected evaluation techniques have been listed in the Table 2.7 and description of

Table 2.7: Evaluation techniques

Internal techniques	External techniques	Statistical techniques	Performance measurement
Silhouette coefficient	adjusted rand score	Precision, Recall	Memory consumption
Calinski and harabaz	normalized mutual info	F1 Score, Kappa	Execution time
		Matthew's coefficient	

the evaluation techniques are discussed in chapter 5. Table 2.8 summarize the finding on leading evaluation techniuqes mostly used by researchers for anomaly detection.

Similarly, based on this literature review, this research has discovered many Spark Streaming performance evaluation parameters, such as processing time, execution time, algorithm run time, job completion time and usage of memory (Bharill, Tiwari, & Malviya, 2016)(Tang & Fong, 2018).

Table 2.8: Leading evaluation techniques used for anomaly detection

References /Evaluation Techniques	Silhouette	Calinski and harabaz	Normalized Mutual Info	Adjusted Rand Score	Precision	Recall	F1-Score	Kappa	Matthew's coefficient
(Saitta, Raphael, & Smith, 2008)	✓	✓							
(Y. Liu, Li, Xiong, Gao, & Wu, 2010)	✓	✓							
(Goldstein & Uchida, 2016)					✓	✓			✓
(Xu & Tian, 2015)	✓		✓	✓			✓		
(Dahiya & Srivastava, 2018)					✓	✓	✓	✓	
(Kremer et al., 2011)	✓			✓			✓		
(Goix, 2016)					✓	✓			✓
(Tomašev & Radovanovic, 2016)	✓	✓		✓					
(J. Zhou et al., 2010)	✓	✓		✓	✓	✓			
(Aghabozorgi et al., 2015)	✓	✓	✓	✓			✓		
(Marir, Wang, Feng, Li, & Jia, 2018)			✓		✓	✓	✓		
(Hafsa & Jemili, 2019)					✓	✓	✓		
(Bharill et al., 2016)			✓	✓	✓	✓	✓		
(Othman, Ba-Alwi, Alsohybe, & Al-Hashida, 2018)					✓	✓			
(Amini, Wah, & Saboohi, 2014)	✓		✓	✓	✓	✓	✓		
(Kabore, Kermarrec, & Lenca, 2018)					✓	✓	✓		
(Juvonen & Sipola, 2014)	✓				✓				✓
(Celebi, Aslandogan, & Bergstresser, 2005)								✓	

2.8 Research challenges

This section emphasizes the most important research challenges in real-time big data processing technologies for anomaly detection. State-of-the-art techniques in anomalous detection, real-time big data processing, and machine learning have been surveyed to identify the research challenges, recommendation and future research directions.

2.8.1 Redundancy

Dealing with large amount of data generated from various network sensors in real-time is a critical factor in big data management, particularly due to the constant repetition of previously generated data. Even though, the existing big data processing technologies, such as Hadoop and Spark frameworks have been developed for handling data replication across multiple clusters, still these technologies are inadequate in addressing the challenges related to data redundancy, data quality, inconsistency and cost of maintaining storage (Bhadani & Jothimani, 2016). Moreover, these technologies lack schema to minimize redundancy and are not capable enough to store huge amount of data (Hashem et al., 2015). Hence, it becomes essential to design a framework that is capable of addressing and minimising redundancy issues aimed at catering present and future needs.

2.8.2 Computational cost

A number of studies have focused on merging or incorporating several techniques to increase the performance of anomaly detection, which leads to increase in computation cost (W.-C. Lin et al., 2015). Moreover, high dimensionality combined with large sample size creates issues, such as heavy computational cost and algorithmic instability (Fan, Han, & Liu, 2014). Therefore, using big data technologies along with cloud will address the computational cost issue, by incorporating parallel and distributed processing, which helps to build multiple clusters leading to minimization of the computation cost. The mass

production of high chips and processors has reducing their costs, hence utilization of these hardware will increase the power of systems that helps to process huge volume of data in real-time, resulting in reducing in computational cost.

2.8.3 Nature of Input data

In an aspect of any model built, the first thing to look into is the nature of input data. Input data is the collection of data instances, such as object, record, point, vector, pattern, event, case, sample, observation, entity. They are various set of attributes for each data instances, such as variable, characteristics, features, field, and dimension. It has two different types of attributes such as binary, categorical or continuous. Each data instance mostly falls under the category of either, univariate or multivariate. The diverse nature of input data makes the anomaly detection techniques struggle in selecting appropriate algorithm to handle that specific data. Basically, anomaly detection techniques will vary based on nature of attributes in that application (Chandola et al., 2009). This issue will be addressed by developing hybrid unsupervised machine learning algorithm in our proposed framework.

2.8.4 Noise and missing value

The streaming data in network sensor consist of different types of data, such as binary, discrete, continuous, audio, video, and image. These data collected from various deployed sensors via a communication channel includes noise and missing values, due to the incoming speed of data (Chandola et al., 2009). Noise and missing values can produce high probabilities for rising the false positive alarm in anomalous detection. Huge quantity of unrelated features produce noise in the input data, which bypass the true anomalies (Erfani et al., 2016). These issue will be addressed by incorporating auto noise cleansing module in the detection framework. The auto cleansing module will also address the

missing value issue by adding NA symbol to datasets.

2.8.5 Parameters Selection

Finding the appreciate parameters for any machine learning algorithms can be challenging (Mirsky et al., 2017). Especially when dealing with real-time anomaly detection, it is essential to consider single, multiple and hyper parameters before choosing them. In addition, a set of parameters that works well at the early stages of the evolution process may not perform well at the later stages and vice versa (Sarker, Elsayed, & Ray, 2014). Parameters were one of the major contributors that decide the performance of the algorithms. Further, it can give huge impact or delay to training the model. Alternatively, we can work on the parameter-free algorithm in identifying the node partitions in streaming, directed, bipartite graphs, and monitor their evolution over time to detect events (Akoglu, Tong, & Koutra, 2015). Employing the likes of eccentricity techniques will address this challenge because it will minimise parameter selection.

2.8.6 Inadequate Architecture

The existing architecture are capable of handling anomaly detection in batch processing and less volume of data, however, they are incapable of handling big data in real-time. Organizations are working to produce the big data architecture to perform better, but when it comes to real-time data it is fundamentally a different architecture than big data. Components of the real-time architecture have to merge application and analytics to propose the new way of working environment that achieves the needs of both, data in motion (fast) and data at rest (big). Big data architecture is inefficient when it is not being integrated with existing enterprise data; the same way an analysis cannot be completed until big data correlates it (Katal, Wazid, & Goudar, 2013). Incorporating various big data technologies with hybrid machine learning algorithms will address the architectural issues.

2.8.7 Data visualizations

Processed and analysed data or report needs to be visualized by the user as well as must provide insight from the report. Nevertheless, challenge lies in selecting appropriate visualization techniques, for the anomalies detection from the various connected devices. Multiple visualization techniques are used in the design of anomalous detection visualization from simple graph to 2D, and 3D views. Heat maps, scatter plots, parallel coordinates, and node-link graphs are easy to showcase the output when it comes to 2D and 3D. 6. The 3D interaction enables users to understand the data for decision making and visualization (Shiravi, Shiravi, & Ghorbani, 2012), hence, better detection of anomalies. Embedding the available open source visualization techniques in framework can address this problem, furthermore, the framework enable the system to automatically select the appropriate visualization technique.

2.8.8 Heterogeneity of data

Unstructured data represents almost every kind of data being produced, like social media interactions, to recorded meetings, to the handling of PDF documents, fax transfers, to emails, and more (L. Zhou et al., 2017). Structured data is always organized into highly mechanized and manageable way. It shows good integration with the database, but unstructured data is completely raw and unorganized. Working with unstructured data is cumbersome and of course costly too. Converting all this unstructured data into structured one is also not feasible. The employment of unsupervised hybrid machine learning algorithms will address the heterogeneous data issue. The incorporation of hybrid machine learning algorithms and real-time big data technologies will help to cluster the incoming data into different categories, which eventually will helps easily identify data types, whereby addressing heterogeneity issue.

2.8.9 Accuracy

Even though the existing technologies are capable of detecting anomalies, still the dependency of the outcome is unreliable due to the accuracy issues. In some cases, better accuracy is produced at the cost of high computational processing time (Ahmad et al., 2017) (Su, 2011b). This issue will be addressed by incorporating real time big data technologies with hybrid machine learning algorithms, which emerge as an alternative powerful meta-learning tool to accurately analyse the massive volume of data generated by modern applications, with less memory and power consumptions.

2.8.10 Scalability

Likewise, other important challenges to focus is scalability. Growing numbers of data which need to be processed in the real-time face major issues with regards to the scalability of the application. However, to address this issue involve adopting the distributed environment incorporating platforms such as Apache Storm, Spark, and Flink (Solaimani, Khan, & Thuraisingham, 2014). Further, growing amount of service loads will be handled by cloud platform which add the resource to the architecture as demand. Particularly, integrating Spark MLlib library into the framework provide better scalability for much higher problems (Meng et al., 2016).

In addition to the above summarized future research directions on research challenges and recommendations in table 2.9, we have identified forthcoming research directions for research communities to develop an adoptable and responsive model for real-time big data processing, which can help to collect data with labelling and converting unstructured data into semi-structure data which will be easier to label in run-time for processing. Likewise, model should support for flexible select specific feature and extract parameters for analysis. These selected parameters can be used for benchmarking various types of threat and real-time processing. Similarly, the proposed model should be more competent and timely

Table 2.9: Summary of research challenges and recommendation for future research directions

Challenges	Recommendations	Future Research Directions	Reference
Redundancy	Employing filtering module into framework will help to reduce the redundancy data.	Data Deduplication. Dimension Reduction. Network Theory.	(ur Rehman et al., 2016)
Computational cost	Including modern technologies like virtualization, cloud, edge and fog computing might help to reduce computation cost. Moreover, big data processing can use multiple clusters assisted by parallel and distributed cloud architecture.	Resources on-demand with costs proportional to the actual usage. Cloud-supported analytics.	(Assunção et al., 2015)
Nature of Input data	Developing hybrid unsupervised machine learning algorithms	Hybrid unsupervised machine learning. Deep learning.	(Weston, 2015)
Noise and missing value	Incorporating auto noise cleaning module framework will address noise and missing value problem.	Eefficient, robust, scalable, and optimized pre-processing techniques for both, historical and streaming big data.	(Assunção et al., 2015)
Parameters Selection	Eccentricity techniques to minimise parameter selection.	Developing dynamic learning algorithm provides better efficiency in parameters selection.	(Mukherjee, Shu, & Wang, 2018)
Inadequate Architecture	Integrating various modern technologies like big data, cloud, fog and edge computing technologies with hybrid or enhanced machine learning algorithms will address these issues.	AWS, open source cloud, Microsoft, IBM. In-memory architecture will be more capable for real-time analytics. Spark, H2O.	(Qiu et al., 2016) (Landset et al., 2015a)
Data visualizations	Implanting the available open source visualization techniques in framework and also spontaneous selection for appropriate visualization technique.	GraphX, HadoopR, Python, Lightning data visualization server.	(Qiu et al., 2016) (Landset et al., 2015a)
Heterogeneity of data	Combining hybrid or enhanced machine learning algorithms and real-time big data technologies will help to cluster the incoming data into different categories such as data types, size and others.	Data cleansing and data curation. The incorporation of hybrid machine learning algorithms and real-time big data technologies.	(Anagnostopoulos, Zeadally, & Exposito, 2016)
Accuracy	Embed the evaluation techniques and meta-learning tool to hybrid machine learning algorithms for accurate analyse.	The reduction of memory and power consumptions when processing large amount of data.	(Qiu et al., 2016) (Landset et al., 2015a)

to train as well as retrain the model more efficiently. Many of the existing works lack in retraining the model for processing, which will be remarkably beneficial in real-time. Furthermore, retraining the model should contain modules for offline and online analysis.

Similarly, the future model should comprise fast, hybrid and incremental learning algorithms for modern incoming real-time data, which can facilitate selection of right time windowing for online analysis. Besides, multiple level visualization techniques improve understanding of processed and analysed data. Furthermore, these techniques can support security analytics, incorporating recent visualization technologies such as, 3D, 4D and augmented and virtual reality for visualizing complex processed data. Lastly, constructing new dataset comprising present structure and unstructured data from various recent technologies like IoT, 3D printing, smart cities, and other connected devices. Developed datasets should be validated with openly available existing datasets to handle multiple distributed threat all around the world.

2.9 Conclusion

In this chapter, real-ime big data technologies and machine learning with the possibility of anomalous detection have been discussed, followed by the examination of recent works in real time big data processing and anomalous detection from a use cases perspective. Examination of these use cases have helped identify the challenges associated with anomaly detection in real-time. This chapter also discussed the inadequacies and challenges of the current anomaly detecting approaches in the specified domain.

In section 2.1, Table 2.1 and 2.2 identified important features, pros and cons of various real-time big data technologies, these study able to finds the Spark real-time big data technologies which best suit for our further evaluation and enhancing existing framework on anomaly detection.

In section 2.2, various recently used anomalies detection techniques with different

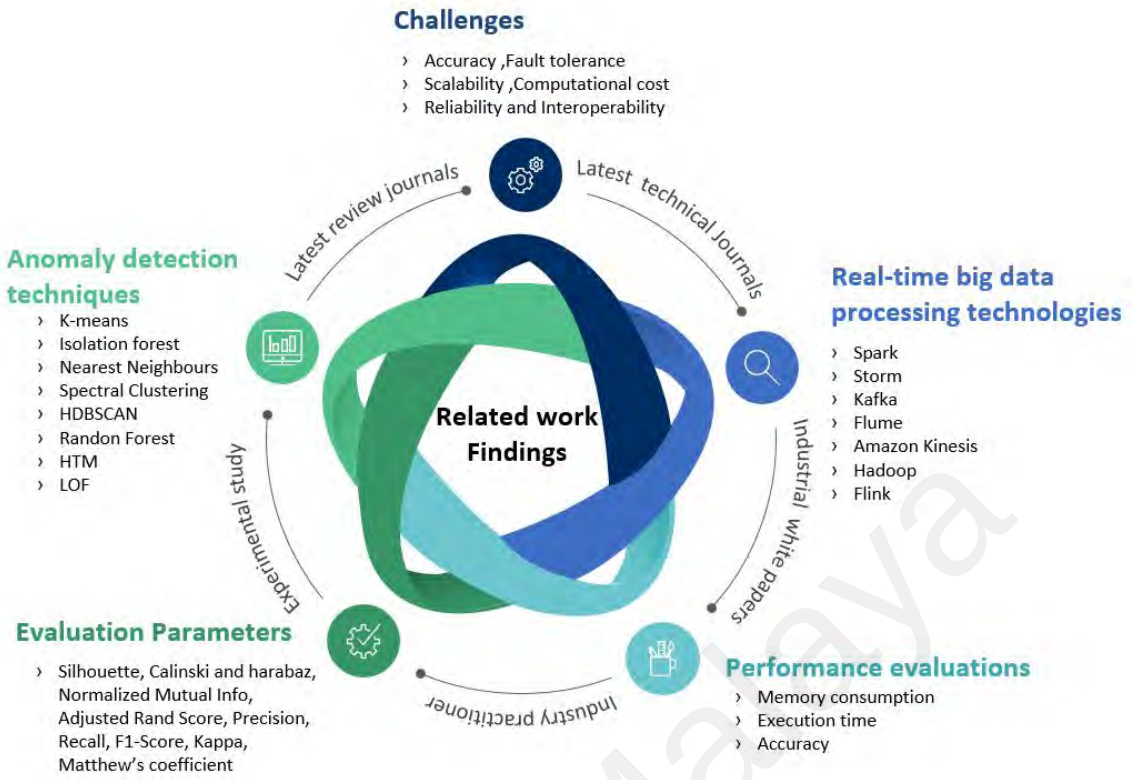

Figure 2.3: Findings from the literature review

domains had been tabulated along with their shortcoming like percentage of accuracy, consumption of longer time for training the model and incompetence in choosing the parameters (see Table 2.3).

In section 2.2.1, four different clustering algorithms that best suits this research domain such as K-means, Spectral Clustering, Isolation trees, and HDBSCAN had been discussed. Further investigations of these algorithms has been presented in chapter 3, where further experiment has been conducted with real-time dataset to prove their shortcoming in terms of accuracy, memory consumption and execution time.

In section 2.3, various anomaly detection domains those have used the big data technologies for their architectural design has been discussed (see Table 2.4); it has been revealed that Spark Streaming has been integrated with their platform for real-time processing among other existing big data technologies.

In section 2.5, some of the very recent works on real-time big data processing for anomaly detection have been critically evaluated (see Table 2.5). Furthermore, potential techniques and their architectural drawbacks have also been discussed, along with their evaluation methods on performance and accuracy. In addition, some of the commercial platforms of real-time big data processing for anomaly detection have been presented. The selected internal and external metrics for our evaluation have been presented in Table 2.7.

With regards to this research, two internal and external metrics have been selected after thoroughly analysing the cluster. In the internal validation, the silhouette index has been performed, which consists of the silhouette coefficient, and calinski and harabaz. In case of the externally validation of the model, an adjusted rand score method has been employed, which was also used to derive the accuracy for the performance parameters and normalized mutual info score method to externally validate the model.

Besides, a taxonomy of our approach has also been developed while the state-of-the-art approaches that pose research challenges in detecting anomaly using real-time big data technologies have also been identified. It is concluded that real-time big data technologies for anomaly detection is in its early stage of development and close attention must be paid to the presented challenges, especially in detection accuracy, memory consumption, and execution time to facilitate the adoption of big data technologies for anomaly detection, which will be a core component of the future computing landscape.

In the next chapter, we have presented experimental setup of the real-time analytics of clustering algorithms, dataset description, features selection, performance metrics, and the exploration of the outcome to substantiate the disadvantages of the existing algorithms. We have evaluated four different algorithms against three different openly available datasets for anomaly detection.

CHAPTER 3: PROBLEM ANALYSIS

This chapter aims to establish the problem that was highlighted in chapter 1 and conduct a deep investigation to show the impact of the clustering algorithms in terms of accuracy, memory consumption and execution time using the conventional method of real-time anomaly detection. This research work used different algorithms that aim to select the best techniques for handling the anomaly detection. In order to analyse the problem various experiments are performed with three different openly available datasets.

The rest of the chapter is organized as follows. Section 3.1 discusses the application and algorithms that are used to analyse the problem. Section 3.2 describes the performance-measuring parameters. Section 3.3 discusses the experimental parameters used to conduct the experiment. Section 3.4 presents results and its discussions. Discussions based on the empirical data analysis are summarized in Section 3.5.We reiterate the findings of the analysis conducted in Section 3.6.

3.1 Empirical study: Experimental setup

This section presents the empirical study conducted to establish the problem. We discuss the experimental setup including a real-time processing environment and the compute-intensive tasks that are developed to perform the analysis.

In this experiment, the accuracy, memory consumption and execution time of real-time analytics for anomaly detection have been evaluated by using four different types of algorithms such as K-means, Isolation Forest, Spectral Clustering, and HDBSCAN in a cloud platform. These four algorithms are selected for its capabilities in partitioning data points better than other machine learning algorithms. Additionally, selected algorithms also properly group the data points in ensemble methods and eigenvectors of laplacian. Data has been processed in real-time and the algorithms have been executed in four cycles,

this helps in the investigation of the accuracy, memory consumption and execution time of real-time analytics for anomaly detection using the traditional approach. The following subsection will further provide details of the experiment setup

3.1.1 Cloud Environment

A cloud platform has been used to carry out the experiments; the specifications of the cloud platform is provided in Table 3.1. We have performed our experiment in public cloud provided by a third party service provider whereby we had full access to their platform to install and monitor the entire cloud, check our execution time and load on the setup. Further, the service provider granted a dedicated dashboard to monitor many of process time and latency. Using a third party service provider is more cost effective as compared to a corporate cloud service provider such Amazon, Microsoft and Google. This is because of adopting the corporate cloud service provider for the experiment analysis were not possible due to many factors such as high rising cost for their service, unknown architectural setup and no administrator access to their platform to monitor their process time. In addition, permission to install any software on their platform was not possible.

We have investigated the effect of four different clustering algorithms in the perspective of accuracy, memory consumption and execution time by passing data to the algorithms in real time (i.e. data is being received constantly and the volume of that data is not constant). We used the adjusted rand score to calculate the accuracy of anomaly detection. Further, memory profiler has been used to evaluate the memory consumption and execution time.

Table 3.1: Specification of the cloud platform

Cloud Components	Specifications
Nodes	12
CPU	12 cores per node
Memory	32 GB memory per node
Operating System	CentOS 7.0.3

3.1.2 Algorithms

To conduct the experiment, four different frequently used clustering algorithms were selected, especially K-Means, Isolation Forest, Spectral Clustering and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). These algorithms were selected based on the literature review findings.

3.1.2.1 K-Means

A partitioning based clustering algorithm tries to find the best partitioning for data points in which intraclass similarity is maximum and interclass similarity is minimum. Many of the modern and well-known algorithms were developed in the extension of the k-means algorithm. (Amini, Saboohi, Ying Wah, & Herawan, 2014) The k-means algorithm is one of the most efficient clustering algorithms. K-means clustering is used to group the samples into various clusters. Cluster group can be classified into nonattack and attack group. Initializing the cluster centers points with the mean values obtained from known data points of appropriate groups (Bhuyan, Bhattacharyya, & Kalita, 2014). Cluster centroids are helpful in cluster or grouping various data points in certain categories. Furthermore, the K-means algorithm uses local optimization to determine the optimal number of clusters. The most widely used method is the Euclidean distance in which a small distance implies a strong similarity whereas a large distance implies a low similarity. However, in the value of k is lower than the number of natural groups, dissimilar instances are forced into the same cluster, and class dominance, which arises when a cluster contains a large number of instances from one class, and fewer numbers of instances from other classes (Bhuyan et al., 2014). Lastly, the cost function is not convex (Karami & Guerrero-Zapata, 2015).

3.1.2.2 Isolation Forest

Isolation forest algorithm uses an ensemble of trees, with the modification that the dimension to cut is chosen uniformly at random. Given a new point p, that algorithm follows that cuts and compute the average depth of the point across a collection of trees. The point is labeled an anomaly if the score exceeds a threshold; which corresponds to average depth being small compared to $log|s|$ where s is suitably sized sample of the data. Moreover, the merits of isolation forest are that different dimensions are treated independently and the algorithm is invariant to scaling different dimensions differently. However, isolation forest either produce too many false alarms or does not have a good recall and also, in many tasks that depend on detecting anomalies, the relevance of different dimensions is often unknown (Guha et al., 2016).

3.1.2.3 Spectral Clustering

Spectral clustering methods use the eigenvectors of the Laplacian to properly group the data-points (Langone et al., 2015). To cluster the data into k-subsets, the spectral clustering algorithm computes the largest k eigenvectors of the normalized Laplacian matrix from the affinity matrix, which represent the similarity between data points. Then it performs the popular k-means clustering algorithm on the resulting k-dimensional feature space. The advantages of spectral clustering are, it only requires a pairwise distance of data points and the algorithm does not require the distance to be metric (Y. Zhou, Yan, & Huang, 2007).A common method used in the spectral clustering is the Gaussian kernel. Nevertheless, when confronted with clusters of different scales, corresponding to a multiscale landscape potential, standard spectral clustering which the first k eigenvectors to find k clusters will fail. The further single scale may fail to correctly cluster multiscale data (Nadler & Galun, 2007).

3.1.2.4 HDBSCAN

Hierarchical Density-based Spatial Clustering of Application with Noise (HDBSCAN) perform the characteristics of DBSCAN with varying epsilon values thereby successfully identifying clusters with varying densities and outliers as unusual data points not belonging to any cluster (Duggimpudi, Abbady, Chen, & Raghavan, 2017). The HDBSCAN algorithm adheres well for scalability which can handle large dataset for analysis (Ošep, Voigtlaender, Luiten, Breuers, & Leibe, 2017). Conversely, the algorithm needs to either assign an instance to a cluster or mark it as an outlier. There is no quantification of the outlierness of an instance.

3.1.3 Datasets

A total of 3 datasets have been used in the experiment: (i) DARPA Intrusion Detection Dataset (created in 1999 by MIT Lincoln Laboratory) which contains real network data and labeled attacks. The size of the dataset is 1.5 GB; (ii) Mid-Atlantic Collegiate Cyber Defence Competition (MACCDC) dataset, is a cyber-defense competition created for students to test their cybersecurity skills (created in the year 2011). The dataset contains the data captured from the network in the event and also the attacks conducted by a specific team. The dataset is approximately 14.2 GB; and (iii) DEF CON 21 dataset, one of the world's largest hacking conference held annually. For this experiment, we used the DEF CON (2013) dataset, which has many challenges such as, lock picking, scavenger hunts and capture the flag. The datasets consists of the network packet capture of the event. We used the captured flag dataset for the experiment. The total size of the dataset is 100.8 GB

3.1.4 Feature Extraction

The process of feature selection facilitates in selecting appropriate features that might help to address the possible issues, where by the repeated, irrelevant and noisy features

will be disregarded.

Datasets considered for analysis might also comprise irrelevant or redundant features that might not help or hinder detections. Manually picking relevant attributes is also a possible scenario, however, it is tedious and time consuming process, particularly with the dataset that has strangely behaving data. Furthermore, having a lot of features might hinder the clustering processes, hence it is worth mentioning that it is crucial to pick appropriate features for detecting anomalies. In this regard, the feature selection approaches help to disregard the unnecessary features without impacting the outcome. There are number of feature selection methods, of which the information gain feature selection method is most popular for feature selection in class imbalanced problem (Wasikowski & Chen, 2010). In this method, each feature is given a score based on the statistical measure applied to the features. The scores given to the features help to determine their relevancy, as they are ranked based on the scores given to them. The information gain feature selection method is univariate and independently considers all the features, or with regards to the dependent variable.

Additionally, this method enables the data to explicate the significance of individual features for the detection, based on information gain, whereby, the most difficult and time consuming manual feature selection process is avoided. It is worth mentioning that web-servers and web-based applications have become easy targets of intruders and cyber criminals; hence it is critical to tighten the security of those targets.

Generally, web application users send queries while seeking information via HTTP request. The request generates HTTP traffic consisting features to identify the anomaly detection. While surfing websites, users might observe a standard pattern of behaviours such as, clicking the links, submitting the forms or prompting interaction scripts on the web pages, however, intruders do not follow the standard behaviours, rather they

Table 3.2: Features extracted for anomaly detection

No	Feature	Description
1	id.resp_p	Responding endpoint's TCP/UDP port (or ICMP code) By default standard value should be 80, if its run on the non-standard port like 443, 1947, 6667, etc are high chance for an anomaly.
2	method	HTTP Request verb: GET, POST, HEAD, PUT, OPTIONS, DELETE. Attacks using these methods are usually used in parallel to a GET flood, in order to try and attack less common areas in the server code. A POST request is usually larger than a GET request, and as a result, a large POST request is less suspicious than a large GET request, and more likely to get to the server un-noticed by the mitigation devices protecting it. OPTIONS helps for various types http flood attacks.
3	request_body_len	Actual uncompressed content size of the data transferred from the client
4	resp_mine_types	MIME types are a way of determining what kind of file you're looking at. PNG images - image/png, JSON files - application/json, JavaScript - text/javascript.

launch hacking tools to discover the weakness of the server to be exploited for extracting exclusive and private information (Juvonen et al., 2015) (Althubiti, Yuan, & Esterline, 2017) (Zolotukhin, Hämäläinen, Kokkonen, & Siltanen, 2014). The footprints of the users remain in the web server as http logs, which paves way for extracting several valuable information. The following are the various parameters present in the HTTP bro log: *ts, uid, id.orig_h, id.orig_p, id.resp_h, id.resp_p, trans_depth, method, host, uri, referrer, user_agent, request_body_len, response_body_len, status_code, status_msg, info_code, info_msg, filename, tags, username, password, proxied, orig_fuids, orig_mime_types, resp_fuids, resp_mime_types.* Based on information gain feature selection method, all the 27 features selected in this study were given scores. Out of which only 4 features were selected based on the top 4 scores (see Table 3.2), which have been used for real-time anomaly detection. These 4 features have been chosen based on the number of clusters k depending on the feature subset.

3.2 Performance Measuring Parameters

The following performance parameters have been used to analyse the accuracy, memory consumption and execution time of the existing real-time approaches for anomaly detection. Following are the three different performance parameters addressed in the objectives in chapter 1.

3.2.1 Accuracy

The accuracy measure that indicates how well the cluster points have been classified into the correct clusters; Adjusted Rand Index (ARI) value are used for calculating the accuracy of the algorithms. ARI is the best performing method for comparing two partitions. ARI computes a similarity measure between two clustering's by considering the pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering. It obtains a value of unity when the two partitions perfectly agree in the current situation, when two nodes are connected to the exact same set of other nodes and values of zero when there is only chance agreement between the partitions (Hoffman, Steinley, & Brusco, 2015). The raw RI score is then "adjusted for chance" into the ARI score using the following scheme: ARI = (RI - Expected_RI) / (max(RI) - Expected_RI)

3.2.2 Memory consumption

Memory consumption, which indicates the amount of memory taken in MB to execute a specific task. In our experiment, it is measured from when the data starts to read the incoming data until the output is received. Moreover, in our experiment the program loops over four cycles before the execution are stopped; Memory profiler is used to check and calculated the memory usage of the interpreter at every line. The increment column allow us to spot those places in the code where amounts of memory are allocated. Furthermore, It

allow to retrieve the memory consumption of a function in real-time, allowing to visualize the memory consumption of a given task over time.

3.2.3 Execution time

Execution time that specifies the time taken in seconds to complete the execution, including the time spent executing run-time or system services. In this context, the execution time is influenced by two factors, data size, and algorithm used. Similarly, memory profilier is used in the calculation of execution time for task.

3.3 Results and Analysis

This subsection illustrates the results acquired in the perspective of accuracy, memory consumption, and execution time using traditional methods for real-time anomaly detection.

3.3.1 Accuracy

To measure the accuracy for each algorithm, we used the adjusted rand score evaluation method. Figure 3.1 shows a comparison of the accuracy of four algorithms and three data sets.

From the 3.1 Figure, it is concluded that the Isolation Forest algorithm outmatches all of the other algorithms in terms of accuracy. Besides, K-Means and Isolation Forest yield better accuracy with smaller size datasets, whereas, Spectral clustering performs better with MACCDC dataset compared to other datasets, on the other hand, the spectral clustering performs better with a medium-sized dataset. Finally, the HDBSCAN algorithm performs better in the DEF CON dataset as against the DARPA and MACCDC datasets.

3.3.2 Memory Usage

Memory consumption is measured using the memory profiler package that is available for installation in Linux Cent OS. Figure 3.2 shows a comparison of the memory consumption

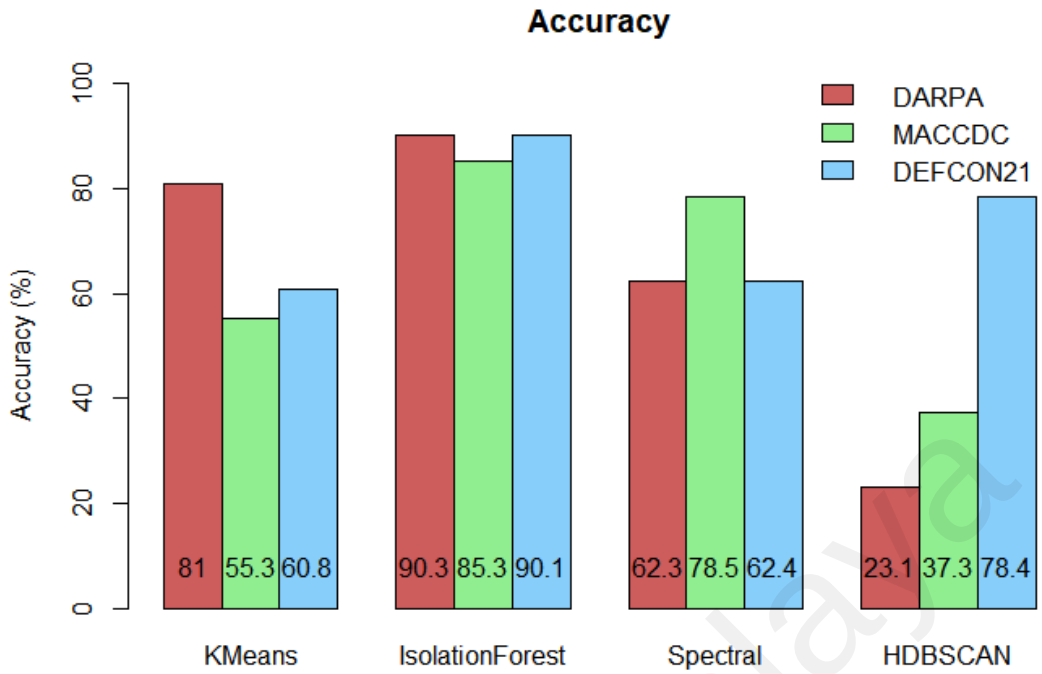

Figure 3.1: Accuracy of existing algorithms

for each algorithm for 4 cycles of execution.

When data is being streamed in real-time it is necessary to take into consideration the amount of memory consumption. As data is being received constantly and if the algorithm consumes a high amount of memory there is a possibility of an application crash. The highest memory is consumed by Isolation Forest for the DARPA dataset. The HDBSCAN consumes the lowest memory in the MACCDC dataset.

3.3.3 Execution time

The execution time is measured using the memory profiler package, the same tool used for measuring memory consumption. Figure 3.3 shows a comparison of the execution time for each algorithm for 4 cycles of execution.

From Figure 3.3 it is evident that Isolation Forest takes the least time in the MACCDC dataset. The highest time consumed is by HDBSCAN in the DARPA dataset. Whereas,

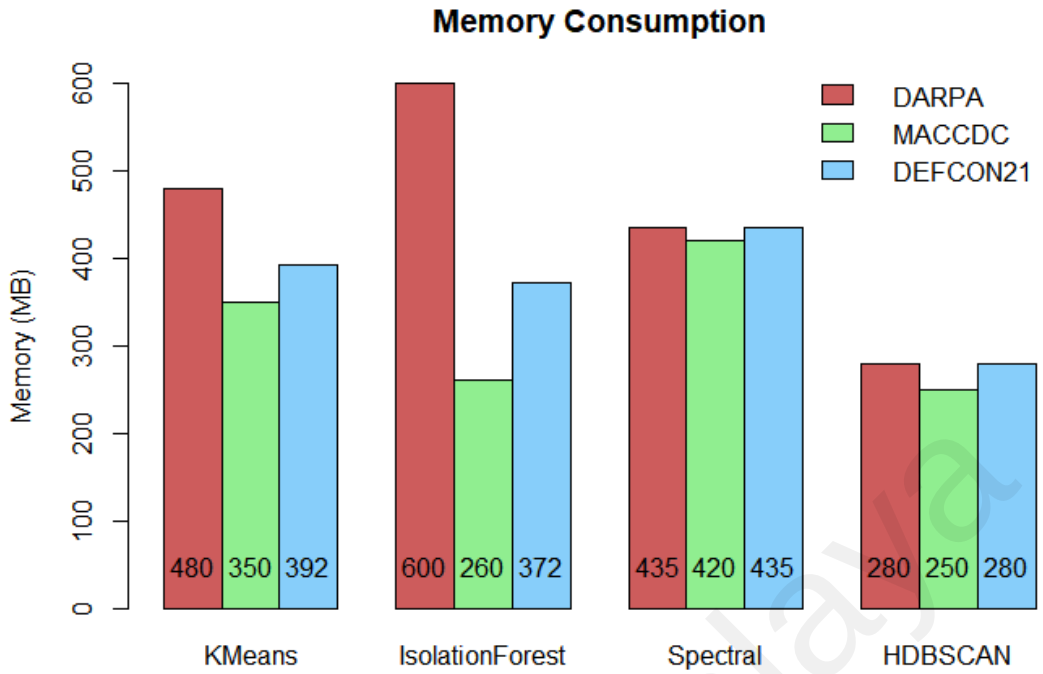

Figure 3.2: Memory consumption of existing algorithms

the K-Means has the similar time consumption on DARPA and MACCDC datasets.

We have illustrated our results as graphs in Figures 3.1, 3.2 and 3.3. Figure 3.1 indicates that the best accuracy is gained from the Isolation Forest algorithm. However, in terms of memory consumption, Isolation Forest consumes higher memory and would not be a great fit for real-time processing (see Figure 3.2). Furthermore, K-Means has a maximum accuracy of 81 percentage (see Figure 3.1) which is not sufficient for anomaly detection and has a high memory consumption of 480 MB (see Figure 3.2). Finally, Spectral Clustering and HDBSCAN have an accuracy of < 80 percentage in all datasets and hence not suitable for real-time anomaly detection (see Figure 3.1). Since, these algorithms lack accuracy, memory consumption or execution time, we have proposed a novel composite algorithm named SSWLOFCC (Streaming Sliding Window LOF Coreset Clustering).

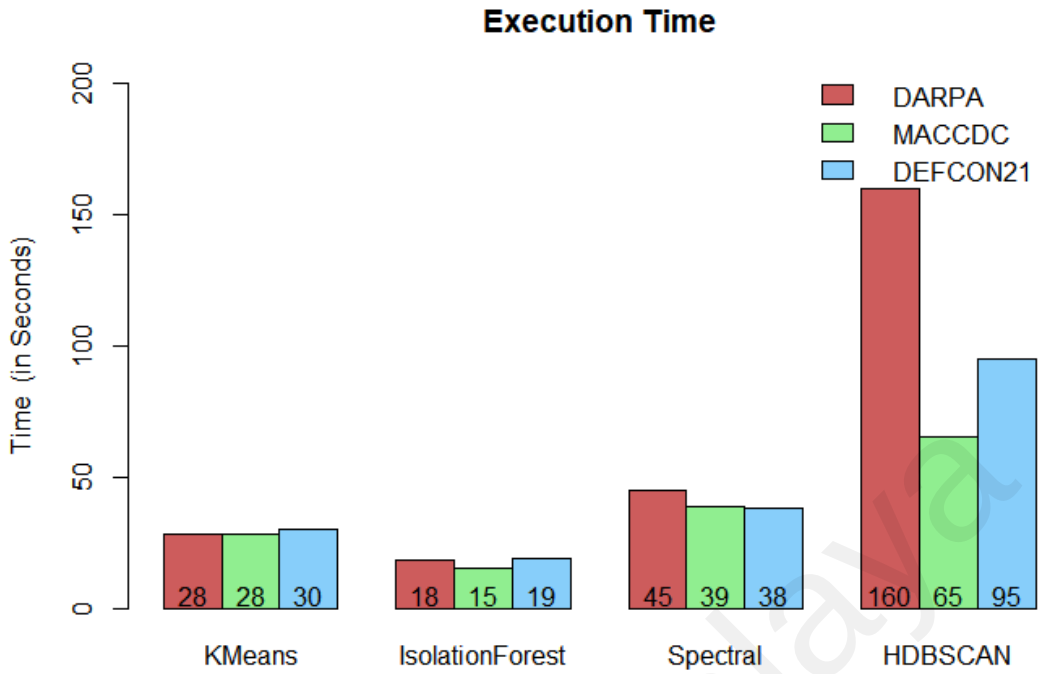

Figure 3.3: Execution time of existing algorithms

3.4 Discussions

The empirical results highlight the following:

- Uncontrollable streaming data reduce the accuracy of the anomaly detection in existing clustering algorithm.
- Performing real-time processing for anomaly detection without considering the incoming data streams can lead to an experiment system crash.
- The inefficient process execution took longer execution time which remarkably degrades the performance of the existing system.
- The existing traditional approach has merits, however, the significantly higher execution time and memory consumption for the processing are challenges that need to be addressed.

This section clearly shows the serious drawbacks of ignoring memory consumption and

execution time when detection the accuracy of a clustering algorithm for an anomaly detection.

3.5 Conclusion

In this chapter, we conducted experiments to analyse the accuracy, performance of the task for real-time big data. We evaluated the impact of performances based on two parameters; memory consumption and execution time.

Based on the analysis results, it is concluded that real-time big data processing for anomaly detection based on existing traditional approach can remarkable degrade the percentage of detection accuracy for clustering. In some cases, higher accuracy results in the longer execution time and higher memory consumption. Lastly, after receiving a few cycles of better accuracy for detection, it's failed to continuously produce better accuracy for next cycle due to insufficient memory and longer execution time for processing the task.

In the next chapter, we propose real-time anomaly detection based on clustering algorithm using big data technologies which comprise of composite clustering algorithm and various big data technologies to ingest and process the incoming real-time data. The adoption of our proposed solutions can produce higher accuracy, shorten execution time and less memory consumption for processing the real-time big data for anomaly detection.

CHAPTER 4: FRAMEWORK

This chapter aims to propose a real time anomaly detection framework and algorithms based on big data technologies. These proposed solution is expected to improve accuracy of anomaly detection in real-time, minimizing memory consumption and execution time using big data technologies and composite clustering algorithms. This research has demonstrated several components in the proposed framework and its functional capability. The proposed framework is named as real-time anomaly detection based on big data technologies (RTADBDT), which deals with the problems highlighted in the chapters 1 and 3.

Moreover, the proposed solution has been validated based on the parameters such as accuracy, memory consumption and execution time. Besides, the pseudo-codes of the proposed algorithms have also been presented in this chapter. Furthermore, the distinctive features of the proposed framework have also been discussed, followed by the presentation of a mathematical expression related to execution time.

The chapter is organized into six sections. Section 4.1 elaborates the proposed framework for real-time anomaly detection based on big data technologies. Section 4.2 illustrates the flowchart of real-time anomaly detection process. Section 4.3 presents the proposed composite clustering algorithms. The performance metrics algorithms and mathematical expression used for evaluations of the proposed framework have been presented in section 4.4, and the highlights the distinctive features of the proposed framework have been presented in section 4.5, finally the summary of the chapter has been presented in section 4.6.

4.1 Framework for Real-Time Anomaly Detection Based on Big Data Technologies

This section provides the details of the proposed framework for real-time anomaly detection based on big data technologies. Figure 4.6 depicts the key components of the framework: BroIDS, Flume, Kafka, Spark Streaming, MLlib, Matplot and HBase. In addition, this section discussed each of these components and also discusses other big data technologies used on real-time anomaly detection.

4.1.1 BroIDS

BroIDS is an open source network security monitoring tool that help to look into previously captured packet from captured files in real-time. Further, it can be used for ingesting, collecting, processing and analysing network data. This tool extensively inspects all network traffic for suspicious activities. The network data is stored in many multiple log files such as, HTTP log, DNS log, and SMTP log. One of largest strength of BroIDS is its ability to turn network events into actionable or useful metadata that provides the context which is the key to quickly detect potential threats (P.R, 2019) (Koning, Buraglio, de Laat, & Grosso, 2018). Figure 4.1 explain the function of BroIDS in the proposed framework as an individual component.

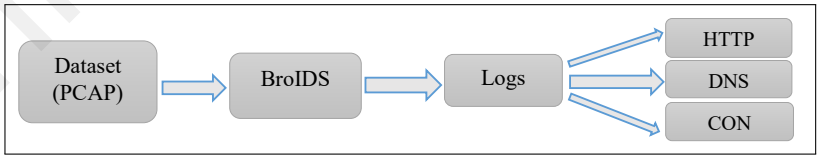

Figure 4.1: Implementation of BroIDS in the proposed framework

In the proposed framework, the BroIDS has been used to read the datasets from pcap files and retrieve the data in individual log files, which generates conventional real-time log files. Besides, the data in many individual files such as DNS log, HTTP log, and SMTP log have been extracted. This experiment has focused on http log used to identify applications

or services that perform http requests. The http log has been selected based on the most recent attacks in web-server and web-based applications. Further, it is crucial to strengthen web service applications to protect from security threats. Http traffic contains various set of attributes, not limited to application request, information related to the senders and receivers (see chapter 3 for feature selection details on http log).

4.1.2 Flume

Flume is a distributed, reliable service that is capable of efficiently collecting, aggregating and moving huge amounts of log data in real-time (Du, Liu, Liu, & Chen, 2014) (Shu, Chen, & Sun, 2017). Flume is robust, fault tolerant and provides many failover and recovery mechanisms.(https://flume.apache.org/).

The Flume facilitates the availability and reliability for the real-time data. Furthermore it reads and tracks the incoming packet data blocks that are handled by SourceAgent. In addition the SourceAgent employs a protocol to relocate the records to RecevingAgent. The Flume is highly effective in handling high volume and velocity of data from a number of applications. These data can be forwarded to other components for processing. Furthermore, Flume agent is capable of creating various channels to process the incoming data and concurrently store raw data for further analysis (Maarala, Rautiainen, Salmi, Pirttikangas, & Riekki, 2015). The availability will be enhanced by the multiple channel selection methods and at the same time the network overhead will be minimised (Shu et al., 2017). Figure 4.2 further illustrates the process flow of flume from sourceagent to receivingagent. It is possible to add many channel as required in order to ease the flow in the Flume. The channel is also called as Flume collector agent.

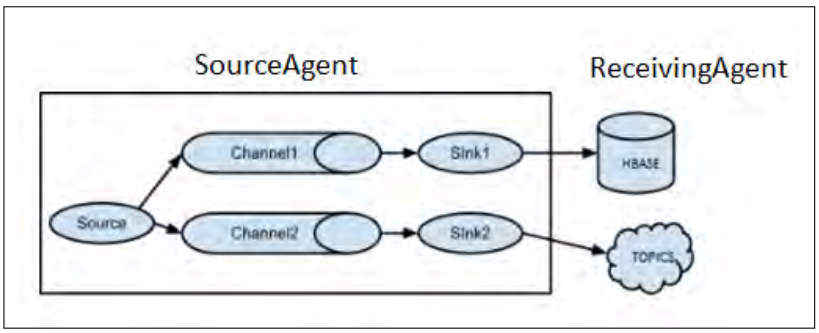

Figure 4.2: Flume Source Collection Architecture

In this work, the Flume has been used to read the log files that are being generated in real time by BroIDS. Moreover, some of the components in the Flume agent have been redesigned to introduce the load balancing using master node (Shu et al., 2017). This modification enables each collector node to fully utilize the memory for process and also this strategy plays vital role to monitor the usage of memory in real-time. Lastly, Flume is used to write the data being read into a Kafka topic for further process.

4.1.3 Kafka

Apache Kafka is a distributed streaming platform, which publishes and subscribes to streams of records. Kafka is similar to a message queue, where data received is stored and retrieved whenever required. Further, it delivers better queuing real-time data streams system with reliable, high-throughput and low-latency (Ranjan, 2014). The data stream from Kafka topics can be received by distributed analytic platform for further processing and implementing diverse analytics capabilities (Fang et al., 2016). Furthermore, most of the real-time systems attain various benefits such as, developing a message passing system to collect information for processing and storing the event in real-time. However, it is crucial to meticulously validate the memory consumption capabilities of message passing system prior to employing into any framework (Wiatr, Słota, & Kitowski, 2018).

Kafka comprises a repository of messages, categorized into different topics, each category has been divided into numerous partitions, which comprises of well-arranged,

absolute sequence of messages. Furthermore, the messages in a partition are labelled with exclusive and sequential IDs. Multiple nodes are disseminated over Kafka clusters in order to accomplish fault-tolerance. In user group, one user is entitled to obtain the accurate message order from any partition of topic category (X. Liu, Iftikhar, & Xie, 2014). The comparison of Kafka with other messaging systems has revealed that the former is effective in handling clustering, multiple queuing, client coordination and fault tolerance for real-time data collection (Ta, Liu, & Nkabinde, 2016). Figure 4.3 displays the topic creation in Kafka which is used in the proposed framework for data pipeline.

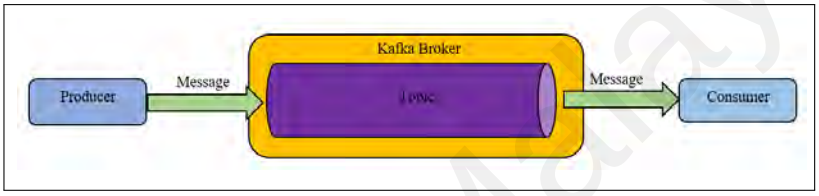

Figure 4.3: Topic creation in Kafka

A study has modified the KafkaProducer API call cost for Garbage collector, which optimizes the kafka topics to reduce the excessive memory consumption and low latency (Wiatr et al., 2018). Kafka as a message queue for the data from Flume to be written to the Kafka topic and for a Spark to read the Kafka topic data for further processing.

4.1.4 Spark Streaming

Spark Streaming is a high performance, scalable, in-memory and fault-tolerant real-time data processing framework. It is capable of processing massive data volume ingested from many different data sources like Kafka, Flume, Kinesis and etc (Son et al., 2016). The Spark Streaming is effective in providing an illustrious abstraction known as discretized stream or DStream, signifying an uninterrupted stream of data. DStreams are generated either from input data streams from sources such as Kafka, Flume, and Kinesis, or by applying high-level operations on other DStreams. Generally, a DStream is characterised

as an array of Resilient Distributed Dataset (RDDs). Fundamentally, Scala, Java or Python are the most common programming platforms of Spark streaming API. This study has used Scala for implementing algorithms. The reason for selecting Scala relies on its iterative computing task, and its capability of supporting a variety of data sources and fault-tolerance (Apache, 2019) (Maarala et al., 2015). Moreover, it is capable of directly processing the complex machine learning algorithms on data streams for analysing.

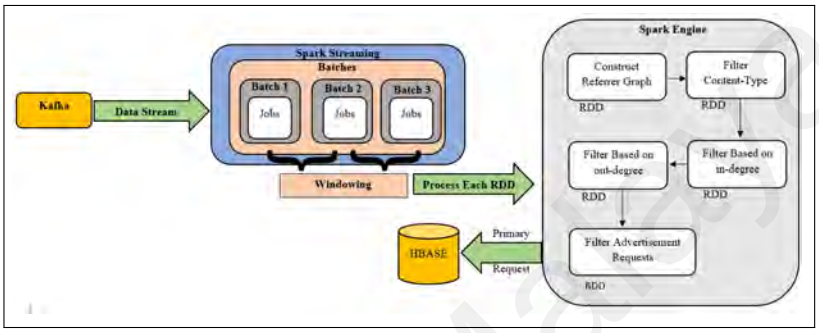

Figure 4.4: Work flow of Spark streaming processing

The above Figure 4.4 describes how Spark Streaming process the streamed data. Initially, Kafka sends data streams into Spark Streaming. Spark streaming executes the data using multiple batches, each batch has multiple jobs. A special feature in spark streaming provides capabilities of windowing, which is combining and executing 2 batches at the same time, this, in turn, improves the performance of spark streaming. Each window is processed as an RDD in the spark engine. The Spark engine consists of 5 processes, which are construct referrer graph, filter content-type, filter based on in-degree, filter based on out-degree, and filter advertisement requests. After these processes have been completed, the RDD can be stored into a Hadoop Distributed File System (HDFS).

In this work, Spark streaming processing is used for processing of network data to detect anomaly in real-time. To begin, Spark streaming ingests stream data from Kafka topic, which acquires data from the simulated http dataset. Moreover, inbuilt methods from Spark have been applied to enhance the retrieval of data. Subsequently, Spark streaming

processes the data stream into Dstream, which comprises a number of batches. Each batch in the Dstream contains data collected from 30 seconds interval. Sliding interval in Spark streaming enables to maintain a stable state for incoming data. Dstream generates the RDD for every slide interval of window time. Lastly, Spark Engine facilitates to process the proposed algorithms into RDDs.

4.1.5 Spark MLilb and Scala

Machine learning library (MLlib) is a standard module of Spark, which provides various statistical, optimization and primitive operation features. Further, it supports various high level API in Scala, Java, Python, and R Programming language. In ddition, MLlib comprises of various supervised and unsupervised algoritahms, which can produce high performances and scalable applications. The proposed Streaming Sliding Window LOF Clustering Coreset (SSWLOFCC) algorithms are implemented in the Spark MLlib module.

MLlib integration with Spark provides numerous benefits to any distributed application, some of the advantages are: the Spark iterative computation architectural enables large-scale machine learning algorithms to achieve high level efficiency in results, and Spark.ml API for pipeline offers developers with extensive range of new module to integrate with their architecture (Meng et al., 2016).

In the proposed framework, garbage collection method has been restructured to leverage algorithmic optimizations of MLlib and retrieve data from the Kafka topic and apply the algorithms. Furthermore, Scala has been used as the primary programming language to implement the proposed algorithms in to Spark streaming framework. In addition, Apache Spark has been built using Scala programming languages. These enable the modification and redesigning of the entire Spark Streaming architecture into the proposed application using Scala programming. Besides, Scala provides better interoperability with java and other high level programming languages. Lastly, Spark provides Scala a superior advantage

as against other programming languages to handle data stream in real-time and enable Scala to do robust data processing of Spark computational engine.

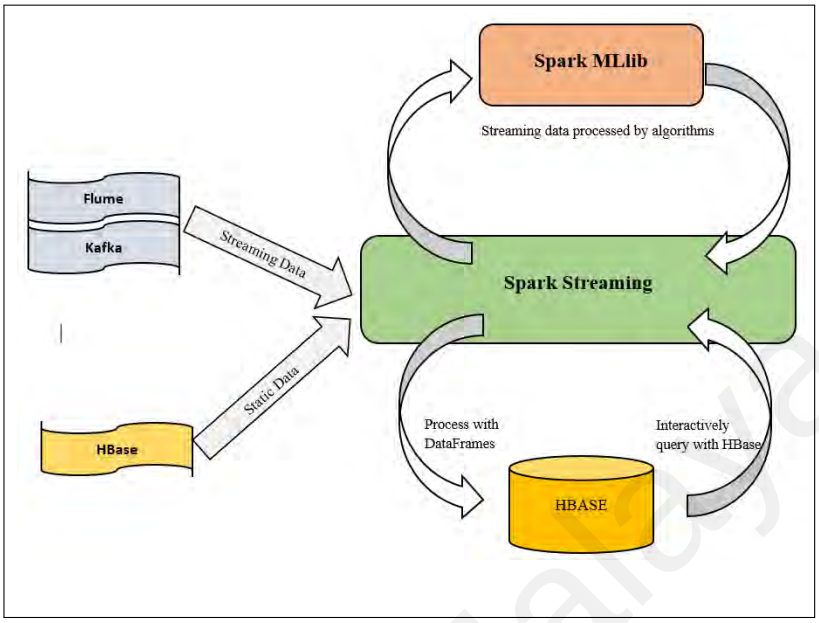

Figure 4.5: Interaction between Spark streaming and Spark MLlib

In the above Figure 4.5 describes the interaction of Spark Streaming and MLlib, there are two types of data sent to Spark Streaming. The streaming data, which is received from Kafka and Flume. The static data received from HBASE. When Spark Streaming receives this data, if the data is a streaming data, the models are processed with live data in Spark MLlib and store via Dataframes in HBase. If the data received is a static data, Spark MLlib will use the existing mechanism and HBase will interactively query the data.

4.1.6 HBase

HBase is the open source distributed storage system formed after Google's Bigtable. It provides modern API in real-time to read and write access to big data storage. Further, it has efficient mechanism of built-in data buffer, MemStore to assemble data in-memory prior to saving in any database (X. Liu et al., 2014). In this work, HBase receives the processed RDD streams, directly connected to the built-in data buffer to be stored in database.

4.1.7 Matplotlib and Python

Matplotlib is a Python 2D library used for plotting. It is capable of producing a variety of plots, histograms, power spectra, bar charts, error charts, and scatterplots (John Hunter, 2019). Python is an open source programming language that is powerful, fast, and user friendly, which omnipresent and is easy to learn(Software, 2019). Beside, pipe() and other packages have been used in this study to integrate some of the JVM scala like jython with python.

The matplotlib has been used in this study for plotting the results of the proposed algorithm. Moreover, a diverse array of big data technologies has been employed to build the novel proposed framework. Each component in the proposed framework has played a substantial role in extracting the significant results.

A variety of components have been configured and modified during the experiment in this research. The table 4 describes those components used along with their version numbers in the proposed framework.

Table 4.1: Tools used in the proposed framework with their version number

Components	Version	Usage
BroIDS	2.4.1	Read the pcap files for real-time simulation.
Apache Flume	1.6.0	Get data from log files and pass to Kafka
Apache Kafka	2.1.0	Queue the log data for spark to read
Apache Spark	1.6.0	Read the data from Kafka and stream the data
Scala	1.6.0	Programming language used to code spark streaming
Spark MLlib	1.6.0	To apply machine learning algorithms
Apache HBase	2.0.2	To save the results for further analysis
Matplotlib	3.0.0	Visualize the processed data
Python	3.7.0	To help plot the results in matplotlib

Each component has been utilized to achieve the results. The components work with the above discussed version numbers, however, there may be compatibility issues with different versions of the tools, and they may not work as expected.

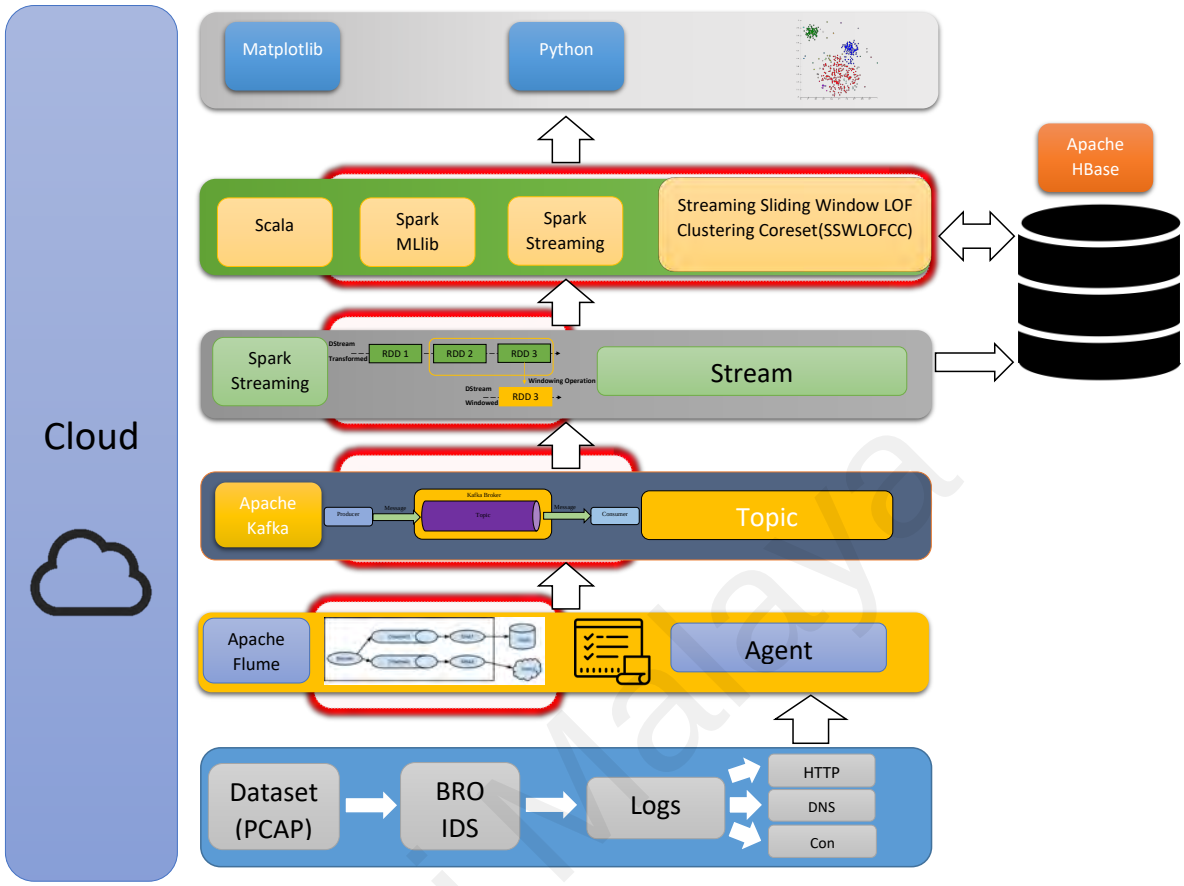

Figure 4.6: Proposed framework for real-time anomaly detection based on big data technologies

Figure 4.6 elucidates the proposed framework. In the initial step, the data is received in pcap format and is read using BroIDS. The BroIDS divides the files into specific log files, from there, the HTTP log files are sent to Flume for processing. The above mentioned step transmits the real-time data to Apache Flume. It reads the incoming data into the http log from BroIDS and sends the data to Kafka. Furthermore, Kafka has the advantage of traditional log aggregators and messaging system. Next, the Kafka queues the data into a kafka topic. Following this, the Spark reads data from the kafka topic and the sliding window is applied to combine Spark Streaming jobs. After that, spark stream RDD is further processed using the proposed algorithms for anomaly detection which is module of Spark MLlib. The data is visualized using matplotlib package. Lastly, the data from Spark

Streaming is piped into HBase and the output of the proposed algorithm is also stored into HBase for further analysis.

4.2 Real-Time Anomaly Detection Process Using Flowchart

The proposed framework has been illustrated using flow diagram. Figure 4.7 shows a flow of steps that are required to perform real-time anomaly detection using big data technologies.

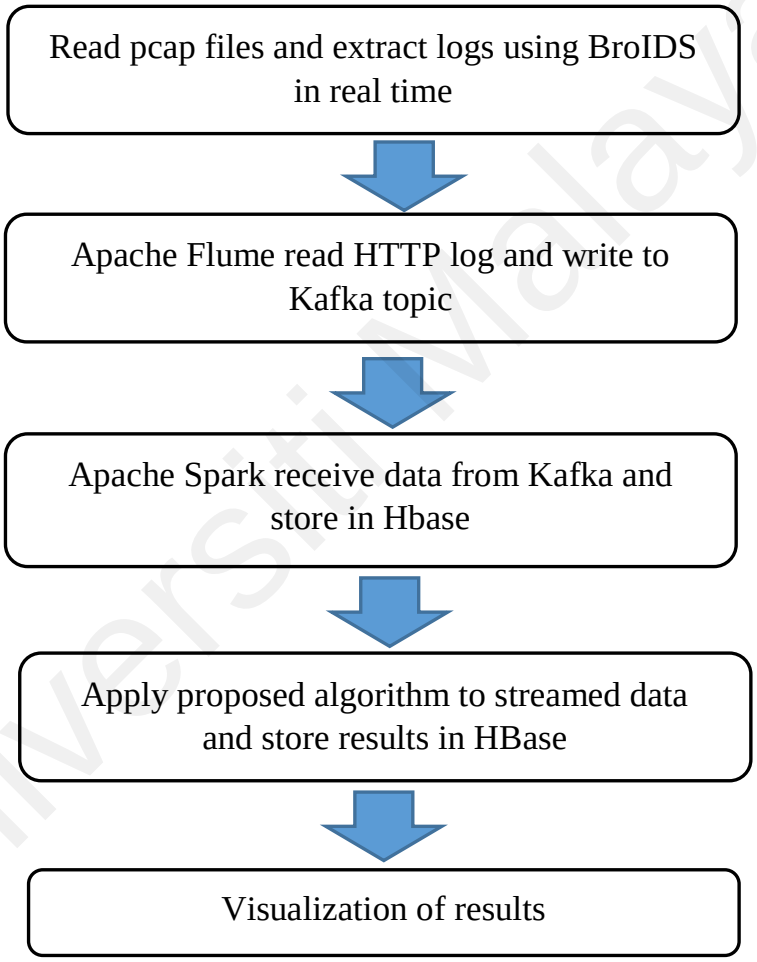

Figure 4.7: Flow diagram of the proposed RTADBDT framework.

The process starts with BroIDS reading pcap files in real time, followed by Apache Flume reading HTTP log files and writing them to a Kafka topic.Then, Apache Spark reads data from Kafka and the streamed data is stored in HBase. Later, the proposed algorithm

is applied to the streamed data and the results are stored in HBase. Finally, the results of processed data are visualised for anomaly detection.

4.3 Proposed Algorithms

In this section, the elements used to develop the proposed Streaming Sliding Window LOF Clustering Coreset (SSWLOFCC) algorithms have been discussed. The proposed algorithms are implemented into Spark MLlib component using Scala programming.

Algorithm 1 Flume Pipeline

1: $x \leftarrow$ sources
2: $y \leftarrow$ sinks
3: $z \leftarrow$ channels
4: Configure the source
5: Configure sink 1 (Kafka)
6: Configure sink 2 (HBase)
7: Use channel 1 to buffer events in memory (Kafka)
8: Use channel 2 to buffer events in memory (HBase)
9: Bind source and sink to channel (Kafka & HBase)
10: *props* $\leftarrow$ setup properties for load balancing
11: *hosts* $\leftarrow$ list hosts
12: Pair each host with port
13: props.put(hosts)
14: Create client with load balancing properties

Algorithm 1 Flume Pipeline defines the sources, sinks and channels initially. After, the sources are configured, the sinks are configured (Kafka & HBase), channels are buffered events in the memory. Further, the sources and sinks are binded to the channel for kafka & HBase. Next, we define the properties for load balancing and list the hosts. After, we pair each host with the respective port number. Finally, we put the properties for the hosts and create client with load balancing properties.

The proposed Streaming Sliding Window LOF Clustering Coreset (SSWLOFCC) is a clustering-based network anomaly detection algorithm with the use of big data technologies. Algorithm 2 is the pseudocode for the proposed algorithm. Initially, the streamed data from Kafka has been defined as streamData. While the streamData is not empty, a representing

Algorithm 2 Streaming Sliding Window LOF Clustering Coreset (SSWLOFCC)

INPUT: Streaming Data from Kafka Topic
OUTPUT: Cluster visualization image
streamData=stream data from Kafka
while (streamData != empty) **do**
    $a \leftarrow$ current streaming data ▷ data streamed in specified time limit
    $b \leftarrow$ next streaming data ▷ next data to be streamed
    windowSize $\leftarrow$ 30 ▷ window length duration for sliding window
    slidingInterval $\leftarrow$ 30 ▷ how long each interval should last
    windowedStream $\leftarrow$
        streamData.reduceBykeyAndWindow((a,b) =>(a+b),
        windowSize, slidingInterval)
    Extract features ('id.resp_p', 'method', 'resp_mime_types',
        'request_body_len')
    bro_matrix = streamData[features] to matrix and normalize
    lof = LocalOutlierFactor (neighbors = 5)
    lof.fit(bro_matrix)
    lof_df = streamData[features][lof.fit_predict(bro_matrix) == -1] ▷ refer to algorithm 3
    lof_matrix = to matrix (lof_df)
    aggclustering = AgglomerativeClustering(n_clusters=4)
        .fit_predict(lof_matrix) ▷ refer to algorithm 4
    pca = PCA(n_componenets=3).fit_transform(lof.matrix))
    lof_df['x'] = pca[:, 0]
    lof_df['y'] = pca[:, 1]]
    lof_df['cluster'] = aggclustering
    Construct the k center clustering coreset
    |C| = O(klog 2(n))
    Costp(C) ≤ 8·Costp(OPT)
    2·Costp (C) x 2·Costp (C) x
    ϵCostp (C)/(8 $\sqrt{d}$) x ϵCostp (C)/(8 $\sqrt{d}$) x
    Costp (C) = maxp ϵP { dist(p,C) }
    |Coreset| = O[($\sqrt{d}/\epsilon$) d·k·log 2(n)]
    Costp (A) = dist(p,A) ≤ dist(q, A) + ($\sqrt{d}$)(ϵ/8 $\sqrt{d}$)Costp (C) ≤
        Cost S(A) + ϵCostp (OPT) ≤ Cost S(A) + ϵCostp (A)
    (1- ϵ)Costp (A) ≤ Cost S(A)
    Costp (A) = dist(p,A) ≥ dist(q, A) - ($\sqrt{d}$)(ϵ/8 $\sqrt{d}$)Costp (C) ≥
        Cost S(A) - ϵCostp (OPT) ≥ Cost S(A) - ϵCostp (A)
    (1+ ϵ)Costp (A) ≥ Cost S(A)
    |Coreset| = O[($\sqrt{d}/\epsilon$) d·k·log 2(n)]
    Scatter plot with matplotlib
    Dt = current date
    Id = Row id
    Cf = HBase Configuration
    Parse streamData and SSWLOFCC(produced result) into hbase object

streaming data, b represents the next streaming data, windowSize represents the length sliding window, slidingInterval indicates the duration of the interval, and windowStream function is for applying the sliding window.

After performing the sliding window, the features such as, *id.resp_p, 'method', 'resp_mime_types', 'request_body-len'* have been extracted from the data. Further, a variable has been defined to hold the features of the stream data and to normalize the results. After that, the number of neighbours has been defined, and call the LOF fit and fit_predict method on the data. The data is further processed by converting it to a matrix and applying agglomerative clustering with defined clusters and use the fit_predict method on the lof_matrix. Furthermore, the PCA has been applied for dimensionality reduction of the data using the fit_transform method.

In addition, the k center clustering corset has been used for efficient outlier detection on the model. Finally, a scatter plot has been created on the x and y points using matplotlib; and the streamed data and the SSWLOFCC produced results are saved into HBase objects.

Algorithm 3 Local Outlier Factor

```
1:  INPUT: k, dp
2:  OUTPUT: lof values
3:  kDistance (dp, pt)
4:  reachabilityDist(pt)
5:  lof = null
6:  for each data point pt do
7:      KNN = kDistance(dp, k)
8:      lrd = reachabilityDist(KNN, k)
9:      for each p in KNN do
10:         lofhold[i]=sum(lrd[o ∈ N(p)])/lrd[i]/|N(p)|
11:         lof = max(lof, lofhold)
    return lof
```

Algorithm 3 takes 2 inputs, which are the k value and data points. Two methods have been defined that calculate the k distance and the reachability distance. For each data point, two variables are stored and named as KNN and lrd, which call the methods kDistance and reachabilityDist respectively. Furthermore, for each point in KNN a temporary sum of

the lrd is held and the max of the lof variable and the temporary lof variable are acquired. Finally, the lof data is returned. This algorithm provide information about their neighbours data points on outlier detection with distance function. At the same time this algorithm detect the lower density of data points than its neighbours data points.

Algorithm 4 Agglomerative Clustering

```
1: INPUT: X (data points)
2: OUTPUT: cluster points
3: Dist_matrix( Dmeans(C_i, C_j) = ||μi — μj||^2)
4: for each datapoint in X do
5:     while only a single cluster remains do
6:         Cluster = datapoint[i]
7:         ClusterMerge= merge two closest clusters
8:         Update Dist_matrix(Cluster+ClusterMerge)
```

In algorithm 4, which is the agglomerative clustering, X is defined as the input of data. Initially, it is necessary to compute the distance matrix for X. After that, it is essential to let each data point in X to be a cluster. Finally, the two clusters are merged and the distance matrix is updated. This process repeats itself until only a single cluster remains.

Lastly, the proposed algorithms have been restructured and improved based on various operation functionalities of sliding window, Linear Outlier Factor (LOF), Agglomerative clustering and Coreset technologies.

4.4 Performance Evaluation Metrics of the Proposed Framework

This subsection explains the evaluation of the proposed RTADBDT framework. To validate the RTADBDT, the accuracy, memory consumption and execution time are addressed as per objectives listed in the chapter 1 and section 1.4.

4.4.1 Accuracy

Accuracy refers to the measurement of how well the cluster points have been categorized into the appropriate clusters. For the evaluation of the proposed algorithms for clustering and anomaly detection, the internal and external cluster evaluation techniques have been

used (see Chapter 2). For internal cluster evaluation technique, Silhouette Index and Calinski and Harabaz, have been used, and for external cluster evaluation technique adjusted rand score and normalized mutual info score methods have been used. Additionally, to evaluate the accuracy of anomaly detection of the proposed algorithms statistical techniques such as, confusion matrix, Precision, Recall, Kappa, Mathew's correlation coefficient and F1-score methods have been used. All these methods have been described in Chapter 5 under evaluation methods section. These are most commonly used evaluation methods for clustering and anomaly detection.

4.4.2 Memory Consumption

The amount of memory taken in MB to execute a specific task is called memory consumption. This study has used a modified openly available package in Linux named memory profiler, which enables the user to monitor the memory consumption. In order to get an accurate reading, the proposed framework has been monitored from the first phase that is when BroIDS starts to read the log files until the results of the proposed framework have been gotten. The proposed framework has been looped over four cycles to get accurate results.

Initially, Algorithm 5 runs by importing necessary packages such as tracemalloc and matplotlib. After, the script, isPython, pid, toMB and Outputfile are defined. If the script is a python script, then while the pid still exists, we define stat which holds the value of tracemalloc take_snapshot that is used to get the memory usage of the script. Furthermore, the memory needs to be converted into MB therefore, we divide the stat.size/toMB. After that, we insert the mem into the Outputfile, print the total memory usage and plot using matplotlib. Lastly, if the script is not a python script, then the algorithm will return a statement that the script is not a python script.

Algorithm 6 is used to calculate the amount of memory consumed per second. We

Algorithm 5 To calculate RAM usage

```
1: import tracemalloc
2: import matplotlib
3: script ← script to be executed
4: isPython ← check to see if script is a python script
5: pid ← os.getpid()
6: toMB ← (2**20)
7: Outputfile ← memory.txt
8: if script = isPython then
9:     while pid do
10:        stat ← next(filter(lambda item: str(item).startswith(script)
11:        tracemalloc.take_snapshot().statistics('script')))
12:        mem ← stat.size / toMB
13:        Insert mem into Outputfile
14:    Print total memory usage
15:    Plot using Matplotlib
16: else return 'script is not a python script'
```

Algorithm 6 To calculate memory consumption per second

```
1: mem ← Output of Algorithm 5 (total memory consumption)
2: total_time ← Output of Algorithm 7
3: MemPerSec ← mem/total_time
4: Display MemPerSec
```

define mem and total_time to hold the output we receive from Algorithm 4 and Algorithm 5 respectively. Further, we define the MemPerSec to hold the mem divided by total_time, which will give the memory consumed per second by the script. Finally, we display the MemPerSec.

4.4.3 Execution Time

Execution time can be defined as the time taken in seconds to complete an execution. The modified memory profiler package has been used to get the reading of the execution time. The time in seconds has been monitored from BroIDS starts to read the log files until the results of the proposed framework have been acquired. The proposed framework has been looped for four cycles to get more accurate results.

The accuracy, memory consumption and execution time have been employed as major performance evaluation metrics, which have been as used by many big data processing and

anomaly detection researchers. For real-time application, memory consumption plays a vital role for processing. Each metric is crucial for evaluating the proposed framework and it has significant impact to determine the efficiency of the proposed framework.

Algorithm 7 To calculate execution time

```
1: import time
2: SSWLOFCC()                          ▷ calling function to be executed
3: time.clock()
4: Display time.clock()
```

Algorithm 7 is used to calculate the execution time to run the script. Time package from the python library was imported here. Further, the method executed to call time.clock() package to receive the time taken to run that specific method. Finally, the results of time.clock() was displayed in seconds.

The study has calculated the execution time of entire framework using the derived mathematical formulae given below. The Spark application is performed by continuous numbers of Spark jobs. Then, aggregating all the jobs from the beginning to end will calculate the entire execution time for the framework.

$$SparkExecutiontime = \sum_{x=0}^{x=N} ETjob_x \tag{4.1}$$

The Spark job execution is known as a graph and each node in the graph is expressed as a stage. On each occasion, the operation needs data exchange and then a new stage is generated. Spark job execution time is calculated by totalling the estimated execution time of all the stages. This concept is represented as:

$$ETjob_x = STjob_x + \sum_{k=0}^{k=NS_x} stageET_x^k + CTjob_x \tag{4.2}$$

Table 4.2: Symbols and explanations

Symbols	Explanation
$ETjob_x$	Job Execution time 'x'
$STjob_x$	Job 'x' Start up time
NS_x	Number of Stages in Job 'x'
$stageET_x^k$	Job 'x' Stage 'y' Execution time
$CTjob_x$	Job 'x' Clean up time

4.5 Distinctive Features of the Proposed Algorithms

In this section, various distinctive features of the proposed algorithms have been discussed (see Chapter 2 for details of several anomaly detection frameworks). Certain number of framework deals with anomaly detection using big data technologies. Nevertheless, these frameworks are incompetent and encounter multiple issues and challenges for real-time anomaly detection.

The proposed framework addressed solutions for the three major performance issues which are the accuracy, memory consumption and execution time. In addition, the proposed framework also support and provide the capabilities for additional features such as fault tolerance, scalability, and uninterruption of incoming data.

4.5.1 Real-Time Processing

In the proposed framework, Spark Streaming module has been modified for data processing and analytical results in real-time data received from http log file pipelined via Kafka topic. As described in chapter 2, it receives and processes terabytes of events for real-time analytics. The proposed framework has been evaluated with three different openly available datasets. Mainly, Spark Streaming receives the process via DStream abstraction. To begin the process, it is essential to establish streamingcontext which comprises Sparkcontext and Sliding interval time. Sliding interval sets the update window where, in coming data streaming are processed and streamingcontext object are working in parallel. In the framework, the input data source is an http log message generated from

BroIDS and received from Kafka messaging system. After the streaming computation logic is finished, the streamingcontext begins to process the incoming data. The proposed algorithms are applied for Dstream before it terminates the data streams. Lastly, these flow process are performed in-memory until it forces to store in the external disk storage and low latency provide assurances for real-time processing possible for the proposed framework.

4.5.2 Uninterruption of incoming data

In building a real-time framework, one of the crucial concern is uninterrupted data flow for processing. Any error or defect in pipeline flow will obstruct the process which will lead to system failure. To handle this, in the proposed framework Apache Flume has been employed to deal with huge moving incoming data for pipeline. Further, load balancing mechanism has been developed to operate the real-time data, which can provide reliable and available service that have abilities to collect and aggregate log data efficiently in instantaneous. These utilized approaches help to uninterruptedly provide the balanced real-time data to Spark MLlib module, hence the proposed algorithms are implemented for anomaly detection in the framework.

4.5.3 Accuracy

In the proposed framework, various composite clustering algorithms have been developed to be implemented into Spark MLlib components. These algorithms are capable to handle real-time anomaly detection for http log file. To begin with, windowing size are derived to consume the incoming data for processing. From these incoming data, the desired feature have been extracted to detect the anomaly. Modifying, improving and integrating these LOF, PCA, Agglomerative clustering and coreset algorithms functionalities yield increased accuracy in detection.

4.5.4 Memory Consumption

The proposed framework is effective in reducing the memory consumption for entire processing. Most of the real-time application suffer from memory issues. The proposed framework utilizes the Spark streaming in-memory functionalities, and algorithms were developed inside RDD module of the framework, which enables to consume less amount of memory. In addition, less computation quality of clustering algorithms further helps to reduce the memory consumption for the proposed framework. In Chapter 5 and 6, the memory consumption of three different dataset for proposed framework have been analysed and compared.

4.5.5 Fault Tolerance

In the proposed framework, Spark streaming component processes real-time data streams from Kafka topics and Flume, which again is convenient to establish fault-tolerance for any real-time applications. Moreover, node or process failure in Spark streaming are capable of detecting and recovering by itself, which is one of the biggest advantage of Spark streaming module. Fault-tolerance are achieved by RDD in stream, and RDD is logically portioned for each to operate at any point of time. In addition, master node keeps track of the all the tasks executed for any operation, which is occupied with metadata checkpoint to save all the necessary information of the streaming computation. These lead to recover from any failure of the node that running streaming application.

4.5.6 Execution Time

The proposed framework facilitates to minimize an execution time by building code inside the Spark MLlib module. This module runs the code in the real-time and also designed for distributed computation, which allows task to be performed by available executors in Spark engine. The executors are dynamically allocated according to required

tasks processing. The tasks are proceed simultaneously in the Spark engine throughout the operation. These lead to tremendously reduce execution time. In addition, data processing is performed in the in-memory, without any intermediary files for processing the task, whereby other ETL tools were used by the middle layer to process it. Lastly, Spark MLlib core receives the process via ML pipelines, which are high level API, developed under the spark.ml.packages. This helps the sequence of tasks to be performed in the framework, such as processing, feature extraction, integration of algorithms in distributed environment, visualization, and storage.

4.5.7 Scalability

In the proposed framework scalability is achieved as it interoperates with several interconnected modules like Flume, Kafka, Spark RDD, MLlib, HBase and lastly cloud platform for storage. Unique features in these modules provide robust and scalable environment to process complex streaming business logic for anomaly detection. Besides, Spark Streaming ingests data from numerous platforms like Kafka partitions, which decide the numbers of Spark parallelism operation and proposed algorithms are implemented into the RDD data frame of Spark Mllib. Lastly, Hbase stores the processed task in the storage table. These leads to high scalable functions for the proposed framework.

4.6 Conclusion

This chapter has presented the proposed real-time anomaly detection framework and algorithms based on big data technologies (RTADBDT) in the form of pseudo-code that help to elucidate understanding of the solutions.

A framework had been developed based on Spark Streaming, where Spark API is responsible for receiving and submitting jobs for processing among any node. The framework comprised of five major components (see Figure 4.6): Flume, Kafka, Spark

Streaming, Spark MLlib, and HBase which are flexible to receive the real-time data for processing. The Flume component had been modified to operate the load balancing of incoming data, which are continuously available for Kafka topics to send the flow data to Spark Streaming. Spark reply on YARN or Mesos for resource scheduling, which dynamically allocate and adjust the available resources to any jobs across applications. This leads to reduction in the memory consumption, and furthermore the composite clustering algorithms developed, support less memory computation for operation. Spark Streaming abstraction concept of Discretized Stream helps to build a stream of data divided into small batches. In Spark MLlib, the developed composite clustering algorithms are applied to Dstream to detect the anomaly in real-time. The processed Dstream had been plotted using matplot for visualization. Directly executing algorithm inside the Dstreams influences shorter execution time. The improved and integrated operation in the proposed algorithms enable higher accuracy detection of anomaly in http log.

The process flow of the implemented framework has been described, while the accuracy, memory consumption and execution time has been identified for performance evaluation metrics. Lastly, the distinctive features of the proposed framework has been discussed to determine their competitiveness.

The next chapter presents the details of the implementation of RTADBDT framework and algorithms. Likewise, various evaluation methods to verify the implemented framework and algorithms have been discussed. The data collected from the implemented framework and algorithms have been presented, which has been later used for critical analysis to be explained in Chapter 6.

CHAPTER 5: EVALUATION

This chapter presents the evaluation of data collected from the proposed framework real-time anomaly detection based on big data technologies. Further, this chapter discusses the experimental setup and dataset used to test the performance of the proposed algorithm and RTADBDT framework. This chapter also describes the evaluation methods used for measuring the accuracy of the algorithms and mathematical model for performance parameter. Lastly, the chapter furnishes the details of data collected for analysing the performance of the proposed algorithms and framework.

This chapter is structured as nine sections: section 5.1 explains the experimental setup and describes the procedure. Section 5.2 describes the dataset used in the performance analysis. Section 5.3 reports the process of data collection methods from the proposed framework. Section 5.4 details the performance evaluation methods: a) internal, b) external, c) statistical, d) execution time and e) memory consumption. Section 5.5 reports the data collected to analyse the impact accuracy of the proposed algorithms. Section 5.6, presents the data collected for evaluating the execution time of the proposed framework. Section 5.7, demonstrates the data collected for evaluating memory consumption of the proposed framework. Section 5.8, presents data collected for evaluating the proposed framework in terms of memory consumption per second. Finally the section 5.9 presents the conclusion for the chapter.

5.1 Experimental Setup

This section offers the information of the experimental setup of this study. To conduct the experiment this study has simulated all solutions on a cloud environment that consists of 12 nodes, where each of the nodes consists of 32 GB of memory and 12 CPU cores. The operating system in the nodes is Linux CentOS 7.0.3. Initially, the installation of the

required components, such as, BroIDS, Apache Flume, Apache Kafka, Apache Spark, Spark MLlib, HBase, Python and Matplotlib on the operating system is ensured. All of the proposed algorithms are implemented on the Spark MLlib component in Spark API. Figure 5.1 presents the flow diagram for proposed framework execution process. To conduct the experiments three different real-time data were simulated using BroIDS. These simulated real-time data have been processed using the proposed algorithms and framework. The testing results data of framework and algorithms were compared against the results obtained from problem analysis results on the particular performance parameter. In this framework, the three different datasets that were used in problem analysis were executed in the simulated environment and their results were compared for memory consumption and execution time. Data has been processed in real-time and the algorithms have been executed in thirty cycles. In addition, the memory profiler component is developed and installed in python to produce the results. As discussed in earlier chapters, primary concerns of this research are the accuracy of the algorithms, memory consumption and execution time for processing. Lastly, to calculate the accuracy the following methods are used: a) Silhouette Index, b) Calinski and Harabaz, c) Adjusted rand score, d) Normalized mutual info score e) Confusion Matrix, f) Precision, g) Recall h) F1-Score.

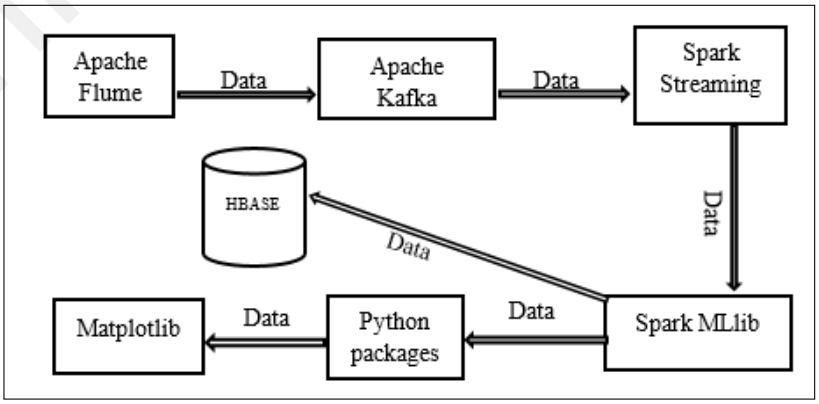

Figure 5.1: Execution flow for proposed framework

5.2 Dataset

In the use case, this research has used cybersecurity dataset to evaluate the performance of the proposed algorithm and framework. These datasets contain normal and anomaly network traffic data. Due to privacy and security concerns, an actual dataset is not available for analysis. Yet, openly available dataset enabled the study to evaluate the performance on these datasets. For this purpose, three different datasets have been selected, which are an old and new public dataset used for our experiment. Namely, Darpa Intrusion Detection dataset (DARPA), Mid-Atlantic Collegiate Cyber Defence Competition (MACCDC) dataset, and DEFCON21 dataset. These three datasets consist of network data which have regular and anomaly data. These three datasets have been selected based on the literature (see chapter 2). Moreover, these datasets are most suitable for evaluating anomaly detection using clustering. The DARPA dataset has been created by MIT Lincoln Laboratory which consists of real-world network data and labelled attack data. The total size of the dataset is 1.5GB. It comprises of the entire payload of each packet in tcpdump format. Further, it consists of 233,428 packets with various types of attack and regular network traffic.

The second dataset is Mid-Atlantic Collegiate Cyber Defence Competition (MACCDC) dataset, created for students to test their cybersecurity skills in cyber-defence competition (created in the year 2011). It consists of 264,973,151 packets which comprise of network data for analysis. Lastly, the third dataset is DEFCON21 dataset, one of the world's largest hacking conference held annually. For this study DEFCON21 (2013) dataset is used. This dataset contains network packet captured from the events like lock picking, scavenger hunts and capture the flag. This study has used the captured flag dataset for the experiment. The total size of the dataset is 100.8 GB.

5.3 Data Collection for RTADBDT Framework

The assessment of the developed framework and algorithms have been validated by comparing and contrasting the output acquired from the implemented experiment results. As discussed in Chapter 1, experiment data were collected to analyse the following three parameters: Accuracy, Memory consumption, and Execution time. At the same time, the data collected can also be used for analysing other parameters as well. Several cycles of real-time processed data are collected for evaluating these three parameters. Likewise, some of the evaluation techniques are evaluated with thirty different cycles of processed data with three different datasets. To analyse the accuracy for three different datasets this study has used various evaluation techniques (see section 5.4). Memory consumption of the proposed algorithms have been calculated by utilizing the prime memory profile package algorithm and lastly, execution time has been calculated by the mathematical model. Results were compared and analysed with two different existing algorithms to show the efficiency of the proposed framework.Figure 5.2 highlight the process flow for the proposed framework data collection and evaluation method to be used for analysis.

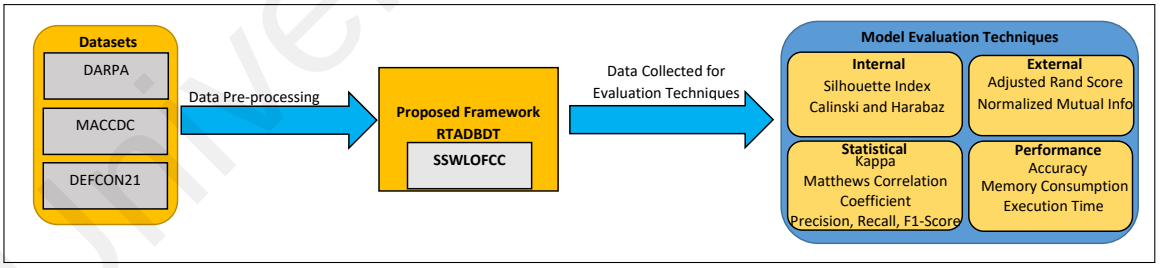

Figure 5.2: Illustrate the data collection process flow for proposed framework

5.4 Performance Evaluation Methods

This section describes various evaluation metrics used for analysing the performance of the proposed framework and algorithms.

5.4.1 Accuracy

This subsection represents the internal and external techniques to evaluate the quality of the clusters formed. The most common evaluation techniques to measure cluster quality is the Silhouette Index and Normalized Mutual Information. Besides, this evaluation has added a few more cluster techniques to examine the cluster quality. Techniques include adjusted rand score and Calinski and Harabaz(see chapter 2 and chapter 4 for details). These techniques are employed to evaluate the internal and external cluster quality up to what extent it is improved as against the other existing methods. Meanwhile, the accuracy of the anomaly detection is evaluated by statistical methods such as confusion matrix, Precision, Recall, Kappa, and F1 score.

5.4.1.1 Silhouette index

Silhouette index is a widely used clustering evaluation technique that measures the cluster quality score. It is a composite index which measures data points in intra and inter clusters. Higher silhouette value denotes an enhanced aspect of a clustering outcome.

The Silhouette index of the i^{th} data record in the cluster C_j=$\{x_{j1},x_{j2},..,x_{jn}\}$, j=$1,...,N_c$

$$SI(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{5.1}$$

where a(i) is the average distance between the i^{th} data record and all the records included in C_j, and b(i) is the minimum average distance between the i^{th} data record and all the records that are located in other cluster $C_{k,k=1,...}N_c$, k not equal to j (J. Zhou et al., 2010).

5.4.1.2 Calinski and Harabaz

Calinski and Harabaz index is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. The greatest index score denotes the model with dense and well separated clusters. These scores are fast to compute. Calinski and Harabaz are

defined as follow (Caliński & Harabasz, 1974). "For k clusters, the Calinski-Harabaz score s is given as the ratio of the between-clusters dispersion mean and the within-cluster dispersion:

$$S(k) = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{N-k}{k-1} \tag{5.2}$$

Where B_k is the between group dispersion matrix and W_k is the within-cluster dispersion matrix defined by:

$$W_k = \sum_{q=1}^{k} \sum_{x \epsilon C_q} (x - c_q)(x - c_q)^T \tag{5.3}$$

$$B_k = \sum_{q} n_q (c_q - c)(c_q - c)^T \tag{5.4}$$

With N be the number of points in the data, C_q be the set of points in cluster q, c_q be the center of cluster q, c be the center of E, n_q be the number of points in cluster q.

5.4.1.3 Adjusted rand score

Adjusted Rand Index (ARI) is an enhancement of the Rand Index. It encourages the index of choice for measuring arrangement between two partitions in clustering analysis with different numbers of clusters. The ARI is also known as the adjusted rand score, which is equal to the index minus the expected index divided by the max index minus expected index (Bharill et al., 2016) (Amini, Wah, & Saboohi, 2014).

$$ARI = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - [\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}] - [\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}]/\binom{n}{2}} \tag{5.5}$$

Where n_{ij} denotes the diagonal (i.e., when i=j), n_i is the row sums and n_j is the column sums.

5.4.1.4 Normalized mutual info score

The normalized mutual information (NMI) is a familiar theoretic information that measures the similarity between two clustering formed (Bharill et al., 2016). Furthermore, it is used to estimate the cluster output developed by an algorithm. NMI is defined as follows:

$$NMI = \frac{\sum_{c=1}^{k}\sum_{p=1}^{m} n_c^p log(\frac{n \cdot n_c^p}{n_c \cdot n_p})}{\sqrt{(\sum_{c=1}^{k} n_c log(\frac{n_c}{n}))(\sum_{p=1}^{m} n_p log(\frac{n_p}{n}))}} \tag{5.6}$$

Where, n is the sum number of data point, n_c and n_p are the numbers of data points in the c^{th} cluster and the p^{th} class and n_c^p is the number of common data points in class p and cluster c.

5.4.1.5 Confusion matrix

A confusion matrix is a list that is used to explain the overall efficiency of an algorithm on a set of test data, for which the true values are found. The values are extracted as true positive (TP), false positive (FP), true negative (TN), and false negative (FN) from the results of an algorithm.

Table 5.1: Confusion matrix explanation table

	Actual positive	**Actual negative**
Predicted positive	True Positive (TP)	False Positive (FP)
Predicted negative	False Negative (FN)	True Negative (TN)

Accuracy (AC) measures precision across all labels

$$AC = \frac{(TP + TN)}{TP + FP + TN + FN} \tag{5.7}$$

5.4.1.6 Precision

Precision is a measure to identify the correctness of the algorithms proposed in this study. It represents the overall percentage of the real anomalies, and helps to identify the number of true positives from the predicted positive values(Dahiya & Srivastava, 2018).

$$precision = \frac{TP}{TP + FP} \tag{5.8}$$

5.4.1.7 Recall

Recall is a measure that indicates the correctly identified positive tuples, and indicates the number of actual positive data that are predicted positive.

$$recall = \frac{TP}{TP + FN} \tag{5.9}$$

5.4.1.8 F1-Score

F1 score is a measure which is the harmonic mean of precision and recall. It helps to define the accuracy of a clustering algorithm (Marir, Wang, Feng, Li, & Jia, 2018).

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.10}$$

5.4.1.9 Matthews correlation coefficient

Matthews correlation coefficient (MCC) is a technique used in machine learning to measure the quality of binary classifications. MCC is a correlation coefficient between the observed and predicted binary classifications. The values of the MCC range from -1 to +1, where +1 indicates a perfect prediction, 0 indicates no better than random prediction, and -1 indicates total disagreement between prediction and observation. Since, there is not a

perfect way to describe the confusion matrix, MCC is considered to be one of the best measures. MCC can be calculated from the confusion matrix using the below formula:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.11}$$

In the above equation, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN the number of false negatives. MCC can also be calculated using the positive predicted value, true positive rate, true negative rate, negative predicted value, false discovery rate, false negative rate, false positive rate, and false omission rate using the below formula.

$$MCC = \sqrt{PPV \times TPR \times TNR \times NPV} - \sqrt{FDR \times FNR \times FPR \times FOR} \tag{5.12}$$

The below formula is the original formula that Matthew had given: N=TN+TP+FN+FP

$$Accuracy, S = \frac{TP + FN}{N} \tag{5.13}$$

$$Predictedpositive, P = \frac{TP + FP}{N} \tag{5.14}$$

$$MCC = \frac{TP \mid N - S \times P}{\sqrt{PS(1 - S)(1 - P)}} \tag{5.15}$$

These above discussed five metrics: Precision, Recall, Kappa, Matthews coefficient and F1-score were applied for spark machine learning application to evaluate the performance of the proposed algorithms in this research (Hafsa & Jemili, 2019).

These techniques are used to show the quality of the proposed cluster algorithms and they are compared with existing methods. Figure 5.3 shows the evaluation techniques and

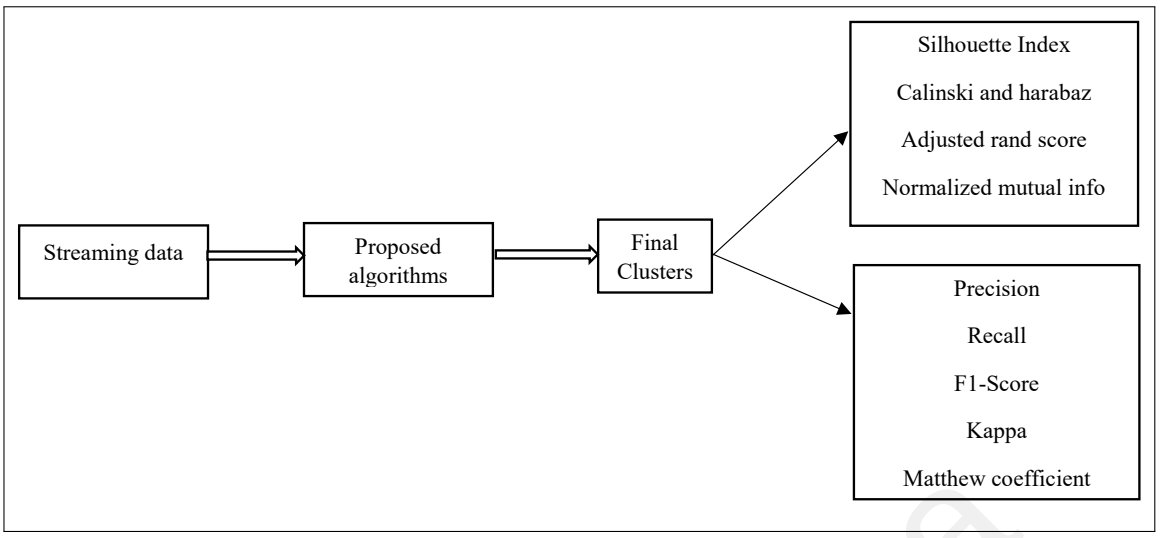

Figure 5.3: Evaluation techniques and its steps for proposed algorithm

its steps used for proposed algorithm.

5.4.1.10 Consumption of memory

To estimate the memory consumption for entire operation, the memory profiler algorithms were utilized to record information about memory usage. The memory consumption can be described as the amount of memory (RAM) consumed for the execution of a specific task. This process has been executed over 4 cycles to derive an approximate memory consumption amount.

5.4.1.11 Execution time

The execution time is described as the time taken in seconds to complete the execution of any given task. The modified memory profiler package has been again used to calculate the execution time in seconds.

The spark architecture has diverse parameters to be set for optimizing the performance. As an initial process, it is essential to recognize the suitable parameters which have an influence on the execution time of process. The study has focused on the parameters which have large changing data in real-time and incorporated the machine learning libraries. The

analysis has calculated the execution time of entire framework using the derived formula discussed in chapter 4. The Spark application is performed by continuous numbers of spark jobs. Then, aggregating all the jobs from the beginning to end will calculate the entire execution time for the framework. The Spark job execution is known as a graph and each node in the graph is expressed as a stage. On each occasion, the operation needs data exchange and then a new stage is generated. Spark job execution time is calculated by totalling the estimated execution time of all the stages.

5.5 Data Collected for Analyzing the Anomaly Detection Accuracy

The accuracy of the developed framework and algorithms integrated in this research have been validated by comparing the outcome of the experiments with various existing datasets. Several evaluation techniques were used to evaluate the data from the framework for analysis. In this section, collected data from the experiment were used to analyse and compare the accuracy of the proposed framework as against the other existing algorithms based on three different datasets i.e. DARPA, MACCDC and DEFCON21.

5.5.1 Silhouette Index

The silhouette Index data is collected using mean intra-cluster distance for three different dataset on proposed algorithms; where 1 indicates the best value and -1 represents the worst value; whereas any index tagged near 0 indicates overlapping clusters, and negative values indicate that a case has been assigned to the inappropriate cluster. Table 5.2 presents the data collected from DARPA, MACCDC and DEFCON21 dataset that have been used in this study. The experiment has yielded the values closer to 1.0 in all three datasets. The analytical results indicate that the proposed framework cluster are in best fit for the processed data. Furthermore higher scores in data indicate that the clusters are compressed and adequately detached.

Table 5.2: Analysis of the internal cluster quality for three different dataset.

Data Traces	Darpa Dataset	MACCDC Dataset	DEFCON 21 Dataset
1	0.813	0.733	0.817
2	0.909	0.884	0.906
3	0.966	0.929	0.929
4	0.994	0.954	0.951
5	1.000	0.967	0.971
6	nan	0.977	0.987
7	nan	0.981	0.992
8	nan	0.982	0.997
9	nan	0.984	0.998
10	nan	0.986	0.999
11	nan	0.990	0.999
12	nan	0.992	1.000
13	nan	0.994	1.000
14	nan	0.994	0.999
Mean	0.936	0.894	0.915
Std. Deviation	0.078	0.095	0.060
T-Value	26.959	21.018	34.132
P-Value		<0.0001	

Based on the data traces, this study has calculated the mean values of silhouette index for all three datasets, and acquired: 0.936, 0.894, and 0.915 for the DARPA, MACCDC, and DEFCON21 datasets respectively. Based on these mean values the standard deviation for each dataset has been calculated, which yielded 0.078, 0.095, and 0.060 for the DARPA, MACCDC, and DEFCON21 datasets respectively. It is noteworthy that the above mentioned standard deviation values are relatively low, which indicates that data points are clustered closely around the mean and hence the reliability is proved.

Furthermore, this analysis has calculated the T-Value and P-Value of the silhouette scoring for each dataset. The T- values for silhouette scoring are: 26.959, 21.018, and 34.132, for the DARPA, MACCDC, and DEFCON21 datasets respectively. The P values calculated for the DARPA, MACCDC, and DEFCON21 datasets were less than 0.0001 for each of the dataset. Hence the P-value attained has been proved to be statistically significant for all the 3 datasets.

5.5.2 Adjusted Rand Index method

To derive the accuracy of the solutions, the proposed model has been executed over a cycle of 30 times and the following results were derived for evaluation of Adjusted Rand Index method. This method is used to calculate the quality of the clusters for the proposed and existing algorithms. The proposed solution has been compared with six different existing algorithms. Subsequently, these algorithms were evaluated with three different datasets.

Table 5.3: Comparison of accuracy between existing and proposed algorithms on DARPA dataset.

Algorithms /Data Trace	K Means	Isolation Forest	Spectral Clustering	HDBSCAN	Agglomerative Clustering	Local Outlier Factor	SSWLOFCC (Proposed Algorithm)
1	84.915	91.405	52.101	12.050	85.206	94.501	98.510
2	84.952	91.415	52.145	12.011	85.201	94.511	98.513
3	84.931	91.482	52.121	12.023	85.200	94.523	98.518
4	84.992	91.443	52.183	12.015	85.201	94.509	98.519
5	84.933	91.426	52.107	12.062	85.262	94.502	98.539
6	84.957	91.457	52.194	12.015	85.221	94.522	98.502
7	84.922	91.431	52.111	12.091	85.251	94.531	98.511
8	84.983	91.435	52.144	12.023	85.244	94.551	98.559
9	84.974	91.482	52.176	12.042	85.271	94.524	98.599
10	84.912	91.471	52.132	12.019	85.290	94.569	98.533
11	84.905	91.464	52.191	12.075	85.209	94.571	98.529
12	84.911	91.451	52.164	12.035	85.210	94.517	98.539
13	84.963	91.427	52.133	12.023	85.216	94.505	98.579
14	84.982	91.434	52.155	12.014	85.270	94.532	98.517
15	84.929	91.416	52.162	12.010	85.230	94.511	98.519
16	84.934	91.406	52.132	12.027	85.234	94.504	98.534
17	84.977	91.401	52.177	12.075	85.251	94.515	98.548
18	84.941	91.400	52.168	12.004	85.243	94.531	98.555
19	84.932	91.417	52.113	12.009	85.222	94.522	98.500
20	84.905	91.435	52.151	12.018	85.201	94.529	98.505
21	84.986	91.417	52.175	12.012	85.211	94.514	98.529
22	84.962	91.481	52.182	12.074	85.224	94.506	98.565
23	84.955	91.492	52.191	12.035	85.274	94.512	98.574
24	84.961	91.437	52.134	12.018	85.262	94.514	98.532
25	84.943	91.415	52.124	12.029	85.234	94.513	98.593
26	84.981	91.410	52.161	12.034	85.281	94.509	98.524
27	84.933	91.427	52.182	12.055	85.202	94.507	98.556
28	84.987	91.401	52.105	12.002	85.203	94.511	98.585
29	84.914	91.405	52.114	12.004	85.207	94.514	98.525
30	84.905	91.403	52.109	12.009	85.208	94.508	98.544
Mean	84.946	91.433	52.148	12.030	85.231	94.520	98.539
Std. Deviation	0.028	0.028	0.030	0.024	0.028	0.017	0.028
T-Value	16431.758	17997.835	9528.768	2695.097	16649.742	29596.569	19604.541
P-Value	0.0001						

The table 5.3 presents the accuracy of the proposed SSWLOFCC algorithm compared with six different existing algorithms namely k Means, Isolation forest, Spectral Clustering HDBSCAN, Agglomerative clustering, Local Outlier Factor on DARPA dataset using adjusted rand index evaluation method. The first column in the table represents the 30 cycles of data traces for analysis and the first row represents the algorithms. The average percentage of these six algorithms yielded the accuracy of 84.95%, 91.43%, 52.15%, 12.03%, 85.23%, and 94.52%; whereas, the proposed SSWLOFCC algorithm has yielded a much higher accuracy of 98.54% for DARPA dataset.

Table 5.4: Comparison of accuracy between existing and proposed algorithms on MACCDC Dataset.

Algorithms /Data Trace	K Means	Isolation Forest	Spectral Clustering	HDBSCAN	Agglomerative Clustering	Local Outlier Factor	SSWLOFCC (Proposed Algorithm)
1	85.001	87.610	73.311	14.010	83.510	91.614	96.311
2	85.021	87.605	73.322	14.041	83.530	91.617	96.344
3	85.003	87.614	73.301	14.063	83.570	91.601	96.352
4	85.020	87.651	73.372	14.095	83.540	91.609	96.383
5	85.015	87.637	73.392	14.014	83.514	91.617	96.398
6	85.063	87.625	73.306	14.001	83.534	91.621	96.341
7	85.024	87.661	73.396	14.002	83.522	91.610	96.323
8	85.091	87.675	73.321	14.006	83.541	91.160	96.305
9	85.052	87.691	73.343	14.071	83.565	91.102	96.372
10	85.031	87.621	73.334	14.012	83.517	91.617	96.310
11	85.076	87.624	73.303	14.060	83.531	91.604	96.331
12	85.078	87.616	73.348	14.054	83.541	91.623	96.311
13	85.051	87.617	73.315	14.021	83.536	91.671	96.364
14	85.030	87.623	73.317	14.038	83.562	91.613	96.341
15	85.055	87.625	73.363	14.067	83.503	91.674	96.388
16	85.031	87.617	73.391	14.053	83.504	91.626	96.395
17	85.010	87.661	73.335	14.041	83.520	91.634	96.314
18	85.044	87.637	73.372	14.061	83.534	91.612	96.379
19	85.039	87.619	73.381	14.012	83.571	91.607	96.307
20	85.093	87.629	73.354	14.031	83.532	91.645	96.379
21	85.031	87.627	73.333	14.051	83.515	91.661	96.334
22	85.041	87.631	73.324	14.057	83.503	91.613	96.356
23	85.055	87.630	73.375	14.064	83.509	91.609	96.382
24	85.058	87.671	73.384	14.037	83.541	91.610	96.303
25	85.091	87.641	73.328	14.001	83.530	91.601	96.398
26	85.042	87.690	73.391	14.005	83.528	91.600	96.324
27	85.077	87.605	73.352	14.006	83.531	91.603	96.380
28	85.072	87.604	73.376	14.003	83.504	91.616	96.346
29	85.014	87.607	73.355	14.009	83.506	91.620	96.372
30	85.047	87.611	73.303	14.001	83.517	91.631	96.390
Mean	85.045	87.633	73.347	14.033	83.529	91.588	96.351
Std. Deviation	0.027	0.025	0.031	0.027	0.020	0.126	0.032
T-Value	17553.018	19387.148	13000.290	2829.524	23248.753	3983.201	16463.881
P-Value	0.0001						

Likewise, the table 5.4 presents the accuracy of the proposed SSWLOFCC algorithm as against six different existing algorithms namely: k Means, Isolation forest, Spectral Clustering HDBSCAN, Agglomerative clustering, Local Outlier Factor on MACCDC dataset, using adjusted rand index evaluation method. The first column in the table represents the 30 cycles of data traces for analysis and the first row represents the algorithms. The average accuracy percentage of these six algorithms are85.05%, 87.63%, 73.35%, 14.03%, 83.53%, and 91.59% whereas, the proposed SSWLOFCC algorithm has yielded a much higher accuracy of 96.35% for MACCDC Dataset

Meanwhile, the table 5.5 presents the accuracy of the proposed SSWLOFCC algorithm compared with six different existing algorithms namely: k Means, Isolation forest, Spectral Clustering HDBSCAN, Agglomerative clustering, Local Outlier Factor on DEFCON21 dataset, using adjusted rand index evaluation method. The first column in the table represents the 30 cycles of data traces for analysis, and the first row represents the algorithms. The average percentage of accuracy of these six algorithms have been recorded as 65.44%, 91.23%, 88.74%, 85.54%, 47.22%, and 87.42%; whereas, the proposed

Table 5.5: Comparison of accuracy between existing and proposed algorithms on DEFCON21 Dataset.

Algorithms /Data Trace	K Means	Isolation Forest	Spectral Clustering	HDBSCAN	Agglomerative Clustering	Local Outlier Factor	SSWLOFCC (Proposed Algorithm)
1	65.400	91.260	88.721	85.520	47.215	87.401	94.632
2	65.410	91.251	88.753	85.511	47.220	87.408	94.671
3	65.432	91.237	88.709	85.510	47.201	87.414	94.664
4	65.491	91.255	88.764	85.530	47.203	87.481	94.691
5	65.473	91.291	88.729	85.534	47.209	87.413	94.624
6	65.469	91.220	88.793	85.581	47.224	87.471	94.614
7	65.432	91.205	88.702	85.515	47.232	87.423	94.645
8	65.445	91.209	88.717	85.511	47.210	87.424	94.632
9	65.491	91.216	88.723	85.584	47.212	87.416	94.671
10	65.433	91.238	88.767	85.532	47.216	87.471	94.692
11	65.421	91.239	88.711	85.567	47.219	87.444	94.681
12	65.411	91.248	88.784	85.575	47.233	87.421	94.665
13	65.401	91.221	88.739	85.581	47.234	87.416	94.639
14	65.403	91.215	88.741	85.592	47.214	87.405	94.697
15	65.415	91.202	88.753	85.514	47.217	87.421	94.662
16	65.417	91.207	88.772	85.536	47.201	87.416	94.631
17	65.435	91.232	88.702	85.587	47.206	87.432	94.644
18	65.482	91.267	88.715	85.551	47.209	87.441	94.659
19	65.491	91.291	88.783	85.536	47.260	87.404	94.604
20	65.461	91.203	88.729	85.533	47.212	87.406	94.687
21	65.440	91.207	88.714	85.537	47.204	87.432	94.634
22	65.425	91.251	88.777	85.581	47.204	87.415	94.699
23	65.414	91.222	88.782	85.561	47.213	87.424	94.662
24	65.412	91.216	88.765	85.564	47.204	87.422	94.651
25	65.401	91.232	88.792	85.544	47.209	87.436	94.692
26	65.409	91.274	88.704	85.514	47.261	87.401	94.684
27	65.429	91.291	88.715	85.553	47.213	87.405	94.655
28	65.436	91.205	88.762	85.571	47.217	87.406	94.643
29	65.484	91.209	88.788	85.501	47.218	87.451	94.632
30	65.403	91.207	88.707	85.505	47.213	87.412	94.610
Mean	65.436	91.234	88.744	85.544	47.217	87.424	94.656
Std. Deviation	0.030	0.028	0.031	0.028	0.015	0.021	0.027
T-Value	11763.158	17741.356	15502.570	16441.000	17538.833	22532.172	19004.932
P-Value	0.0001						

SSWLOFCC algorithm has yielded a much higher accuracy of 94.66% for DEFCON21 dataset.

Based on the data traces, this study has calculated the mean values for the accuracy for all three datasets and 7 algorithms. This study has acquired a mean value of 84.946 for K Means, 91.433 for Isolation Forest, 52.148 for Spectral Clustering, 12.030 for HDBSCAN, 85.231 for Agglomerative Clustering, 94.520 for Local Outlier Factor, and 98.539 for the proposed SSWLOFCC algorithm for the DARPA dataset. For the MACCDC, the mean values are 85.045 for K Means, 87.633 for Isolation Forest, 73.347 for Spectral Clustering, 14.033 for HDBSCAN, 83.529 for Agglomerative Clustering, 91.588 for Local Outlier Factor, and 96.351 for the proposed SSWLOFCC algorithm. For the DEFCON21 dataset, the mean values are 65.436 for K Means, 91.234 for Isolation Forest, 88.744 for Spectral Clustering, 85.544 for HDBSCAN, 47.217 for Agglomerative Clustering, 87.424 for Local Outlier Factor, and 94.656 for the proposed SSWLOFCC algorithm.

Based on these mean values the standard deviation for each dataset and each algorithm has been calculated, which yielded the value of 0.028 for K Means, 0.028 for Isolation

Forest, 0.030 for Spectral Clustering, 0.024 for HDBSCAN, 0.028 for Agglomerative Clustering, 0.017 for Local Outlier Factor, and 0.028 for the proposed SSWLOFCC algorithm for the DARPA dataset. For the MACCDC the standard deviation values are 0.027 for K Means, 0.025 for Isolation Forest, 0.031 for Spectral Clustering, 0.027 for HDBSCAN, 0.020 for Agglomerative Clustering, 0.126 for Local Outlier Factor, and 0.032 for the proposed SSWLOFCC algorithm. For the DEFCON21 dataset, the standard deviation values have been recorded as: 0.030 for K Means, 0.028 for Isolation Forest, 0.031 for Spectral Clustering, 0.028 for HDBSCAN, 0.015 for Agglomerative Clustering, 0.021 for Local Outlier Factor, and 0.027 for the proposed SSWLOFCC algorithm. It is noteworthy that the above mentioned the standard deviation values are relatively low, which indicates that the data points are clustered closely around the mean and hence their reliability is proved.

Furthermore, the T-Value and P-Value for the accuracy for each dataset has been calculated. The T values for DARPA dataset are: (i) 16431.758 for K Means, (ii) 17997.835 for Isolation Forest, (iii) 9528.768 for Spectral Clustering, (iv) 2695.097 for HDBSCAN, (v) 16649.742 for Agglomerative Clustering, (vi) 29596.569 for Local Outlier Factor, and (vii)19604.541 for the proposed SSWLOFCC algorithm. Furthermore, for the MACCDC the T values are: (i) 17553.018 for K Means, (ii) 19387.148 for Isolation Forest, (iii) 13000.290 for Spectral Clustering, (iv) 2829.524 for HDBSCAN, (v) 23248.753 for Agglomerative Clustering, (vi) 3983.201 for Local Outlier Factor, and (vii) 16463.881 for the proposed SSWLOFCC algorithm. Additionally, for the DEFCON21 dataset the T-values are: (i) 11763.158 for K Means, (ii) 17741.356 for Isolation Forest, (iii) 15502.570 for Spectral Clustering, (iv) 16441.000 for HDBSCAN, (v) 17538.833 for Agglomerative Clustering, (vi) 22532.172 for Local Outlier Factor, and (vii) 19004.932 for the proposed SSWLOFCC algorithm. The P values calculated for the DARPA, MACCDC,

and DEFCON21 datasets were 0.0001 apiece. Hence the P-value attained has been proved to be statistically significant for all the 3 datasets.

5.5.3 Normalized Mutual Info (NMI)

The Normalized Mutual Info score data has been collected by normalizing the mutual information score, where by the results are scaled between 0 to1; in which 0 represents no mutual information and 1 indicates perfect correlation. The generalized mean of the entropies in each clustering represent the normalizing value. The results of the application of normalized score method to the three different datasets has been presented in table 5.6.

Table 5.6: Proposed algorithm normalized mutual info score is compared for three different datasets

Data Traces	DARPA Dataset	MACCDC Dataset	DEFCON 21 Dataset)
1	0.983	0.982	0.963
2	0.981	0.982	0.963
3	0.984	0.982	0.965
4	0.981	0.982	0.961
5	0.981	0.983	0.960
6	0.980	0.984	0.962
7	0.982	0.981	0.963
8	0.983	0.984	0.964
9	0.980	0.984	0.962
10	0.982	0.982	0.965
11	0.983	0.983	0.962
12	0.981	0.984	0.964
13	0.981	0.981	0.962
14	0.984	0.984	0.964
15	0.984	0.984	0.964
16	0.980	0.981	0.962
17	0.982	0.984	0.961
18	0.982	0.981	0.962
19	0.981	0.980	0.961
20	0.982	0.981	0.960
21	0.981	0.982	0.961
22	0.981	0.984	0.963
23	0.984	0.982	0.964
24	0.983	0.983	0.963
25	0.982	0.981	0.962
26	0.984	0.984	0.960
27	0.982	0.984	0.963
28	0.981	0.983	0.963
29	0.981	0.981	0.960
30	0.983	0.981	0.961
Mean	0.982	0.983	0.962
Std. Deviation	0.001	0.001	0.001
T-Value	4505.054	3965.144	3785.204
P-Value	0.0001		

Based on the data traces the mean of normalized mutual info for all the three datasets have been calculated and acquires mean values of 0.982, 0.983, and 0.962 for the DARPA, MACCDC, and DEFCON21 datasets respectively.

Based on these mean values the standard deviation for each dataset has been calculated, which yielded 0.001, for the DARPA, MACCDC, and DEFCON21 datasets. It is worth

mentioning that the above mentioned standard deviation values are relatively low, which indicates that the data points are clustered closely around the mean and hence the reliability is proved.

Furthermore, the T-Value and P-Value of the normalized mutual info for each dataset have been calculated. The T-values for normalized mutual info is 4505.054, 3965.144, and 3785.204 for the DARPA, MACCDC, and DEFCON21 datasets respectively. The P-values of DARPA dataset, MACCDC dataset, and DEFCON21 dataset is 0.0001 apiece. Hence the P-value attained has been proved to be statistically significant for all the 3 datasets.

This analysis has been able to receive a normalized mutual info score of 0.98 for the DARPA and MACCDC. As well as, 0.96 for DEFCON21 dataset. In all the three cases the proposed SSWLOFCC algorithm has achieved higher scores, hence proving better correlation among the clusters.

The above precisely discussed internal and external cluster evaluation metrics had revealed that the proposed SSWLOFCC algorithm has yielded significantly higher values in all the selected performance metrics as against the other existing methods.

5.5.3.1 Data collected for cluster validation

This section presents the processed data of proposed SSWLOFCC algorithm. These data were collected from three different datasets with sample of 40 data traces (see tables 5.7, 5.8, and 5.9). Our results have been primarily evaluated using the values of TP, TN, FP and FN, derived from the confusion matrix. The evaluation is done with regards to accuracy, Precision, Recall, Kappa, and F1 score. From the TP, TN, FP and FN, values we are able to classify the results into either correctly or incorrectly identified anomalies. Additionally, the accuracy of our model provide confidence in evaluating the anomalies. Further, the precision results indicate the relevant instances from the retrieved information in which higher precision signify the better detection results. Similarly, the recall values

indicate the relevant instances that have been successfully identified. Kappa in this context is used to show the inter-rater agreement of the items. Generally, a higher Kappa value indicates a more accurate output. Finally, the F1-score which uses two values i.e. precision and recall has been used to test the accuracy. In this case, a higher F1-score indicates a more accuracy in test results. Through our thorough analysis we found significant results from these evaluation metrics which proven the appropriateness of our data.

Table 5.7: Cluster data processed by SSWLOFCC algorithm for DARPA dataset.

Ts	id.resp_p	method	resp_mime_types	request_body_len	x	y	cluster
1/8/2009 3:03	80	GET	-	0	1.509429937	-0.427010812	1
1/8/2009 3:22	80	GET	text/html	0	0.399891266	1.154606463	2
1/8/2009 3:22	80	GET	text/html	0	0.273855152	1.168686934	2
1/8/2009 3:23	80	GET	-	0	1.572861436	-0.464440242	1
1/8/2009 3:23	80	GET	-	0	1.512732874	-0.440643502	1
1/8/2009 3:23	80	GET	text/html	0	0.375632276	1.174564378	2
1/8/2009 3:24	80	GET	-	0	1.638034956	-0.458033827	1
1/8/2009 3:24	8180	GET	text/html	0	1.524139639	0.628923787	3
1/8/2009 3:25	80	GET	-	0	1.541874121	-0.453769926	1
1/8/2009 3:25	8180	GET	text/html	0	1.477691136	0.676562771	3
1/8/2009 3:25	80	GET	text/html	0	0.313959177	1.086894554	2
1/8/2009 3:25	8180	GET	text/html	0	1.549467338	0.649213108	3
1/8/2009 3:25	80	GET	-	0	1.648513999	-0.440167963	1
1/8/2009 3:25	8180	GET	text/html	0	1.501004756	0.638779453	3
1/8/2009 3:33	80	GET	-	0	1.520441838	-0.419010587	1
1/8/2009 3:33	80	GET	text/html	0	0.446515754	1.150169621	2
1/8/2009 3:34	80	GET	text/html	0	0.325050233	1.110407112	2
1/8/2009 3:51	80	GET	-	0	1.567311991	-0.453027794	1
1/8/2009 3:51	80	GET	text/html	0	0.353996669	1.166865573	2
1/8/2009 3:51	80	GET	-	0	1.578157256	-0.429975444	1
1/8/2009 3:51	80	GET	text/html	0	0.385305313	1.118848823	2
1/8/2009 3:51	80	GET	-	0	1.533976856	-0.478609953	1
1/8/2009 3:51	80	GET	text/html	0	0.390181859	1.116525858	2
1/8/2009 4:02	80	GET	-	0	1.646483187	-0.42224983	1
1/8/2009 4:02	80	GET	text/html	0	0.299834915	1.13767814	2
1/8/2009 4:02	80	GET	text/html	0	0.232482258	1.063576603	2
1/8/2009 4:21	80	GET	text/plain	0	-0.351223023	-0.090599988	0
1/8/2009 4:21	80	GET	text/html	0	0.366668915	1.063468191	2
1/8/2009 4:24	80	GET	text/plain	0	-0.375815852	-0.060236839	0
1/8/2009 4:24	80	GET	text/html	0	0.382471826	1.163087435	2
1/8/2009 4:24	80	GET	text/html	0	0.475346013	1.111485465	2
1/8/2009 4:24	80	GET	text/plain	0	-0.431821404	-0.116520304	0
1/8/2009 4:24	80	GET	text/plain	0	-0.319061675	-0.018642022	0
1/8/2009 4:24	80	GET	text/plain	0	-0.318632487	-0.064704314	0
1/8/2009 4:25	80	GET	text/plain	0	-0.359746211	-0.151037488	0
1/8/2009 4:25	80	GET	text/html	0	0.331972547	1.106529889	2
1/8/2009 4:25	80	GET	text/html	0	0.314120454	1.129552671	2
1/8/2009 4:25	80	GET	text/html	0	0.313096083	1.146146342	2
1/8/2009 4:25	80	GET	text/html	0	0.385836604	1.145351102	2

Table 5.8: Cluster data processed by SSWLOFCC algorithm for MACCDC dataset.

Ts	id.resp_p	method	resp_mime_types	request_body_len	x	y	cluster
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	-	0	0.10967	1.065494	0
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	-	0	0.10967	1.065494	0
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	-	16	0.10967	1.065494	0
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	-	0	0.10967	1.065494	0
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/plain	0	-0.63501	-0.11065	1
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3
18/2/2015 8:51	80	GET	text/html	0	0.771351	-0.16473	3

Table 5.9: Cluster data processed by SSWLOFCC algorithm for DEFCON21 dataset.

Ts	id.resp_p	method	resp_mime_types	request_body_len	x	y	cluster
51:08.0	80	GET	text/plain	0	-0.67384	-0.12702	3
51:08.1	80	GET	text/plain	0	-0.67384	-0.12702	3
51:08.5	80	GET	text/plain	0	-0.67384	-0.12702	3
51:08.6	80	GET	text/plain	0	-0.67384	-0.12702	3
51:08.7	80	GET	text/html	0	0.738839	-0.15515	0
51:08.9	80	GET	text/plain	0	-0.67384	-0.12702	3
51:09.4	80	GET	text/plain	0	-0.67384	-0.12702	3
51:08.2	80	GET	-	0	0.05411	1.074689	2
51:09.6	80	GET	text/html	0	0.738839	-0.15515	0
51:09.7	80	GET	text/html	0	0.738839	-0.15515	0
51:09.8	80	GET	text/plain	0	-0.67384	-0.12702	3
51:04.3	80	GET	-	0	0.05411	1.074689	2
51:09.9	80	GET	text/html	0	0.738839	-0.15515	0
51:10.3	80	GET	text/plain	0	-0.67384	-0.12702	3
51:03.9	80	GET	-	16	0.05411	1.074691	2
51:10.6	80	GET	text/plain	0	-0.67384	-0.12702	3
51:10.7	80	GET	text/plain	0	-0.67384	-0.12702	3
51:10.8	80	GET	text/html	0	0.738839	-0.15515	0
51:11.0	80	GET	text/plain	0	-0.67384	-0.12702	3
51:11.0	80	GET	text/html	0	0.738839	-0.15515	0
51:11.3	80	GET	text/html	0	0.738839	-0.15515	0
51:11.5	80	GET	text/html	0	0.738839	-0.15515	0
51:11.8	80	GET	text/plain	0	-0.67384	-0.12702	3
51:12.2	80	GET	text/plain	0	-0.67384	-0.12702	3
51:12.4	80	GET	text/plain	0	-0.67384	-0.12702	3
51:12.6	80	GET	text/plain	0	-0.67384	-0.12702	3
51:07.0	80	GET	-	0	0.05411	1.074689	2
51:12.8	80	GET	text/plain	0	-0.67384	-0.12702	3
51:12.9	80	GET	text/plain	0	-0.67384	-0.12702	3
51:13.2	80	GET	text/html	0	0.738839	-0.15515	0
51:13.4	80	GET	text/plain	0	-0.67384	-0.12702	3
51:13.6	80	GET	text/plain	0	-0.67384	-0.12702	3
51:13.6	80	GET	text/html	0	0.738839	-0.15515	0
51:14.2	80	GET	text/plain	0	-0.67384	-0.12702	3
51:14.6	80	GET	text/plain	0	-0.67384	-0.12702	3
51:14.6	80	GET	text/plain	0	-0.67384	-0.12702	3
51:14.2	80	GET	text/html	0	0.738839	-0.15515	0
51:14.8	80	GET	text/html	0	0.738839	-0.15515	0
51:15.0	80	GET	text/html	0	0.738839	-0.15515	0

The above tables 5.7, 5.8, and 5.9 comprises of a variety of network attacks and normal network data extracted from the DARPA, MACCDC, and DEFCON21 datasets respectively. These data has been extracted after the implementation of the proposed algorithm. The columns *id.resp_p, method, resp_mime_types* and *request_body_len* present the features extracted from the original dataset. The cluster column in the table holds the records of clusters. The columns x and y define where the record will be plotted in the scatter plot (the x and y axsis).

Confusion matrix are generated by the proposed algorithm based on above data. These confusion matrix data has been used for statistical analysis of anomaly detection. Furthermore, these confusion matrix are generated from three different dataset and compared with two different algorithms. Confusion matrix results are presented in the heat map below and the data have been scaled to a maximum of 1.0 value to fit for the heat map. Figure 5.4, Figure 5.5, Figure 5.6 represent the confusion matrix for proposed SSWLOFCC algorithm.

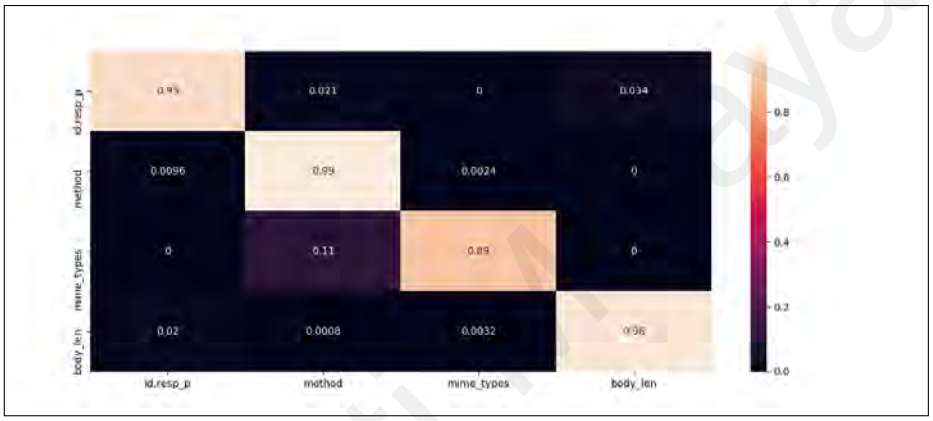

Figure 5.4: Confusion matrix for SSWLOFCC algorithm on DARPA dataset

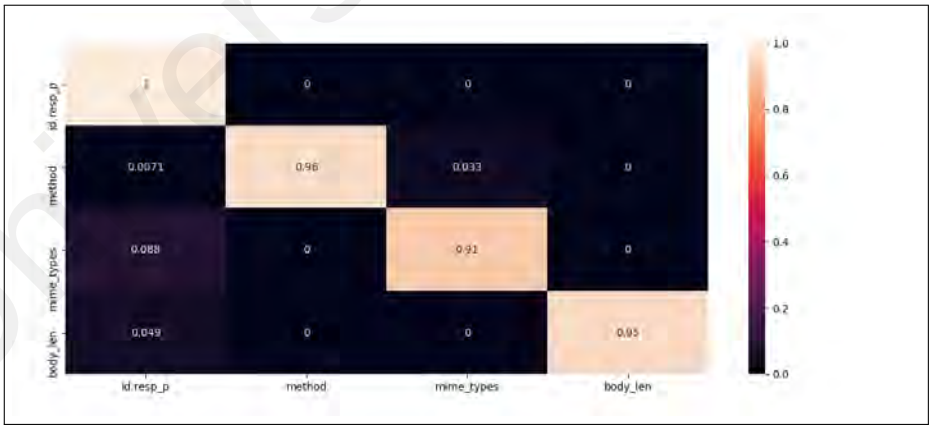

Figure 5.5: Confusion matrix for SSWLOFCC algorithm on MACCDC dataset

Figure 5.6: Confusion matrix for SSWLOFCC algorithm on DEFCON21 dataset

Similarly, in figure 5.7, 5.8, and 5.9 illustrate the confusion matrix for existing local factor outlier algorithm on three different datasets.

Figure 5.7: Confusion matrix for local factor outlier algorithm on DARPA datasets

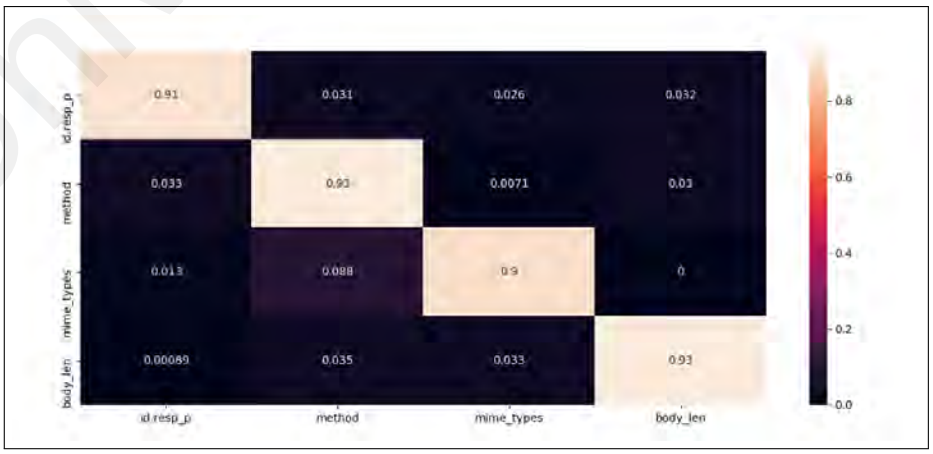

Figure 5.8: Confusion matrix for local factor outlier algorithm on MACCDC datasets

Figure 5.9: Confusion matrix for local factor outlier algorithm on DEFCON21 datasets

Lastly, in figure 5.10, 5.11, 5.12 illustrate the confusion matrix for existing Agglomerative Clustering algorithm on three different dataset.

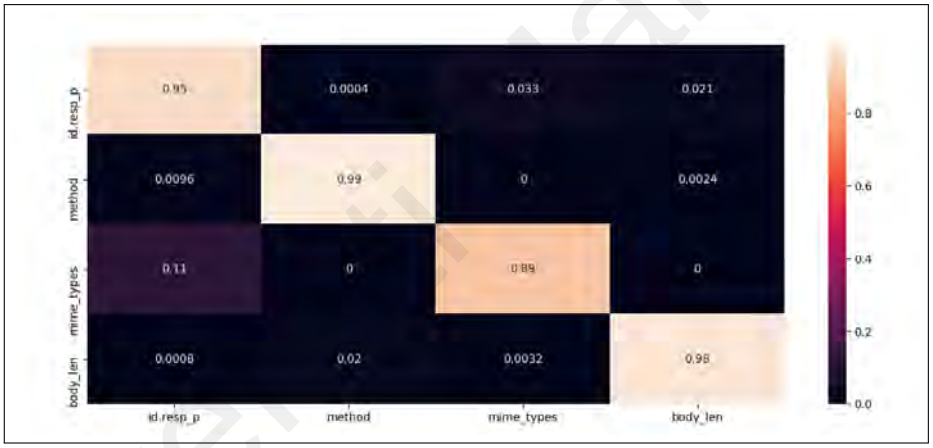

Figure 5.10: Confusion matrix for Agglomerative Clustering algorithm on DARPA dataset

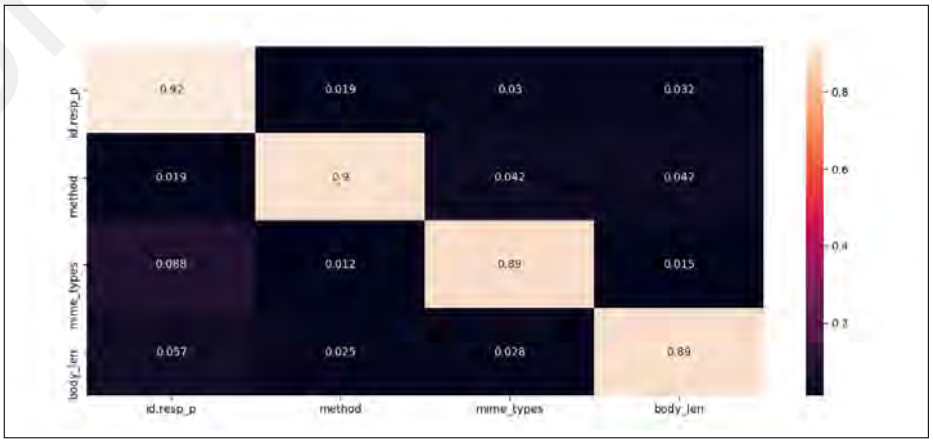

Figure 5.11: Confusion matrix for Agglomerative Clustering algorithm on MAC-CDC dataset

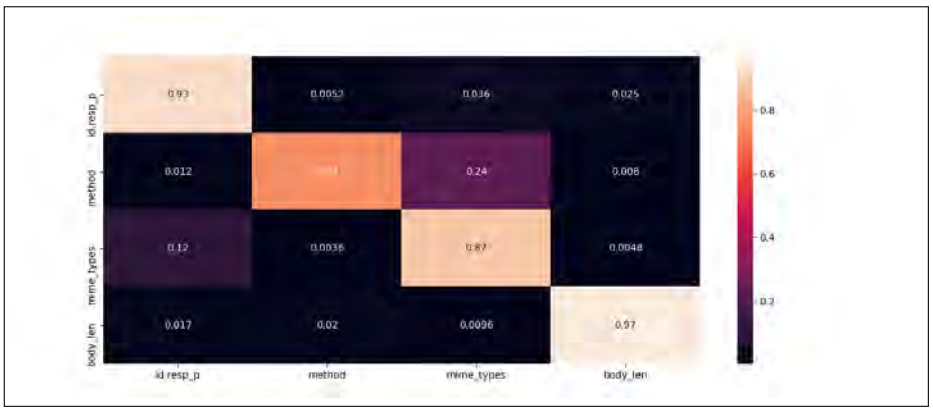

Figure 5.12: Confusion matrix for Agglomerative Clustering algorithm on DEF-CON21 datasets

These confusion matrix values for proposed and two existing algorithms has been compared in the chapter 6 and the true positive, true negative, false positive and false negative have been derived to calculate the accuracy, Precision, Recall, Kappa, and F1-score.

5.6 Data Collected for Process Execution Time

This section presents the execution time data collected by running the proposed algorithms and framework on three different datasets. Further, the proposed SSWLOFCC algorithm has been compared with six different algorithms namely K-means, Isolation Forest, Spectral clustering, HDBSCAN, Agglomerative Clustering, Local outlier factor. In this context, the execution time is described as the time taken in seconds to complete the execution of a task. The execution time has been measured in seconds using the developed execution time algorithm. Figure 5.13 presents the flow diagram of data collection for execution time of proposed framework.

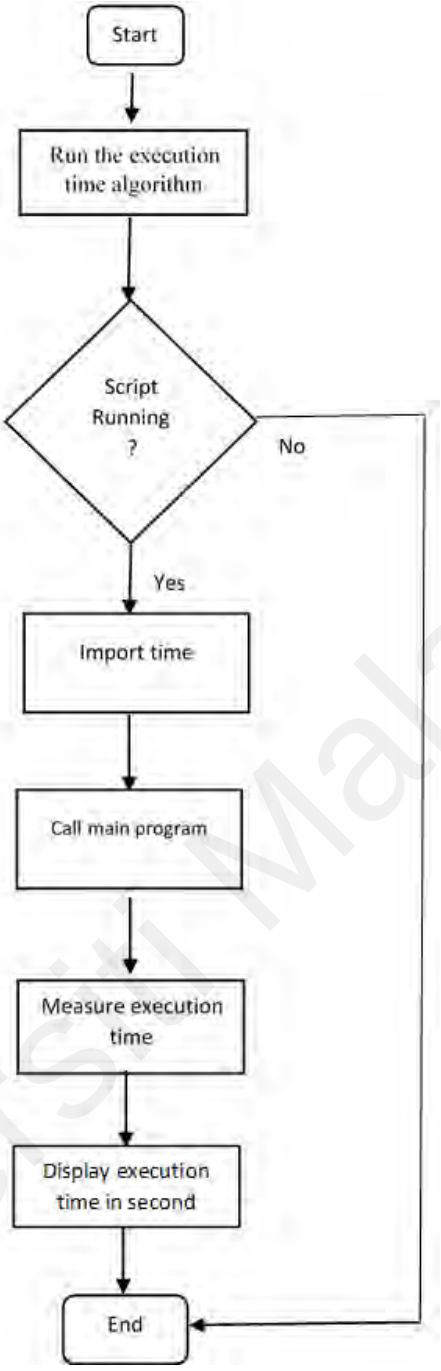

Figure 5.13: Flowchart of collecting execution time on proposed framework

The table 5.10 shows the data derived to compare the execution time of proposed and existing methods with DARPA, MACCDC and DEFCON21 datasets.

Table 5.10: Comparison of the execution time results obtained from proposed solutions with six different algorithms for DARPA, MACCDC, and DEFCON21 datasets.

Datasets / Algorithms	DARPA	MACCDC	DEFCON21	Mean	Std. Deviation	T-Value	P-Value
K Means	28.000	28.000	30.000	28.667	1.155	43.000	
Isolation Forest	18.000	15.000	19.000	17.333	2.082	14.422	
Spectral Clustering	45.000	39.000	38.000	40.667	3.786	18.605	
HDBSCAN	160.000	65.000	95.000	106.667	48.563	3.804	0.0005
Agglomerative Clustering	15.000	14.000	17.500	15.500	1.803	14.892	
Local Outlier Factor	14.000	18.000	17.5.00	16.500	2.179	13.113	
SSWLOFCC (Proposed Algorithm)	**13.000**	**12.000**	**15.000**	**13.333**	**1.528**	**15.119**	

The proposed SSWLOFCC algorithm has consumed lesser execution time in the three datasets as against the six existing algorithms.

From the data traces the mean for the framework execution time of all seven algorithms in the three datasets has been calculated and acquired a mean value of : (i) 28.667 for K Means, (ii) 17.333 for Isolation Forest, (iii) 40.667 for Spectral Clustering, (iv) 106.667 for HDBSCAN, (v) 15.500 for Agglomerative Clustering, (v1) 16.500 for Local Outlier Factor, and (vii) 13.333 for the proposed SSWLOFCC algorithm for the DARPA, MACCDC, and DEFCON21 datasets.

Based on these mean values the standard deviation for each dataset and each algorithm have been calculated, which yielded value of: (i) 1.155 for K Means, (ii) 2.082 for Isolation Forest, (iii) 3.786 for Spectral Clustering, (iv) 48.563 for HDBSCAN, (v) 1.803 for Agglomerative Clustering, (vi) 2.179 for Local Outlier Factor, and (vii) 1.528 for the proposed SSWLOFCC algorithm for the DARPA, MACCDC, and DEFCON21 datasets. It is worth mentioning that the above mentioned standard deviation values are relatively low, which indicates that the data points are clustered closely around the mean and hence the reliability is proved.

Furthermore, the T-Value and P-Value for the framework execution time for each algorithm for all three datasets have been calculated; wherein the following T values have been recorded: 43.000 for K Means, 14.422 for Isolation Forest, 18.605 for Spectral Clustering, 3.804 for HDBSCAN, 14.892 for Agglomerative Clustering, 13.113 for

Local Outlier Factor, and 15.119 for the proposed SSWLOFCC (proposed algorithm). Additionally, the P values calculated for the DARPA dataset, MACCDC dataset, and DEFCON21 dataset was 0.0005 apiece. Hence the P-value attained has been proved to be statistically significant for all the 3 datasets.

5.7 Data Collected for Spark Streaming Execution Time

This section presents the data calculated by the formula developed in this study (see chapter 5) for spark streaming execution time. The spark streaming execution has been performed by combining jobs into batches. Each batch contains many jobs, and contain scheduling delay, processing time and total delay. This analysis has collected execution time for the three different datasets with 10 data traces. The table below shows the spark execution time for proposed SSWLOFCC algorithm in the three datasets.

Table 5.11: Comparison of framework execution time of proposed SSWLOFCC algorithm for DARPA, MACCDC, and DEFCON21 datasets.

Data Trace	DARPA (Seconds)	MACCDC (Seconds)	DEFCON21 (Seconds)
1	5.000	6.000	6.000
2	3.000	2.000	3.000
3	3.000	4.000	1.000
4	1.000	5.000	2.000
5	6.000	4.000	2.000
6	4.000	5.000	4.000
7	2.000	2.000	5.000
8	2.000	1.000	5.000
9	1.000	1.000	2.000
10	3.000	3.000	1.000
Mean	3.000	3.300	3.100
Std. Deviation	1.633	1.767	1.792
T-Value	5.809	5.906	5.471
P-Value		0.0003	

Based on the data traces the spark execution time mean of all three datasets has been calculated, and acquired a mean value of 3.000, 3.300, and 3.100 for the DARPA, MACCDC, and DEFCON21 datasets respectively.

Based on the mean values the standard deviation for each dataset has been calculated, which yielded a value of 1.633, 1.767, 1.792 for the DARPA, MACCDC, and DEFCON21 datasets respectively. The above mentioned standard deviation values have proven to be relatively low, which indicates that the data points are clustered closely around the mean and hence the reliability is proved.

Furthermore, the T-Value and P-Value of the spark execution time for each dataset have been calculated. The T- values for spark execution time is 5.809, 5.906, and 5.471 for the DARPA, MACCDC, and DEFCON21 datasets respectively. The P values calculated for the DARPA dataset, MACCDC dataset, and DEFCON21 dataset was less than 0.0003 apiece, which indicates that it is statistically significant for all the 3 datasets.

5.8 Data Collected for Framework of Memory Consumption

This section presents the overall RAM memory consumption by the data collected by running the proposed algorithms and framework on three different datasets. Further, proposed SSWLOFCC algorithm has been compared with six different algorithms namely K-means, Isolation Forest, Spectral clustering, HDBSCAN, Agglomerative Clustering, Local outlier factor. In this context, the memory consumption is described as the amount of memory (RAM) consumed for the execution of specific task. The memory consumption has been measured using the developed RAM usage algorithm. This process has been executed over 4 cycles to derive an approximate amount of memory consumption. The table 5.12 shows the data derived to compare memory consumption of the proposed and existing methods with DARPA, MACCDC and DEFCON21 datasets.

From the data traces, the mean value of memory consumption for the framework for all seven algorithms in the three datasets have been, which yielded the mean values of 407.333 for K Means, 410.667 for Isolation Forest, 430.000 for Spectral Clustering, 270.000 for HDBSCAN, 235.000 for Agglomerative Clustering, 210.000 for Local Outlier Factor,

Table 5.12: Comparison of memory consumption results from proposed solutions with six different algorithms for DARPA, MACCDC, and DEFCON21 datasets.

Datasets / Algorithms	DARPA (MB)	MACCDC (MB)	DEFCON21 (MB)	Mean	Std. Deviation	T-Value	P-Value
KMeans	480.000	350.000	392.000	407.333	66.343	10.635	
IsolationForest	600.000	260.000	372.000	410.667	173.267	4.105	
Spectral Clustering	435.000	420.000	435.000	430.000	8.660	86.000	
HDBSCAN	280.000	250.000	280.000	270.000	17.321	27.000	0.0005
Agglomerative Clustering	240.000	225.000	240.000	235.000	8.660	47.000	
LOF	200.000	200.000	230.000	210.000	17.321	21.000	
SSWLOFCC	**190.000**	**183.000**	**210.000**	**194.333**	**14.012**	**24.022**	

and 194.333 for the proposed SSWLOFCC algorithm for the DARPA, MACCDC, and DEFCON21 datasets.

From the mean values the standard deviation for each dataset and each algorithm has been calculated, which yielded 66.343 for K Means, 173.267 for Isolation Forest, 8.660 for Spectral Clustering, 17.321 for HDBSCAN, 8.660 for Agglomerative Clustering, 17.321 for Local Outlier Factor, and 14.012 for the proposed SSWLOFCC algorithm for the DARPA, MACCDC, and DEFCON21 datasets. As the standard deviation values are relatively low it indicates that the data points are clustered closely around the mean, and hence their reliability has been proved.

Furthermore, the T-Value and P-Value for the memory consumption of the framework for each algorithm for all three datasets have been calculated. The T values have been recorded as: 10.635 for K Means, 4.105 for Isolation Forest, 86.000 for Spectral Clustering, 27.000 for HDBSCAN, 47.000 for Agglomerative Clustering, 21.000 for Local Outlier Factor, and 24.022 for the proposed SSWLOFCC algorithm. Additionally, the P values calculated for the DARPA dataset, MACCDC dataset, and DEFCON21 has been recorded as: 0.0005 apiece. Considering the P value, the data is considered to be statistically significant for all the 3 datasets.

The proposed SSWLOFCC algorithm has consumed lesser memory in the three datasets as against the six existing algorithms.

5.9 Conclusion

In this chapter, the performance of the proposed framework has been evaluated by implementing it in the real setup. The proposed SSWLOFCC algorithm has been implemented into the spark MLlib package by using spark API and other pipeline components. Furthermore, The chapter had described the characteristics of eight evaluation methods namely Silhouette Index, Normalized Mutual Information, adjusted rand score, Calinski and Harabaz, confusion matrix, precision, recall, and F1 score.

The chapter had presented the data collected to analyse the proposed framework on silhouette index for three different datasets (see table 5.2). For all the three datasets higher and closer to 1.0 values denote the best fit for the processed data. Likewise, tables 5.3, 5.4, and 5.5, had presented the data collected to analyse the accuracy using the adjusted rand index for three different datasets with proposed and six existing algorithms. Besides, the table 5.6 had presented the data collected to analyse the Normalized mutual info for three different datasets. Furthermore, tables 5.7, 5.8, and 5.9 had presented data collected to analyse the cluster validation of the proposed SSWLOFCC algorithm on three different datasets. Additionally the figures 5.3 to 5.11 had illustrated the confusion matrix for three different datasets on three different algorithms which has been used for evaluating other validation parameters presented in chapter 6.

The table 5.10 had illustrated the data collected to compare the execution time obtained for proposed and six different algorithms on three different datasets. Similarly the table 5.11 had presented the spark streaming execution time on three different datasets for the proposed framework. These were derived by the mathematical expression developed in chapter 4. The table 5.12 had presented the analytical results of memory consumption for the entire process execution.

In this research, three different datasets namely, DARPA, MACCDC, and DEFCON21

had been used to evaluate the quality of the proposed algorithm. The benchmarking had been done by comparing and evaluating the proposed SSWLOFCC algorithm with six different existing algorithms namely K Means, Isolation Forest, Spectral Clustering, HDBSCAN, Agglomerative Clustering, and Local Outlier Factor. Data had been collected by sampling the evaluation parameters up to 30 traces for the proposed solution. The overall evaluation had revealed that the proposed SSWLOFCC algorithm had consumed lesser memory, completed tasks with lower execution time and had yielded 96% of accuracy for anomaly detection for three different datasets. The results obtained have proved that the proposed solutions had outperformed the existing anomaly detection mechanisms. Fundamentally, the evaluation test had ensured that the proposed solutions have yielded a higher percentage of anomaly detection with much lesser memory consumption and execution time as against the other existing anomaly detection mechanisms. The next chapter presents the findings of data analysis.

CHAPTER 6: RESULTS AND DISCUSSION

This chapter presents the performance evaluation of the RTADBDT framework and elucidates the performance of the proposed algorithms that have been tested with three different datasets, and the outcome of comparative analysis with existing algorithms has also been presented. This chapter also explains the data analysis collected for this research (see Chapter 5). Additionally, the internal and external evaluation of the framework as against the experiment results has also been presented. The proposed framework has also been critically evaluated by means of various statistical techniques. Ultimately, the chapter also substantiates the efficacy of the proposed solution in terms of execution time and memory consumption by comparing it with existing algorithms.

The chapter has been organised as follows: (i) details of performance evaluation of RTADBDT framework has been presented in section 6.1; (ii) the proposed solution has been compared and contrasted with existing algorithms on three different datasets and the details has been presented in section 6.2, and the outcomes of the comparative analysis to validate the accuracy of the proposed solution as against existing algorithms using different statistics techniques has also been presented in this section;(iii) details of analysis in terms of execution time of the proposed solution as against existing algorithms on three different datasets have been presented in section 6.3; similarly the efficacy of the proposed solution in terms of memory consumption as opposed to existing algorithms have been presented in section 6.4; finally the chapter has been summarised in section 6.5.

6.1 RTADBDT Evaluation Parameters

RTADBDT is capable of delivering real-time anomaly detection based on big data technologies and composite clustering algorithms, it is an appropriate and ingenious framework suitable for uninterrupted real-time processing and also fault tolerant capable

to produce higher accuracy, and consumes lesser computational time and memory when compared to other existing approaches. The performance of RTADBDT framework has been measured based on the following parameters:

1. Accuracy
2. Execution time
3. Memory consumption

6.2 RTADBDT Performance Analysis on Accuracy

This section discusses the performance analysis carried out to verify and validate the accuracy of the proposed RTADBDT framework. An experiment was conducted to evaluate the accuracy of RTADBDT framework on simulated real-time environment on cloud. RTADBDT framework has been tested with three different datasets on proposed and various existing algorithms. Silhouette Index, normalized mutual information, Calinski and Harabaz, and adjusted rand scores were collected for the proposed framework to measure the internal and external quality of the clusters. Similarly, Precision, Recall, F1-score values were collected for true and predicted label of processed data to measure the accuracy of the anomaly detection. The following sub sections presents the comparison graphs to authenticate the results and validation of the proposed framework.

6.2.1 Silhouette Index

Figure 6.1, 6.2 and 6.3 illustrates the silhouette score of the clustering results for three different datasets. The silhouette values are reported in the graphs given below (see figures 6.1 to 6.3). The experiment with SSWLOFCC has yielded the values closer to 0.9 in all three datasets which produces compact and well-separated four clusters for analysis.

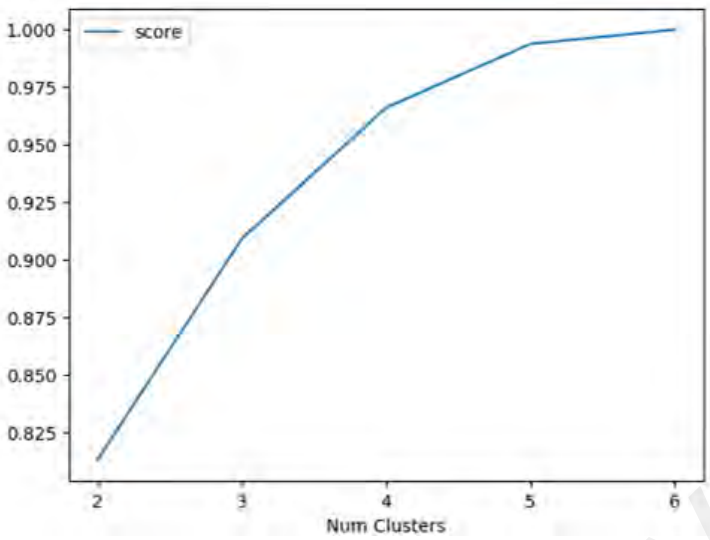

Figure 6.1: Silhouette Scoring of SSWLOFCC for DARPA dataset

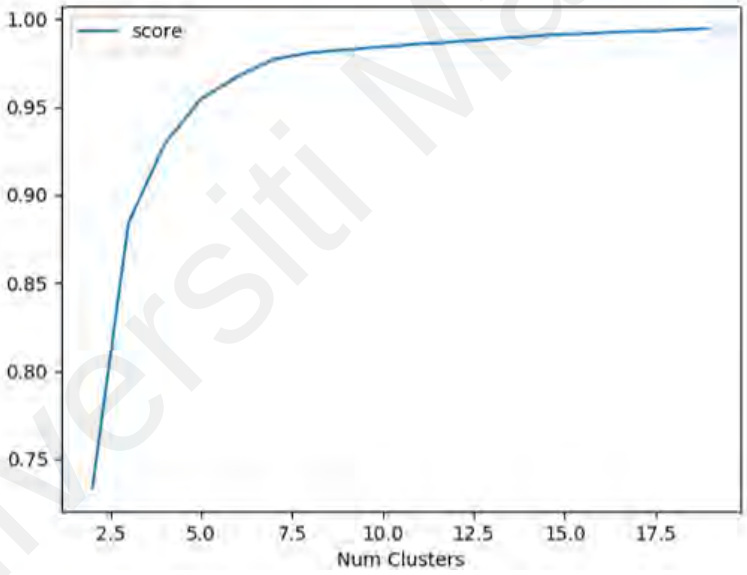

Figure 6.2: Silhouette Scoring of SSWLOFCC for MACCDC dataset

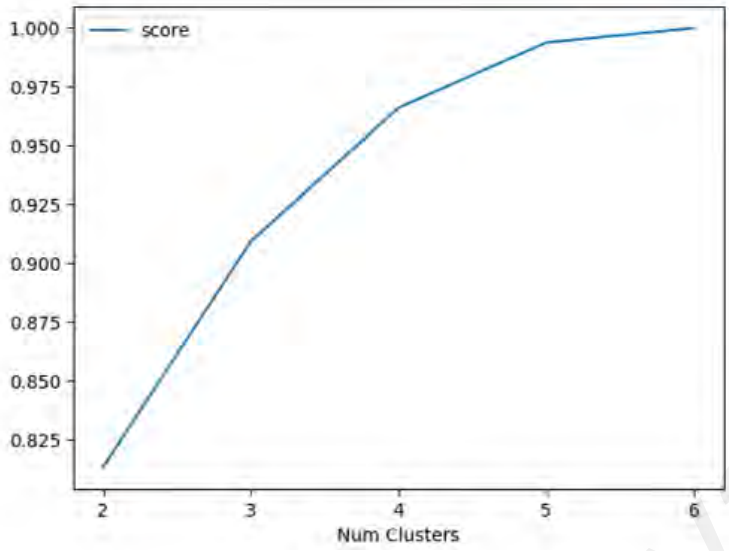

Figure 6.3: Silhouette scoring of SSWLOFCC for DEFCON21 dataset

6.2.2 Calinski and Harabaz

Figures 6.4, 6.5, and 6.6 illustrate the Calinski and Harabaz of the clustering results for three different datasets. The experiment with SSWLOFCC has yielded the values closer to 0.9 in all three datasets, which produces compact and well-separated four clusters for analysis. Internal cluster quality values derived from the Calinski and Harabaz, and silhouette scoring are closer to 0.9, which strongly proves that the coordination's were strongly distributed among the clusters. Based on the Calinski and Harabaz, and silhouette scoring analysis for three different datasets, it is evident that four clusters would be optimal for the experiment.

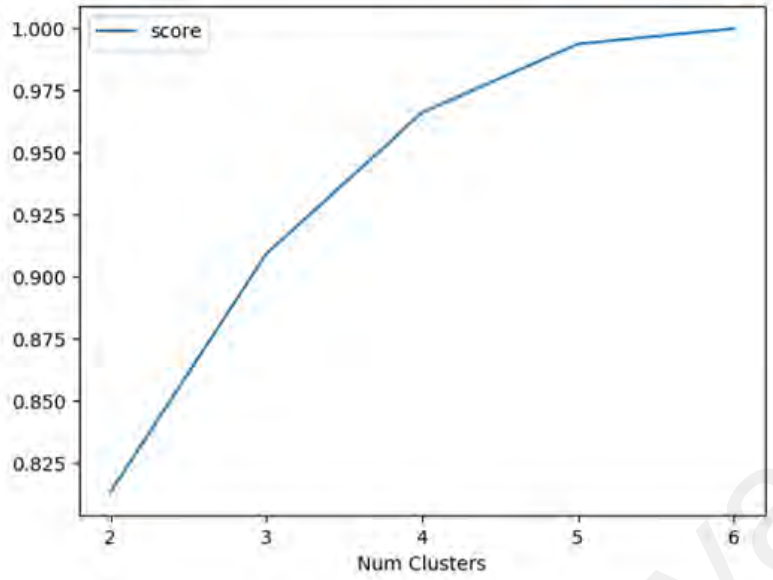

Figure 6.4: Calinski and Harabaz of SSWLOFCC for DARPA dataset

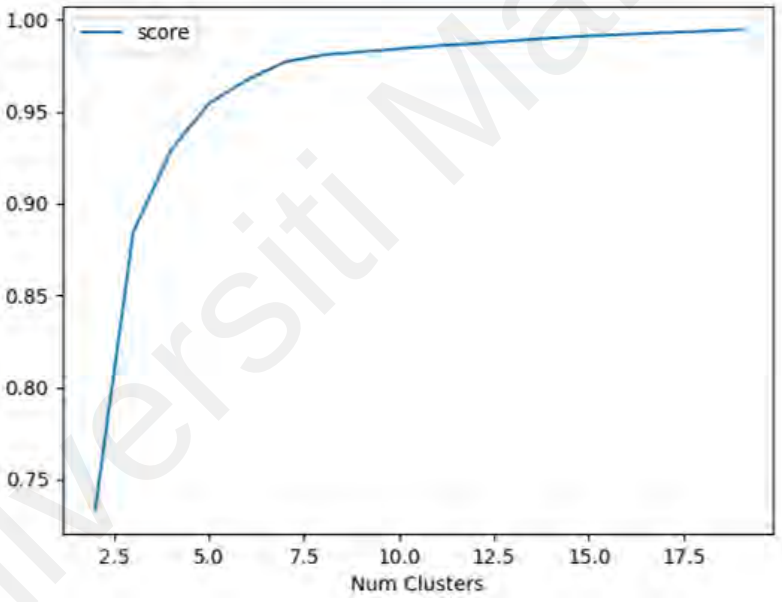

Figure 6.5: Calinski and Harabaz of SSWLOFCC for MACCDC datasets

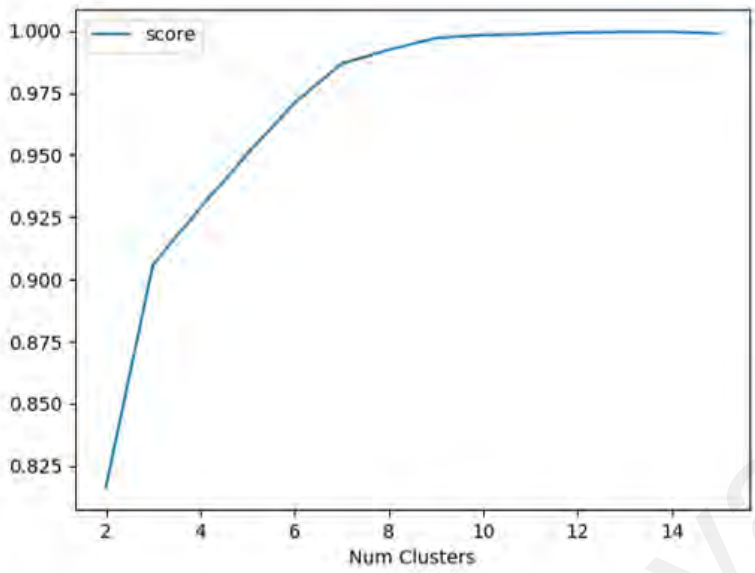

Figure 6.6: Calinski and Harabaz of SSWLOFCC for DEFCON21 datasets

6.2.3 Adjusted Rand Score

Figure 6.7 presents the comparison of external cluster quality on accuracy computed through adjusted rand index methods for five different algorithms on Darpa datasets. The y-axis shows the accuracy, whereas x-axis represents the six algorithms for thirty different data traces. The results prove that the SSWLOFCC has achieved better accuracy of 98.54 % on a DARPA datasets. SSWLOFCC has achieved higher percentages compared to other algorithms due to the composite nature of algorithm that process the clusters. The results are computed at data trace of 1 to 30 of all the processed data.

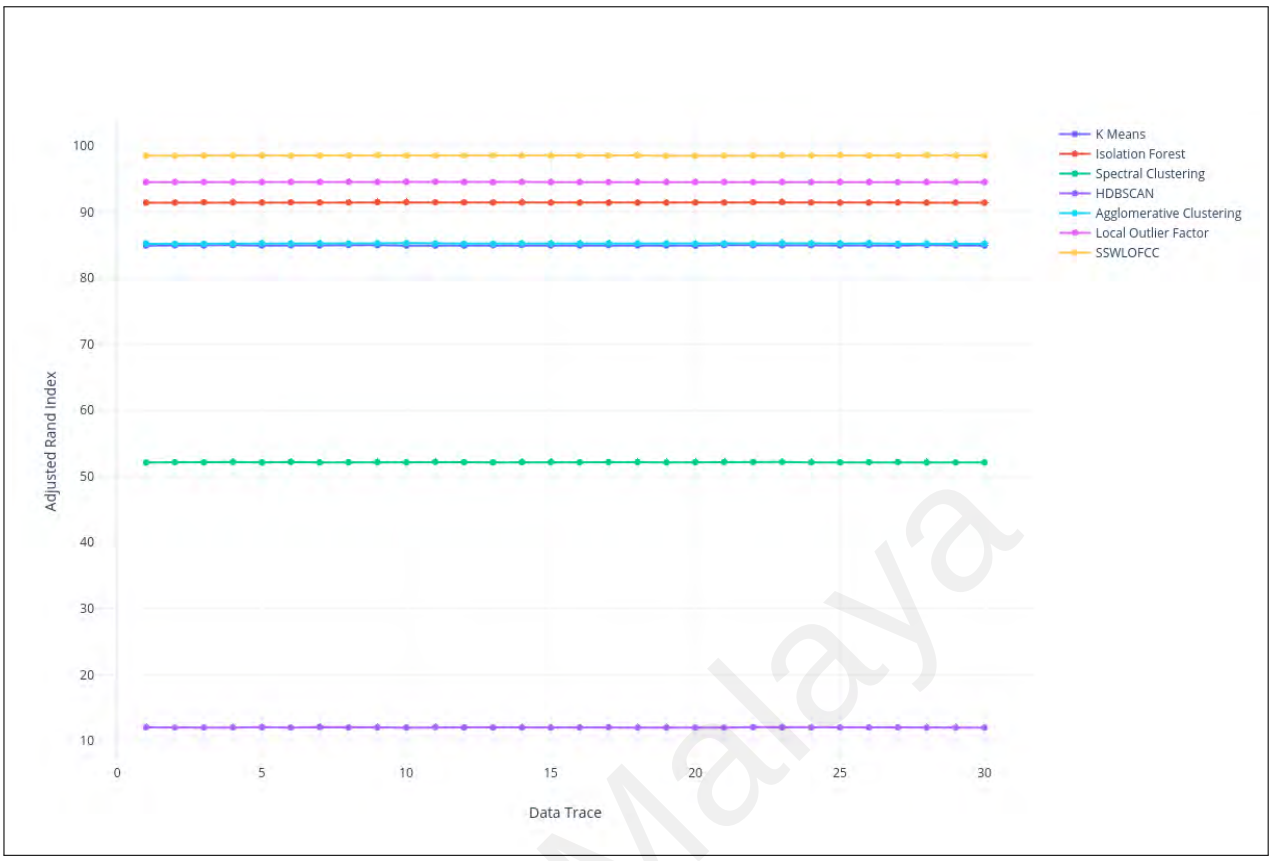

Figure 6.7: Comparison of accuracy between existing and proposed SSWLOFCC algorithms on DARPA dataset

Similarly, Figure 6.8 presents the comparison of external cluster quality on accuracy computed through adjusted rand index methods for five different algorithms on MACCDC datasets. The y-axis shows the accuracy, and the x-axis represents the six algorithms for thirty different data traces. The results prove that the SSWLOFCC has achieved better accuracy of 96.35 % on a MACCDC datasets. SSWLOFCC has achieved higher percentages compared to other algorithms due to composite nature of algorithm that process the clusters. The results are computed at data trace of 1 to 30 of all the processed data.

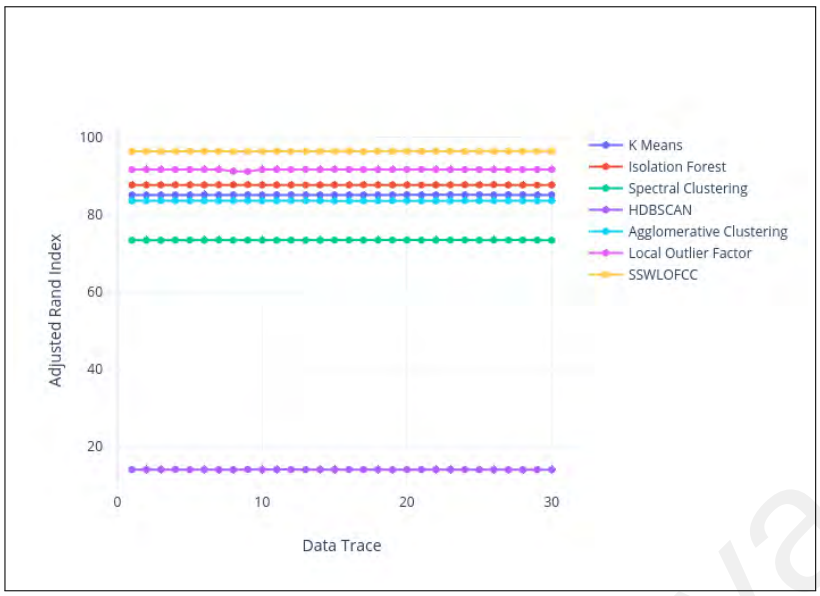

Figure 6.8: Comparison of accuracy between existing and proposed SSWLOFCC algorithms on MACCDC dataset

Likewise, figure 6.9 presents the comparison of external cluster quality on accuracy computed through adjusted rand index methods for five different algorithms on DEFCON21 datasets. The y-axis shows the accuracy and the x-axis represents the six algorithms for thirty different data traces. The results prove that the SSWLOFCC has achieved better accuracy of 94.66 % on a DEFCON21 datasets. SSWLOFCC has achieved higher percentages compared to other algorithms due to composite nature of algorithm that process the clusters. The results are computed at data trace of 1 to 30 of all the processed data.

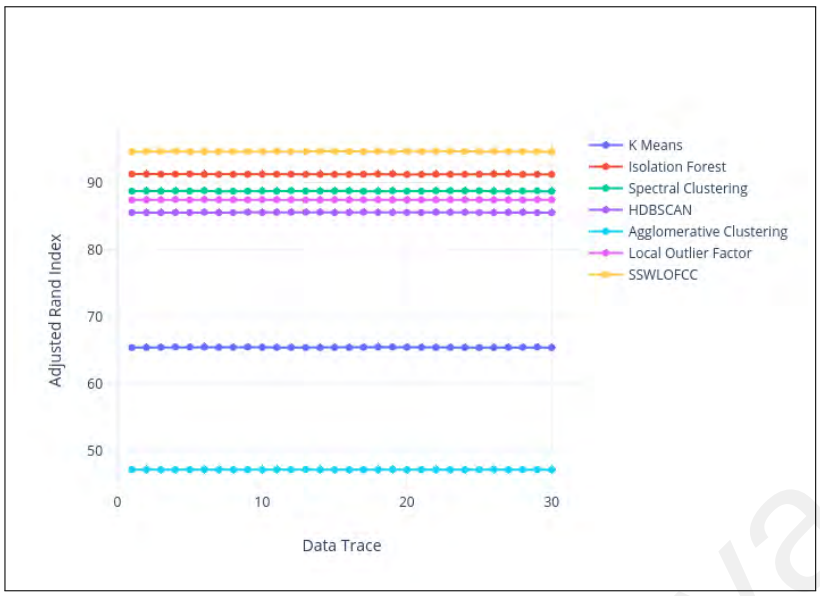

Figure 6.9: Comparison of accuracy between for existing and proposed SSWLOFCC algorithms on DEFCON21 dataset

6.2.4 Normalized Mutual Info (NMI)

This section illustrates the normalized mutual info scores generated for the proposed algorithms on three different datasets. Figure 6.10 presents that all the three datasets values are closer to 1, which indicates that the proposed algorithm has an absolute correlation with the external cluster quality. The y-axis shows the normalized mutual info score and the x-axis represents the thirty different data traces for three different datasets.

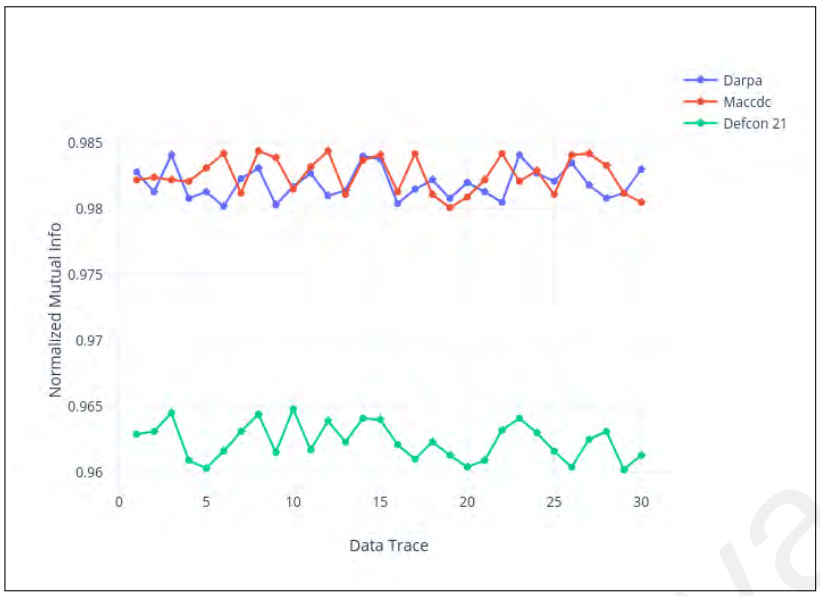

Figure 6.10: Comparison of normalized mutual info score with three different datasets on proposed SSWLOFCC algorithm

Precisely the above discussed internal and external cluster evaluation metrics has revealed that the proposed SSWLOFCC algorithm has yielded significantly higher values in all the selected performance metrics as against the other existing methods.

6.2.5 Precision

This section presents the precision value generated for the proposed algorithms on three different datasets. Precision also known as positive predicted value is the number of relevant instances among the retrieved instances. In simpler terms, out of all the classes how many has been predicted correctly. The precision is measured with a score from 0 to 1. The precision has been classified as follows:

1. 0 - 0.25 Very Weakly Predicted
2. 0.26 - 0.50 Weakly Predicted
3. 0.51 - 0.75 Well Predicted
4. 0.76 – 1.00 Very Well Predicted

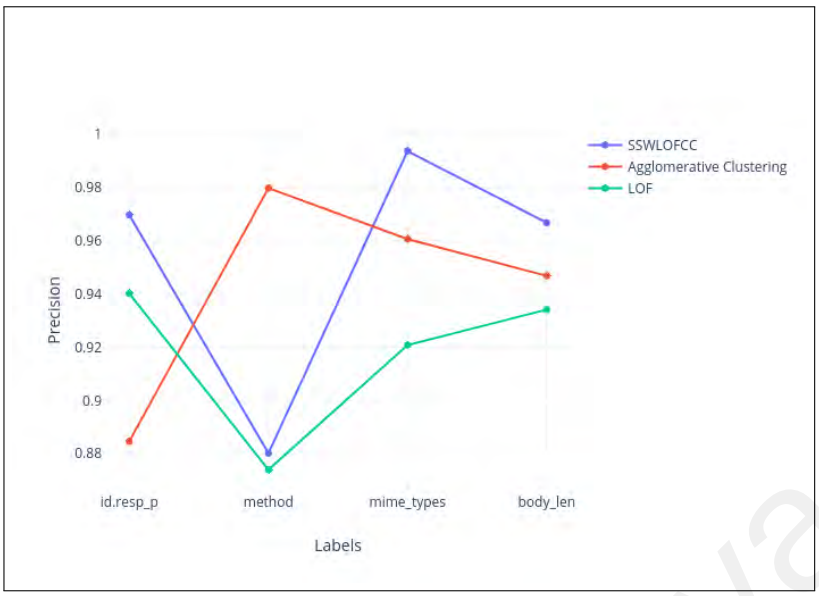

Figure 6.11: Precision score for DARPA dataset

Figure 6.11 and Table 6.1 shows the precision score results which is very well predicted for DARPA dataset.

Table 6.1: Precision score for DARPA dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.969709	Very Well Predicted	0.884615	Very Well Predicted	0.940171	Very Well Predicted
method	0.880156	Very Well Predicted	0.97981	Very Well Predicted	0.874001	Very Well Predicted
mime_types	0.993739	Very Well Predicted	0.960657	Very Well Predicted	0.920826	Very Well Predicted
body_len	0.966785	Very Well Predicted	0.946828	Very Well Predicted	0.934132	Very Well Predicted

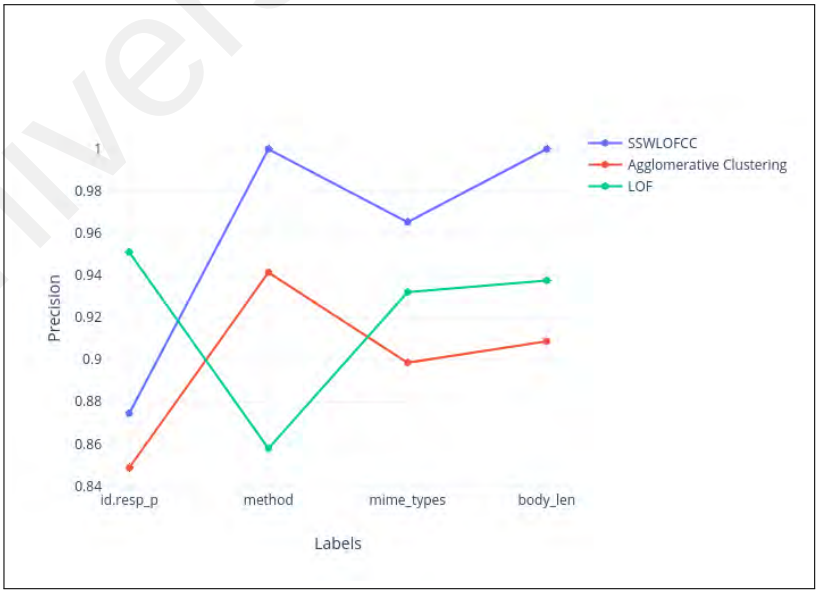

Figure 6.12: Precision score for MACCDC dataset

Figure 6.12 and Table 6.2 shows the precision score results which is very well predicted

for MACCDC dataset.

Table 6.2: Precision score for MACCDC dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.874516	Very Well Predicted	0.848733	Very Well Predicted	0.951017	Very Well Predicted
method	1.00	Very Well Predicted	0.94145	Very Well Predicted	0.857843	Very Well Predicted
mime_types	0.965323	Very Well Predicted	0.898564	Very Well Predicted	0.932048	Very Well Predicted
body_len	1.00	Very Well Predicted	0.908762	Very Well Predicted	0.937556	Very Well Predicted

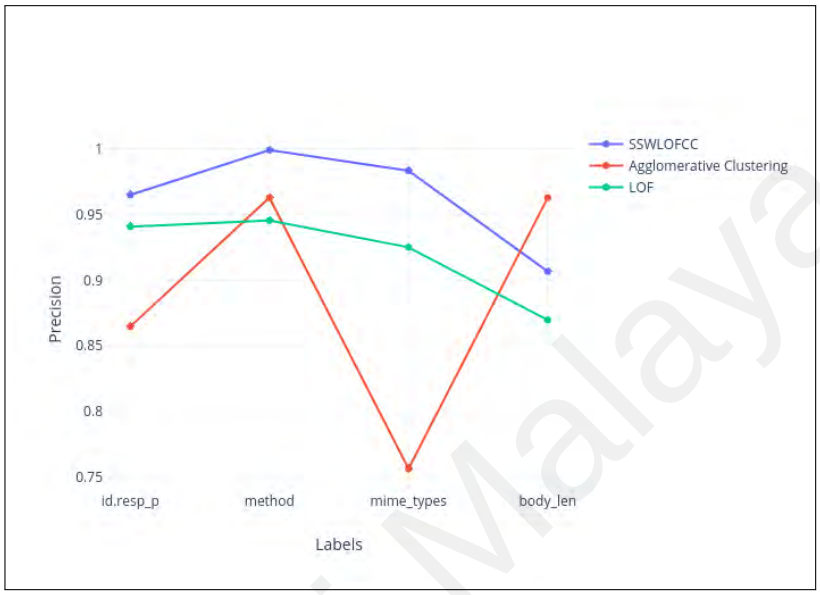

Figure 6.13: Precision score for DEFCON21 dataset

Figure 6.13 and Table 6.3 shows the precision score results which is very well predicted for DEFCON21 dataset.

Table 6.3: Precision score for DEFCON21 dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.96495	Very Well Predicted	0.864695	Very Well Predicted	0.940792	Very Well Predicted
method	0.999086	Very Well Predicted	0.962829	Very Well Predicted	0.945469	Very Well Predicted
mime_types	0.983366	Very Well Predicted	0.756477	Very Well Predicted	0.925064	Very Well Predicted
body_len	0.906656	Very Well Predicted	0.962684	Very Well Predicted	0.869613	Very Well Predicted

Details on the precision scores received for the three datasets have been presented in Tables 6.1, 6.2, and 6.3. This study has achieved a precision score classification of "Very Well Predicted" for all the classes/labels in all the datasets.

As seen in figures 6.11 to 6.13, some of the parameters may have seen a decrease in precision score for each dataset. However, this is heavily dependent on the incoming data stream, since our proposed model is based on real-time. At the time of running our model,

the data that was processed may have had a possible variation in the true positive and the false positive values. On the other hand, comparing all the four parameters on the three datasets makes it clearly evident that the SSWLOFCC has outperformed all of its rivals.

6.2.6 Recall

This section presents the Recall value generated for the proposed algorithms on three different datasets. Recall which is also known as sensitivity is the number of relevant instances retrieved over the total amount of relevant instances. Simply, from the positive classes how many have been predicted correctly. Recall is measured with a score from 0 to 1. Recall has been classified as follows:

1. 0 - 0.25 Very Weakly Predicted
2. 0.26 - 0.50 Weakly Predicted
3. 0.51 - 0.75 Well Predicted
4. 0.76 – 1.00 Very Well Predicted

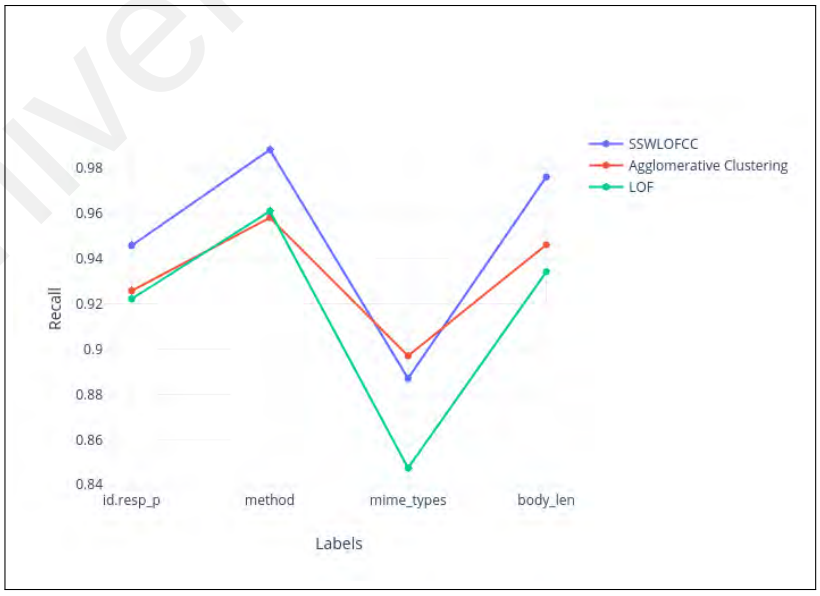

Figure 6.14: Recall score for DARPA dataset

Figure 6.14 and Table 6.4 shows the recall score results which is very well predicted for DARPA dataset.

In Figure 6.14, SSWLOFCC has outperformed the other two algorithms for id.resp_p, method and body_len parameter in the DARPA dataset. However, there is a slight reduction in the recall score for SSWLOFCC as compared to the Agglomerative Clustering algorithm within mime_types parameter. This indicate that Agglomerative Clustering algorithm performed better than SSWLOFCC, due to the variation in the true positive and false negative, which is heavily affected by the incoming stream data. However, it occurs only temporary since the algorithm stop performing in our framework.

Table 6.4: Recall score for DARPA dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.945709	Very Well Predicted	0.925709	Very Well Predicted	0.922156	Very Well Predicted
method	0.988024	Very Well Predicted	0.958025	Very Well Predicted	0.960878	Very Well Predicted
mime_types	0.887026	Very Well Predicted	0.897026	Very Well Predicted	0.847525	Very Well Predicted
body_len	0.976048	Very Well Predicted	0.946048	Very Well Predicted	0.934132	Very Well Predicted

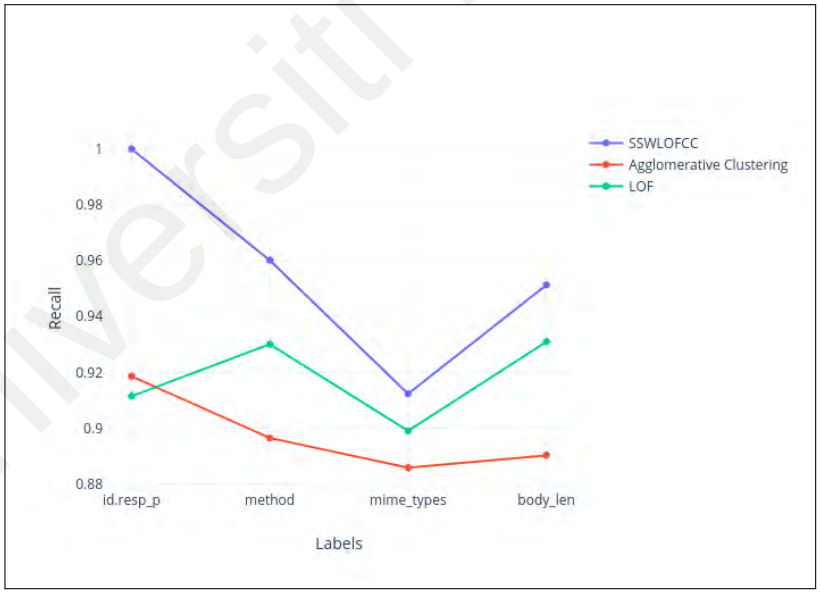

Figure 6.15: Recall score for MACCDC dataset

Figure 6.15 and Table 6.5 shows the recall score results which is very well predicted for MACCDC dataset.

Figure 6.16 and Table 6.6 shows the recall score results which is very well predicted for DEFCON21 dataset.

Table 6.5: Recall score for MACCDC dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	1.00	Very Well Predicted	0.918584	Very Well Predicted	0.911426	Very Well Predicted
method	0.960142	Very Well Predicted	0.89646	Very Well Predicted	0.930027	Very Well Predicted
mime_types	0.912312	Very Well Predicted	0.885841	Very Well Predicted	0.899026	Very Well Predicted
body_len	0.951284	Very Well Predicted	0.890265	Very Well Predicted	0.930912	Very Well Predicted

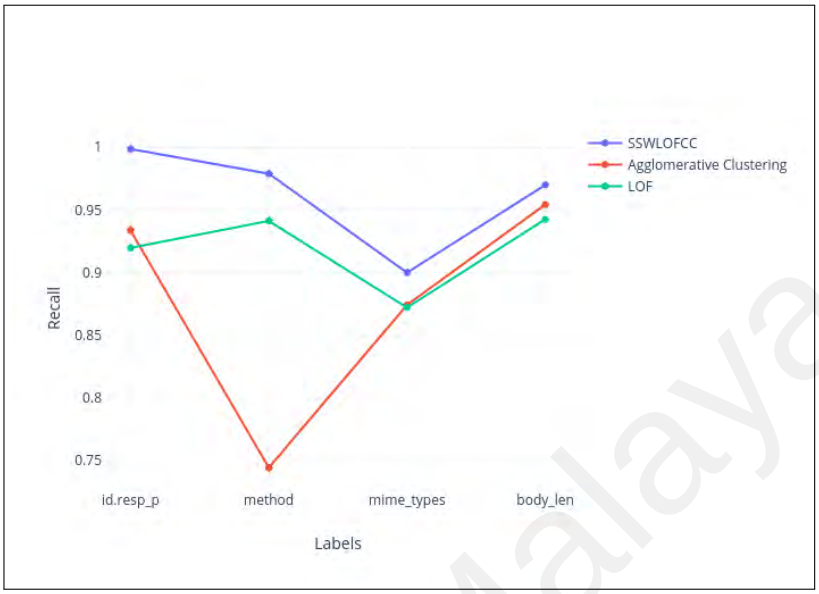

Figure 6.16: Recall score for DEFCON21 dataset

Table 6.6: Recall score for DEFCON21 dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.998657	Very Well Predicted	0.933733	Very Well Predicted	0.91976	Very Well Predicted
method	0.978952	Very Well Predicted	0.744511	Very Well Predicted	0.941317	Very Well Predicted
mime_types	0.900134	Very Well Predicted	0.874251	Very Well Predicted	0.872255	Very Well Predicted
body_len	0.969996	Very Well Predicted	0.954349	Very Well Predicted	0.942515	Very Well Predicted

The tables 6.4, 6.5 and 6.6 presented above have detailed the recall scores received for the three datasets. This study has achieved a recall score classification of "Very Well Predicted" for all the classes/labels in all the datasets. The recall scores indicate that the proposed algorithm SSWLOFCC has performed very well compared to the other existing algorithms in predicating the data correctly.

6.2.7 F1 Score

This section presents the F1 score value generated for the proposed algorithms on three different datasets.

F1 score is the measure of the test's accuracy. Precision and recall are used to compute the F1 score. The F1 score is said to be the harmonic average of precision and recall. The

F1 score ranges from 0 to 1.

The F1 score has been classified as follows:

1. 0 - 0.25 Very Low
2. 0.26 - 0.50 Low
3. 0.51 - 0.75 Good
4. 0.76 – 1.00 Very Good

In Figure 6.17 *method* parameter has seen a reduction in the F1 score for the SSWLOFCC. Nonetheless, the SSWLOFCC F1 score higher for other parameters. The F1 score measures the tests accuracy and is dependent on the precision and recall values. As discussed in the precision and recall section, the true positive, false positive and false negative based on the incoming data streams have caused the increase in F1 score for Agglomerative Clustering within method parameter.

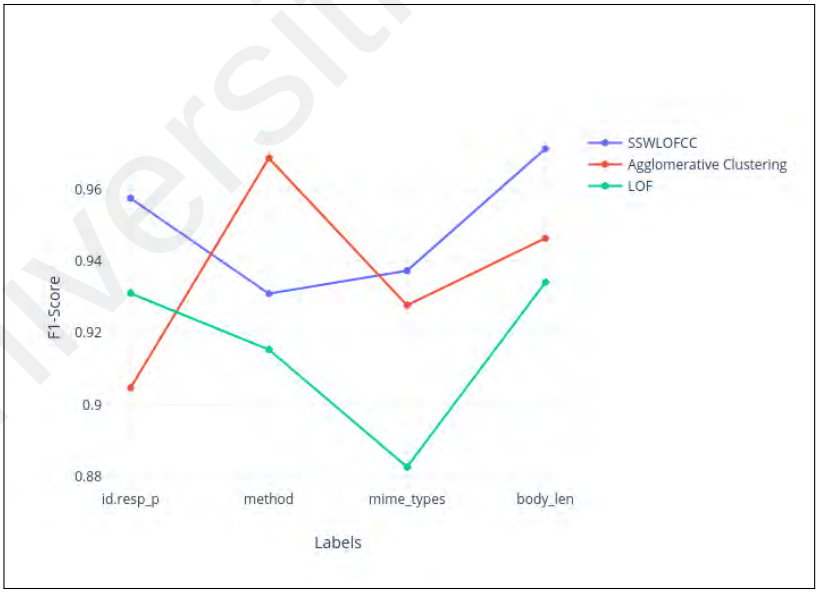

Figure 6.17: F1 score for DARPA dataset

Figure 6.17 and Table 6.7 shows the F1 score results which is classified as very good for DARPA dataset.

Table 6.7: F1 score for DARPA dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.957559	Very Good	0.904696	Very Good	0.931076	Very Good
method	0.930976	Very Good	0.968795	Very Good	0.915383	Very Good
mime_types	0.937355	Very Good	0.927752	Very Good	0.882656	Very Good
body_len	0.971395	Very Good	0.946438	Very Good	0.934132	Very Good

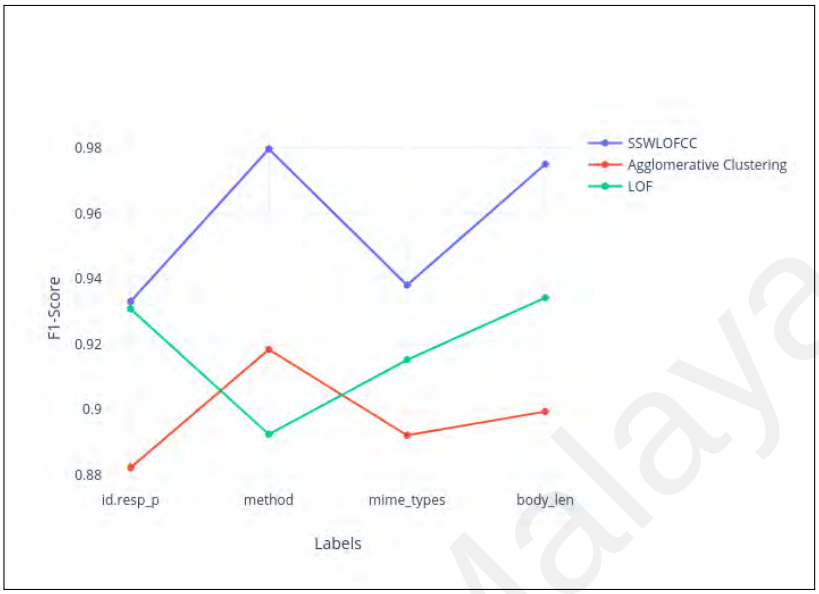

Figure 6.18: F1 score for MACCDC dataset

Table 6.8: F1 score for MACCDC dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.933058	Very Good	0.882278	Very Good	0.930801	Very Good
method	0.979666	Very Good	0.918404	Very Good	0.892478	Very Good
mime_types	0.938069	Very Good	0.892157	Very Good	0.915239	Very Good
body_len	0.975034	Very Good	0.899419	Very Good	0.934222	Very Good

Figure 6.18 and Table 6.8 shows the F1 score results which is classified as very good for MACCDC dataset. Figure 6.19 and Table 6.9 shows the F1 score results which is classified as very good for DEFCON21 dataset. In Figure 6.19 *body_len* parameter has seen a reduction in the F1 score for the SSWLOFCC. Nonetheless, the SSWLOFCC F1 score higher for other parameters. The F1 score measures the tests accuracy and is dependent on the precision and recall values. As discussed in the precision and recall section, the true positive, false positive and false negative based on the incoming data streams have caused the increase in F1 score for Agglomerative Clustering within *body_len* parameter.

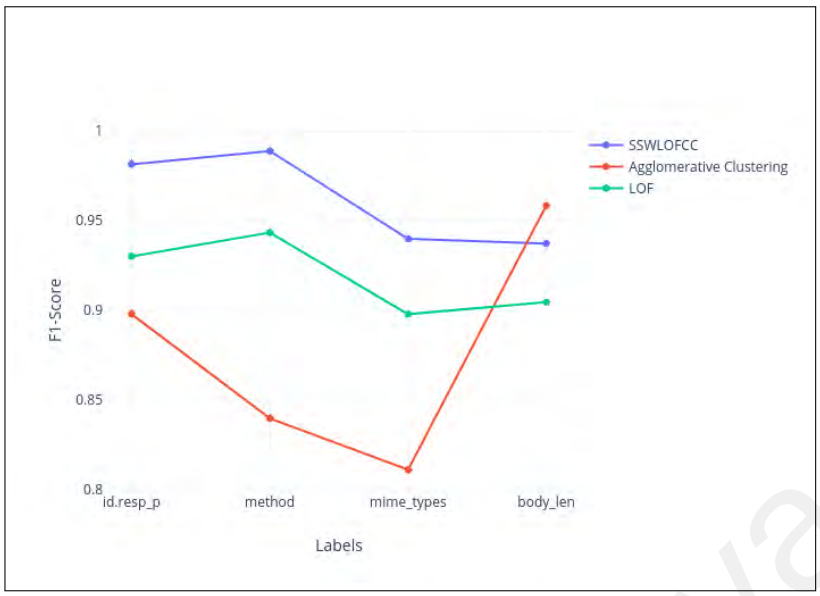

Figure 6.19: F1 score for DEFCON21 dataset

Table 6.9: F1 Score for DEFCON21 Dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.981514	Very Good	0.897889	Very Good	0.930157	Very Good
method	0.988917	Very Good	0.839712	Very Good	0.943389	Very Good
mime_types	0.939911	Very Good	0.811111	Very Good	0.897884	Very Good
body_len	0.937257	Very Good	0.958498	Very Good	0.904598	Very Good

The above tables 6.7, 6.8 and 6.9 detail on the F1 scores received for the three datasets. This study has achieved a F1 score classification of "Very Good" for all classes/labels in all the datasets. The F1 scores indicate that the proposed algorithm SSWLOFCC has performed better compared to the other existing algorithms.

6.2.8 Matthews's Correlation Coefficient

This section presents the Matthews's correlation coefficient generated for the proposed algorithms on three different datasets. Matthews's correlation coefficient (MCC) is used to measure the quality of binary classifications. The MCC is a correlation coefficient of the observed and predicted binary classifications. A value of -1 to +1 is returned. The MCC can be classified into 3 categories as follows.

1. +1 Perfect Prediction
2. 0 No Better than Random Prediction

3. -1 Total Disagreement between Prediction and Observation

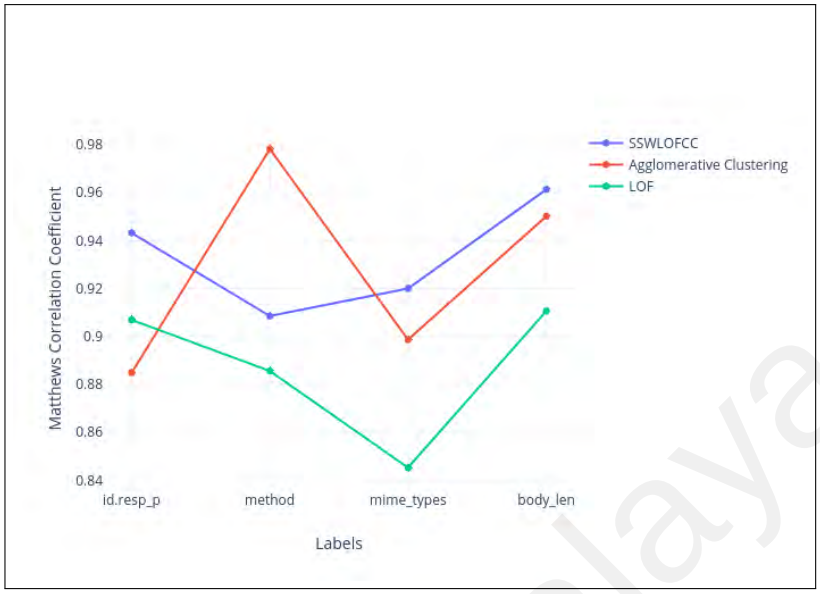

Figure 6.20: Matthews correlation coefficient for DARPA dataset

Figure 6.20 and Table 6.10 shows the Matthews correlation coefficient score results which is classified as close to perfect prediction for DARPA dataset. In Figure 6.20 *method* parameter has seen a reduction in the MCC value for the SSWLOFCC. This is because it is impacted by the true positive, false positive and false negative values gained from the confusion matrix. The confusion matrix is impacted by the incoming data streams as they come in real time. However, it is evident that SSWLOFCC has taken the lead against other algorithms.

Table 6.10: Matthews correlation coefficient for DARPA dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.943206133	Close to Perfect Prediction	0.884875999	Close to Perfect Prediction	0.906928924	Close to Perfect Prediction
method	0.908507189	Close to Perfect Prediction	0.978192445	Close to Perfect Prediction	0.885627382	Close to Perfect Prediction
mime_types	0.920047044	Close to Perfect Prediction	0.898660664	Close to Perfect Prediction	0.845277016	Close to Perfect Prediction
body_len	0.961327957	Close to Perfect Prediction	0.968157141	Close to Perfect Prediction	0.910634157	Close to Perfect Prediction

Figure 6.21 and Table 6.11 shows the Matthews correlation coefficient score results which is classified as close to perfect prediction for MACCDC dataset.

Figure 6.22 and Table 6.12 shows the Matthews correlation coefficient score results which is classified as close to perfect prediction for DEFCON21 dataset. In Figure 6.22

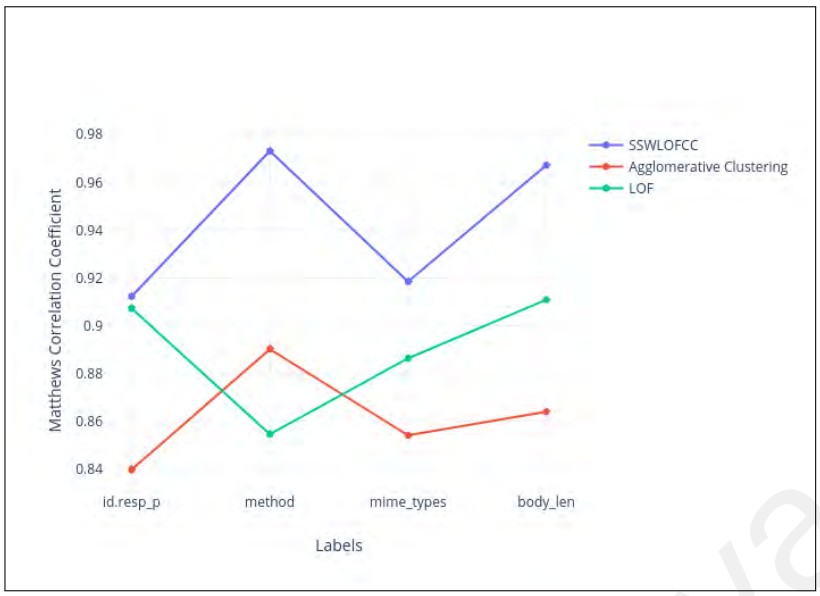

Figure 6.21: Matthews correlation coefficient for MACCDC dataset

Table 6.11: Matthews correlation coefficient for MACCDC dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.912264138	Close to Perfect Prediction	0.839757002	Close to Perfect Prediction	0.907224179	Close to Perfect Prediction
method	0.973119214	Close to Perfect Prediction	0.890206794	Close to Perfect Prediction	0.854595198	Close to Perfect Prediction
mime_types	0.918467159	Close to Perfect Prediction	0.854124969	Close to Perfect Prediction	0.886355702	Close to Perfect Prediction
body_len	0.967171117	Close to Perfect Prediction	0.86395696	Close to Perfect Prediction	0.910874609	Close to Perfect Prediction

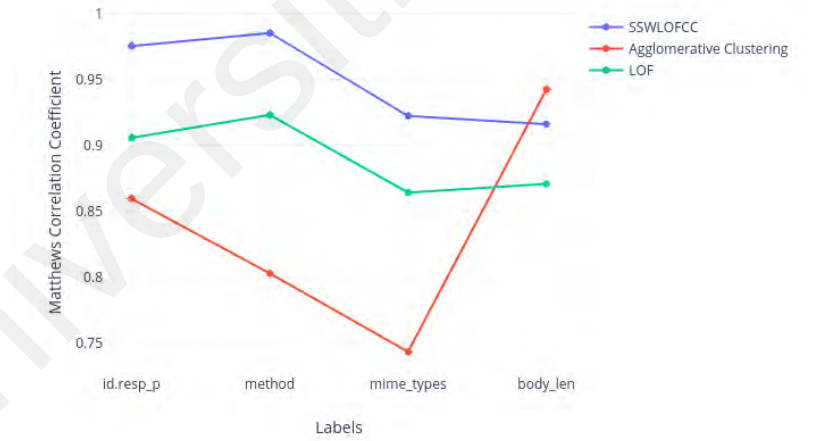

Figure 6.22: Matthews correlation coefficient for DEFCON21 dataset

body_len parameter has seen a reduction in the MCC value for the SSWLOFCC. This is also due to the true positive, false positive and false negative values gained from the confusion matrix. The confusion matrix is impacted by the incoming data streams as they come in real time. However, it is evident that SSWLOFCC has taken the lead against other

algorithms.

Table 6.12: Matthews correlation coefficient for DEFCON21 dataset

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
id.resp_p	0.975221989	Close to Perfect Prediction	0.859755051	Close to Perfect Prediction	0.90582175	Close to Perfect Prediction
method	0.985199249	Close to Perfect Prediction	0.80317405	Close to Perfect Prediction	0.923154068	Close to Perfect Prediction
mime_types	0.922376755	Close to Perfect Prediction	0.743864644	Close to Perfect Prediction	0.864413061	Close to Perfect Prediction
body_len	0.916177088	Close to Perfect Prediction	0.942359308	Close to Perfect Prediction	0.870903105	Close to Perfect Prediction

The above tables 6.10, 6.11, and 6.12 detail the MCC scores received for the three datasets. This study has achieved a MCC score classification close to "Perfect Prediction" for all classes/labels in all the datasets. Comparatively, the proposed algorithm SSWLOFCC has values closer to perfect prediction than the other existing algorithms.

6.2.9 Kappa

This section illustrates the Kappa values generated for proposed algorithms on three different datasets. Kappa is a statistical method that measures inter-rater agreement for qualitative items. It is a robust measure compared to percent agreement calculation. Kappa can be classified into 5 categories as follows.

1. < 0.20 Poor
2. 0.21 - 0.40 Fair
3. 0.41 - 0.60 Moderate
4. 0.61 - 0.80 Good
5. 0.81 - 1.00 Very Good

Table 6.13: Kappa value for DARPA, MACCDC, and DEFCON21 datasets

	SSWLOFCC	Classification	Agglomerative Clustering	Classification	LOF	Classification
Darpa	0.932	Very Good	0.608	Good	0.888	Very Good
Maccdc	0.941	Very Good	0.864	Very Good	0.89	Very Good
Defcon21	0.949	Very Good	0.836	Very Good	0.892	Very Good

The Figure 6.23 and table 6.13 have presented the Kappa values received for the three datasets. This study has achieved a Kappa score classification of "Very Good" for most of

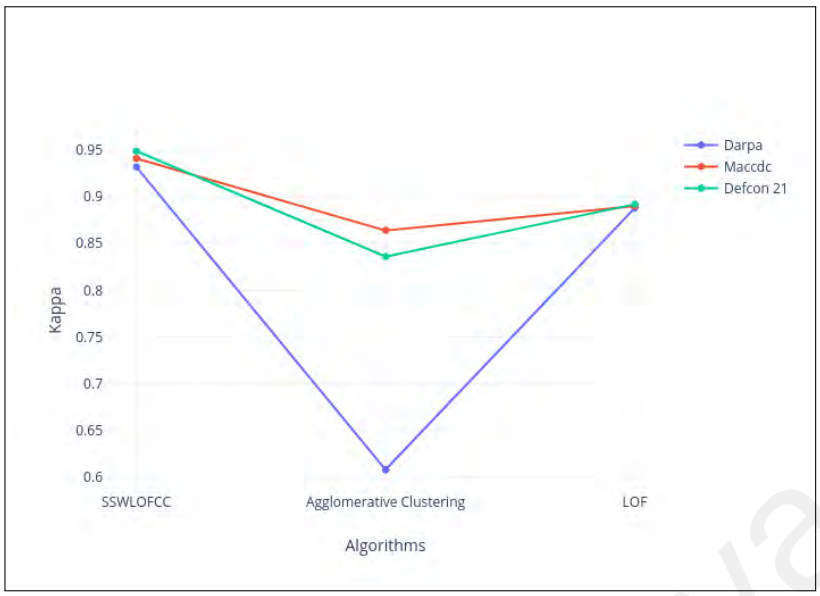

Figure 6.23: Kappa for all three datasets

classes/labels in all the datasets. Comparatively, the proposed algorithm SSWLOFCC has performed better than the other existing algorithms.

6.3 RTADBDT Performance Analysis on Execution Time

This section presents the performance analysis carried out to validate the execution time of the proposed RTADBDT framework with five existing algorithms on three different datasets. The mathematical expression developed has been used to calculate the Spark Streaming execution time, and memory profiler algorithm has been used to calculate entire execution time of algorithms in the framework.

Three different datasets have been employed to perform a comparative analysis for the proposed and existing agglomerative clustering and LOF algorithms without modification. Figure 6.24 illustrates the impact of execution time in the real time anomaly detection in big data technologies.

Based on the figure 6.24 it is evident that the algorithm proposed in this study has achieved an execution time of 13 seconds compared to the agglomerative, which took 15 seconds, while the LOF took 14 seconds in the DARPA dataset for 4 cycles.,

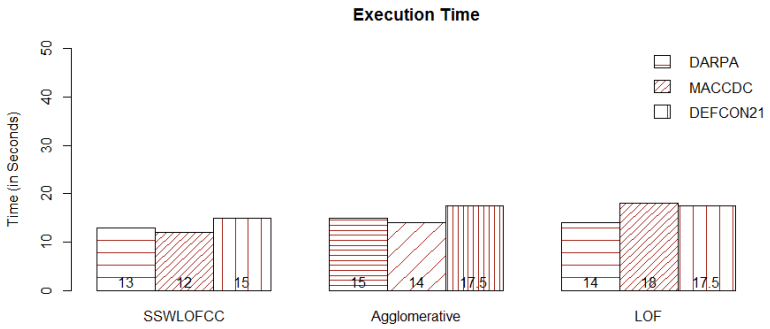

Figure 6.24: Execution time for proposed SSWLOFCC compared with two existing algorithm in three different datasets

In contrast, the comparison of execution time within the MACCDC dataset for 4 cycles indicate that the proposed algorithm has an execution time of 12 seconds as compared to the agglomerative, which took 14 seconds and the LOF with 18 seconds in the MACCDC dataset for 4 cycles.

Furthermore, within the DEFCON21 dataset for 4 cycles, the result indicate that the proposed algorithm has achieved an execution time of 15 seconds as against the agglomerative and the LOF algorithms, where both have resulted in 17.5 seconds of execution time apiece.

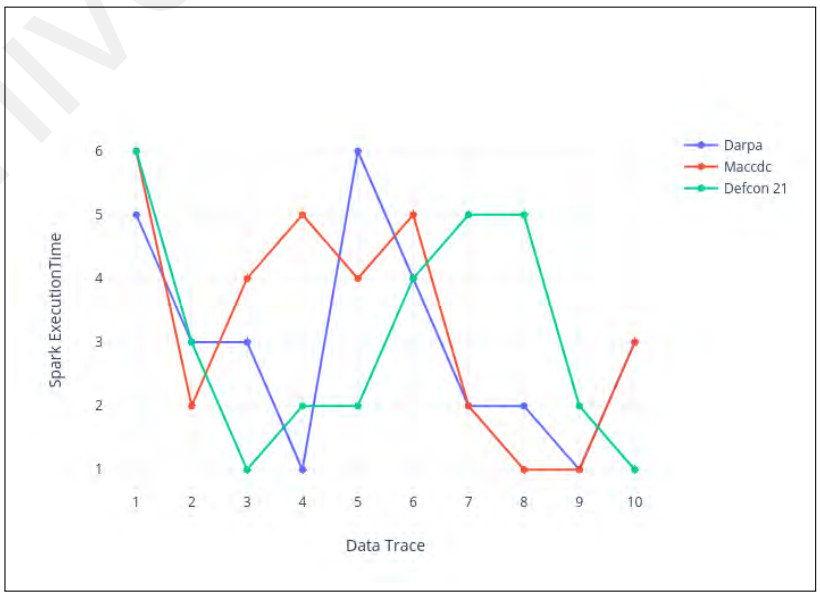

Figure 6.25: Spark Streaming execution time for proposed SSWLOFCC algorithm on three different datasets

The figure 6.25 shows the execution time taken by Spark for each dataset. The datasets seem to have different execution time each data trace, this is because each job that is executed on Spark has a scheduling delay and processing time. The total execution time for each data trace is the scheduling delay added with the processing time. A scheduling delay is the time taken by the scheduler to submit the jobs of the batch. On the other hand, the processing time is the time taken to compute a given batch for all its jobs. Since each batch have different scheduling delay and processing time, each data trace has a different execution time as shown in the above figure.

The above discussions clearly indicate that due to the modification made in the pooling_func, contamination and n_jobs, the algorithms proposed in this study execute within shorter time period as against the agglomerative and LOF algorithms. In addition, the Spark Streaming provides multiple executor to process the jobs, which leads to the reduction of the execution time in the proposed algorithm.

6.4 RTADBDT Performance Analysis on Memory Consumption

This section presents the performance analysis carried out to validate the memory consumption of the proposed RTADBDT framework with five existing algorithms on three different datasets. Memory profiler algorithm has been used for calculation of memory consumption of algorithms in the framework for four cycle and the memory consumption per second for the algorithms has been derived.

Three different datasets have been used to comparatively analyse the proposed and existing agglomerative clustering and LOF algorithms without modification. Fig 6.26 shows the impact of memory consumption towards the real time anomaly detection in big data technologies.

Based on the figure 6.26 it is evident that the algorithm proposed in this study has consumed lesser memory consumption (190 MB) as compared to the agglomerative and

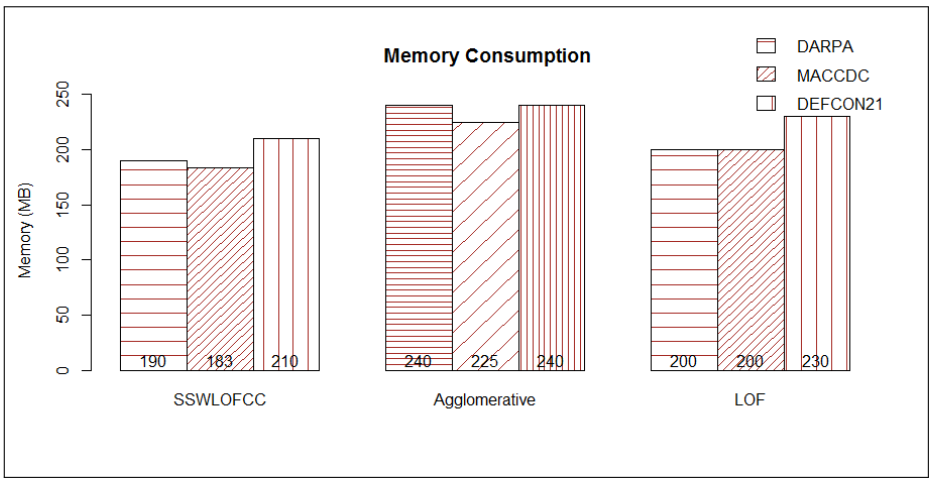

Figure 6.26: Memory consumption for proposed SSWLOFCC compared with two existing algorithm in three different dataset

the LOF algorithms with 240 MB and 200 MB respectively in the DARPA dataset for 4 cycles. In addition, the analysis has also proved that the proposed algorithm has consumed lesser memory (183 MB) as against the agglomerative and the LOF algorithms with 225 MB and 200 MB respectively in the MACCDC dataset for 4 cycles.

On the other hand, it is also evident that the proposed algorithm has utilized lesser memory (210 MB) as compared to the agglomerative and the LOF algorithms which took 240 MB and 230 MB in the DEFCON21 dataset for 4 cycles, respectively.

The above results substantiate that the proposed algorithm has consumed less memory as against other existing algorithms tested in this study. The achievement can be attributed to the embedding of the proposed algorithm into Spark Streaming, which has enabled in-memory shared cache to make it easy to connect to the real time input data into the part of application. Furthermore, Scala application has provided IgniteRDD (resilient distributed dataset) which allows to share in-memory data.

6.5 Conclusion

In this chapter, we evaluated the performance of RTADBDT by delivering the outcomes of our experiments on a real setup, in graphic representation. We measure the accuracy,

memory consumption and execution time of RTADBDT on three different datasets. Four well-known evaluation metrics including Silhouette Index, normalized mutual information, Calinski and Harabaz, and adjusted rand scores, Precision, Recall, F1-score, and Kappa had been adopted to prove the efficacy of the proposed algorithm. Furthermore, the evaluation of three different datasets using the quality metrics has revealed that the proposed SSWLOFCC had attained optimal results against other existing algorithms.

The proposed solution was measured for three different datasets which produced average internal and external quality percentage of 94.33% on silhouette index, 95.83% on Calinski and Harabaz, 96.52% on adjusted rand index and 97.5% on normalized mutual info. Moreover, accuracy measurement produced by statistical techniques such as precision, recall, Fl predicated well for the selected parameters. Likewise, Mathew's correlation coefficient and Kappa also close to the prediction on the selected parameters. Most of the evaluation parameters were critically analyzed and compared with five different existing algorithms to prove the efficiency with the proposed solution.

We have measured overall and Spark Streaming job execution time in seconds. While the former execution time is 13.3 seconds for the entire cycle, the latter processed within 3.13 seconds which is far less than existing algorithms. On the other hand, within the memory consumption measurement average for three different datasets of 194.3 MB and 14.61MB per second which also lower than the other algorithms on the framework.

Likewise, by varying three different datasets to process in proposed SSWLOFCC measured the quality of the clusters. In addition, the statistical analysis determined the selected range of parameter provide the best prominent results.

The complexity analysis RTADBDT shows that the time and memory consumption for the computation is less than the existing approach. It is worth to note that, despite the lower computation of RTADBDT, the accuracy obtained is much higher than that obtained

by another existing approach.

The results of the comparative analysis between the proposed and other existing real-time anomaly detection algorithms substantiate that the RTADBDT is a proficient and capable processing framework for real-time anomaly detection with lesser memory consumption and execution time.

CHAPTER 7: CONCLUSION

This chapter presents the conclusive remarks and summary of the research performed in this thesis. It analyses the aims and objectives of the research, by re-examining them it assures that the research has accomplished its aims and objectives. Furthermore, the significance and limitations of their research have also been emphasized, followed by the presentation of the list of journals published in the course of this research process. Lastly, suggestions for future research directions have also been presented.

The chapter comprises the following sections: section 7.1 re-examines the objectives of this research work, section 7.2 highlights the contribution of the research, section 7.3 presents the list of journals published in the course of this research process, section 7.4 exhibits the significance and limitations of the proposed solution / research, and ultimately the section 7.5 recommends the direction of the future works in this domain.

7.1 Reappraisal of the Research Objectives

This research had aimed at improving the performance of anomaly detection by integrating real-time big data technologies based on composite clustering algorithms. In this section, the aims and objectives of the research have been reinvestigated to determine their accomplishment in this thesis. The four objectives derived in research (See chapter 1) have been re-examined and interpreted to establish how this research has fulfilled those objectives.

The first objective was aimed at reviewing the state-of-the-art on anomaly detection and real-time big data technologies with respect to detection performance issue. This had been accomplished by critically analyzing the numerous problems constraining the adoption of real-time anomaly detection using big data technologies and reviewing corresponding results by creating a taxonomy (See Chapter 2 - Section 2.6 - Figure 2.2). Furthermore,

to study the state-of-the-art research carried out in the RTADBDT, several online digital libraries incorporating Elsevier, IEEE, ACM and Springer had been utilized. Consequently, various review and research papers in the broad spectrum of RTADBDT was analysed. Qualitative and quantitative analyses was conducted on chosen state-of-the-art anomaly detection and real-time big data technologies to identify the open issues and research challenges. Lastly, diverse open research issues in these domains were detected.

The second objective was aimed at investigating the impact of various existing solutions for anomaly detection on real-time data to establish the research problem. To achieve this objective original data traces of anomaly detection method in a traditional approach was collected. Further, three different datasets were evaluated to analyse the performance of the existing approaches. The experiment was implemented in the lab environment and the consequence on performance was studied. The performance of the anomaly detection was evaluated by measuring the following parameters(see chapter 3):

1. Accuracy
2. Memory Consumption
3. Execution Time

The empirical analysis had revealed that the existing approaches yielded lesser accuracy in detection and consumed higher computation complexity for processing the real-time anomaly detection. This had motivated this research to propose real-time anomaly detection based on big data technologies to address issues related to accuracy and performance problems.

The third objective was aimed at proposing and implementing a real-time anomaly detection based on big data technologies framework, for improving accuracy, minimizing the memory consumption and shorten the execution time. The proposed solution has been

named as Streaming Sliding Window LOF Clustering Coreset (SSWLOFCC) algorithms which has contributed to a deal in addressing the issue of lower accuracy. Furthermore, Spark streaming and Spark MLlib technologies helps to handle the issues of higher memory consumption and longer execution time. The framework comprises five major components: Flume, Kafka, Spark Streaming, Spark MLlib, and HBase which are flexible to receive the real-time data for processing. Thereby, the proposed solution processes the real-time data in-memory to detect the accurate anomaly and reduce memory consumption. Lastly, Dstreams shortens the execution time.

The fourth objective was aimed at evaluating the performance of the proposed RTADBDT framework and SSWLOFCC algorithms with various other existing techniques. The proposed and existing techniques were compared in terms of three parameters such as, accuracy, memory consumption, and execution time. Moreover, to validate the results, numerous internal, external, and statistical techniques were adopted. Eight evaluation techniques namely, Silhouette Index, Normalized Mutual Information, adjusted rand score, Calinski and Harabaz, confusion matrix, Precision, Recall, and F1 score were employed. The proposed SSWLOFCC algorithm has been implemented in a real cloud environment. Furthermore, three different datasets namely, DARPA, MACCDC, and DEFCON21 had been used to evaluate the quality of the proposed solutions. Finally, the performance of the proposed solution was carried out by comparing and evaluating SSWLOFCC algorithm with six different existing algorithms namely K Means, Isolation Forest, Spectral Clustering, HDBSCAN, Agglomerative Clustering, and Local Outlier Factor.

The comparative analysis had revealed that the performance of RTADBDT was much better than the existing algorithms in terms of accuracy, memory consumption, and execution time. Average results produced for three different datasets on the internal and external quality are as follows: (i) 94.33% on silhouette index, (ii) 95.83% on Calinski and

Harabaz, (iii) 96.52% on adjusted rand index, and (iv) 97.5% on normalized mutual info. Moreover, accuracy measurement produced by statistical techniques such as Precision, Recall, Fl had well predicated for the selected parameters. Likewise, Mathew's correlation coefficient and Kappa also were close to the prediction on the selected parameters. In addition to this the execution time is 13.3 seconds for the entire cycle, the latter processed within 3.13 seconds which is far less than existing algorithms. On the other hand, within the memory consumption measurement average for three different datasets of 194.3 MB and 14.61MB per second which also lower than the other algorithms on the framework.

7.2 Research Contributions

The contributions of this research work to the body of knowledge are detailed below:

7.2.1 Thematic Taxonomy

This research offers a thematic taxonomy, which categorizes various prospects of anomaly detection and real-time big data technologies. The categories are named as techniques, application, anomalies, modes, data, big data processing, and record. Furthermore, it emphasizes their similarities, strengths, and weaknesses in current solutions. Lastly, the critical research analysis reveals the major open research challenges in this domain of anomaly detection and real-time big data technologies (see figure 2.2).

7.2.2 Framework for Real-Time Anomaly Detection Based on Big Data Technologies

In this research experiment, a framework has been designed and implemented for real-time anomaly detection based on big data technologies (see figure 4.6). Among the many choices, the proposed novel framework is the best framework, which uses several big data technologies for anomaly detection. The framework comprises five major components: Flume, Kafka, Spark Streaming, Spark MLlib, and HBase which are flexible to receive the

real-time data for processing on anomaly detection. Furthermore, developed framework uses Spark API which are responsible for receiving and submitting jobs for processing among the nodes.

7.2.3 Proposed Algorithms

In this research, five different algorithms were developed and implemented in the proposed framework (see section 4.3 and 4.4). These algorithms help to resolve the issues of higher memory consumption, increased execution time and lower accuracy. The algorithms implemented in the proposed framework are as follows:

1. Algorithm 1: Flume Pipeline
2. Algorithm 2: Streaming Sliding Window LOF Clustering Coreset (SSWLOFCC)
3. Algorithm 3: To calculate RAM usage
4. Algorithm 4: To calculate memory consumption per second
5. Algorithm 5: To calculate execution time

7.2.4 Mathematical Model for Validation

This research had formulated a mathematical model to calculate the execution time of the Spark jobs. The expression obtains the individual Spark job execution time to calculate the entire cycle of the processing execution time of the algorithms(see section 4.4). The developed expression was validated with proposed and existing algorithms. Moreover, the experimental data were compared with three different datasets to prove the efficiency of the proposed solution.

7.2.5 Performance Evaluation of Proposed Solution

This research had performed an evaluation by critically analysing the accuracy of the proposed algorithms with five different existing algorithms on three different datasets.

The evaluation results had revealed that in terms of accuracy the proposed solution has outperformed all the five existing algorithms. Similarly, the memory consumption and execution time of the proposed solution were proved to be lower than the existing algorithms (see chapter 5 and 6 for detailed analysis and report).

7.2.6 Statistical and Evaluation Techniques

The research has contributed to the body of knowledge by designating and discussing various statistical methods, internal and external evaluation to determine the quality of results achieved for the proposed solutions (see section 5.4). In this regard Silhouette Index, Normalized Mutual adjusted rand score and Calinski and Harabaz had been employed to analyse the internal and external quality of the clusters. The statistical methods such as, confusion matrix, Precision, Recall, Kappa, Mathew's correlation coefficient, and F1 score had been utilized to evaluate the accuracy of the anomaly detection. The results of statistical evaluation methods had exhibited better significant results as against the existing solution.

7.3 Publications

In course of this research, the following papers have been produced:

- Published

  Habeeb, Riyaz Ahamed Ariyaluran, Fariza Nasaruddin, Abdullah Gani, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran. "Real-time big data processing for anomaly detection: A Survey." International Journal of Information Management (2018). (Impact Factor 4.5).

  Habeeb, Riyaz Ahamed Ariyaluran, Fariza Nasaruddin, Abdullah Gani, Mohamed Ahzam Amanullah, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran. "Clustering based real-time anomaly detection - A breakthrough in big

data technologies" Transactions on Emerging Telecommunications Technologies. (Impact Factor 1.7).

- Proceedings

  Fan Xiu Ming, **Habeeb, Riyaz Ahamed Ariyaluran**, Fariza Nasaruddin, Abdullah Gani, Real-time carbon dioxide monitoring based on IoT and Cloud technologies, ICSCA2019, International Conference proceedings series by ACM indexed by Ei compendex and scopus.

7.4 Significance and Limitations of the Proposed Solution

A number of researchers have proposed anomaly detection solutions in the past and off late more research interest have been invested in this domain owing to its significance. However, only few researchers have employed big data technologies for anomaly detection limited to batch processing and mainly focused on data storage, and those researchers did not focus on real-time anomaly detection. Real-time anomaly detection using big data technologies approaches are complicated due to various reasons, noteworthy mention is interoperability and other factors which affect the processing of the incoming data, and embedding algorithms into a real-time platform is also a complex process. In this research, a thorough critical analysis was conducted to modify the integrated big data technologies into the proposed framework. Most importantly, Spark MLlib which enable our proposed framework to run the algorithms into in-memory processing helps to process data in real-time. Furthermore, Apache Flume allow our proposed framework to pipeline the incoming data into Spark Streaming without interruption. Furthermore, Dstream and RDD in Spark Streaming ingest data to be processed in Spark MLlib. The proposed framework turns out to be competent, scalable and yields better accuracy in anomaly detection. The study had evaluated the proposed framework with three different parameters namely, accuracy, memory consumption, and execution time to prove the efficacy the

proposed framework in addressing the problems identified early in this research.

However, there are some limitations that have been highlighted as follows:

Non availability of latest dataset is the foremost limitation, as discussed earlier this research had been conducted with the validated datasets that are openly available, nevertheless, it is worth mentioning that this research would have been more sophisticated if it has employed most recent attack datasets like, Singapore hospital attack, and IoT attack launched in a major organization website in U.S, however, these datasets are not available for this research purpose due to their privacy and data protection policies. The next limitation is consumption of memory by the big data technologies while running in real-time. The datasets used in this research might have enabled it to validate other parameters such as, job execution delay, and usage of storage, etc. due to time constraints this research has not included those parameters as validation criteria.

7.5 Future Work

In this section the possible future guidelines that arise from this research work has been presented. Even though this research has not examined all the problems associated with RTADBDT various research efforts were performed to address the identified problems. This research has only stressed on the incorporation of real-time big data technologies and composite clustering algorithms for anomaly detection. To investigate the open research challenges, several future research directions have been recommended, wherein carrying advance research that can extend the analysis in these following areas:

1. This research has used the following three parameters to measure the proposed solution for experimental and concept validation:
    a) Accuracy
    b) Memory Consumption

c) Execution Time

However, the data collected by proposed solution is also capable of validating other parameters such as, job execution delay, and usage of storage. Future researchers can enhance their proposals by including these parameters for validation.

2. The solution proposed in this study was validated only with openly available datasets, however it was limited in using most recent attack datasets, hence future researchers are recommended herewith to attempt using most recent attack datasets to increase the validity of this proposed solution.
3. RTADBDT framework was implemented and validated in the cluster cloud environment. Due to limited resources, it was not feasible for this study to test the abilities of RTADBDT on multiple clusters and with parallel Kafka topics. Hence, future work shall be executed on multiple clusters.
4. To extend further, the proposed framework shall be implemented in real-time anomaly detection of Internet of things data sources or other network infrastructure like, gateway, network monitoring, and smart city applications.

REFERENCES

Abdullah, J., & Chanderan, N. (2017). Hierarchical density-based clustering of malware behaviour. *Journal of Telecommunication, Electronic and Computer Engineering JTEC*, *9*(2-10), 159-164.

Aburomman, A. A., & Reaz, M. B. I. (2017). A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems [Journal Article]. *Information Sciences*.

Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clustering–a decade review. *Information Systems*, *53*, 16–38.

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, *262*, 134-147.

Ahmed, E., Yaqoob, I., Hashem, I. A. T., Khan, I., Ahmed, A. I. A., Imran, M., & Vasilakos, A. V. (2017). The role of big data analytics in internet of things [Journal Article]. *Computer Networks*, *129*, 459-471.

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques [Journal Article]. *Journal of Network and Computer Applications*, *60*, 19-31.

Akoglu, L., Tong, H., & Koutra, D. (2015). Graph based anomaly detection and description: a survey [Journal Article]. *Data mining and knowledge discovery*, *29*(3), 626-688.

Ali, O., Shrestha, A., Soar, J., & Wamba, S. F. (2018). Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review [Journal Article]. *International Journal of Information Management*, *43*, 146-158.

Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., & Taha, K. (2015). Efficient machine learning for big data: A review. *Big Data Research*, *2*(3), 87–93.

Althubiti, S., Yuan, X., & Esterline, A. (2017). Analyzing http requests for web intrusion detection.

Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). Multi-level hybrid support

vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Systems with Applications*, *67*, 296-303.

Amini, A., Saboohi, H., Ying Wah, T., & Herawan, T. (2014). A fast density-based clustering algorithm for real-time internet of things stream. *The Scientific World Journal*, *2014*.

Amini, A., Wah, T. Y., & Saboohi, H. (2014). On density-based data streams clustering algorithms: A survey. *Journal of Computer Science and Technology*, *29*(1), 116–141.

Apache. (2019). *Spark streaming* [Web Page]. Retrieved from `https://spark.apache.org/streaming/`

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., & Buyya, R. (2015). Big data computing and clouds: Trends and future directions [Journal Article]. *Journal of Parallel and Distributed Computing*, *79*, 3-15.

Bang, J.-h., Cho, Y.-J., & Kang, K. (2017). Anomaly detection of network-initiated lte signaling traffic in wireless sensor and actuator networks based on a hidden semi-markov model [Journal Article]. *Computers & Security*, *65*, 108-120.

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.

Bhadani, A. K., & Jothimani, D. (2016). Big data: Challenges, opportunities, and realities. In *Effective big data management and opportunities for implementation* (pp. 1–24). IGI Global.

Bharill, N., Tiwari, A., & Malviya, A. (2016). Fuzzy based scalable clustering algorithms for handling big data using apache spark. *IEEE Transactions on Big Data*, *2*(4), 339–352.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, *16*(1), 303–336.

Birjali, M., Beni-Hssane, A., & Erritali, M. (2017). Analyzing social media through big

data using infosphere biginsights and apache flume. *Procedia computer science*, *113*, 280–285.

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection [Journal Article]. *IEEE Communications Surveys & Tutorials*, *18*(2), 1153-1176.

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, *3*(1), 1–27.

Camacho, J., Macia-Fernandez, G., Diaz-Verdejo, J., & Garcia-Teodoro, P. (2014). Tackling the big data 4 vs for anomaly detection [Conference Proceedings]. In *Computer communications workshops (infocom wkshps), 2014 ieee conference on* (p. 500-505). IEEE.

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k means clustering algorithm. *Expert systems with applications*, *40*(1), 200-210.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey [Journal Article]. *ACM computing surveys (CSUR)*, *41*(3), 15.

Chen, W.-Y., Song, Y., Bai, H., Lin, C.-J., & Chang, E. Y. (2011). Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, *33*(3), 568-586.

Cloud Strategy Partners, L. (2015). *Cloud based solutions for big data* (Tech. Rep.). IEEE Educational Activities and IEEE Cloud Computing. Retrieved from `http://ieeexplore.ieee.org/courses/details/EDP405?tag=1`

Cozar, J., Puerta, J. M., & Gamez, J. A. (2017). An application of dynamic bayesian networks to condition monitoring and fault prediction in a sensored system: a case study [Journal Article]. *INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE SYSTEMS*, *10*(1), 176-195.

Cui, B., & He, S. (2016). Anomaly detection model based on hadoop platform and weka interface. In *Innovative mobile and internet services in ubiquitous computing imis 2016 10th international conference on* (p. 84-89).

Dahiya, P., & Srivastava, D. K. (2018). Network intrusion detection in big dataset using spark. *Procedia computer science*, *132*, 253–262.

Ding, Z., & Fei, M. (2013). An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, *46*(20), 12-17.

Du, Y., Liu, J., Liu, F., & Chen, L. (2014). A real-time anomalies detection system based on streaming technology. In *2014 sixth international conference on intelligent human-machine systems and cybernetics* (Vol. 2, pp. 275–279).

Duggimpudi, M. B., Abbady, S., Chen, J., & Raghavan, V. V. (2017). Spatio-temporal outlier detection algorithms based on computing behavioral outlierness factor. *Data & Knowledge Engineering*.

Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning [Journal Article]. *Pattern Recognition*, *58*, 121-134.

Fan, J., Han, F., & Liu, H. (2014). Challenges of big data analysis. *National science review*, *1*(2), 293–314.

Fang, C., Liu, J., & Lei, Z. (2016). Fine-grained http web traffic analysis based on large-scale mobile datasets. *IEEE Access*, *4*, 4364–4373.

Farnaaz, N., & Jabbar, M. (2016). Random forest modeling for network intrusion detection system [Journal Article]. *Procedia Computer Science*, *89*, 213-217.

Feng, W., Zhang, Q., Hu, G., & Huang, J. X. (2014). Mining network data for intrusion detection through combining svms with ant colony networks [Journal Article]. *Future Generation Computer Systems*, *37*, 127-140.

Fernandes, G., Carvalho, L. F., Rodrigues, J. J., & Proença, M. L. (2016). Network anomaly detection using ip flows with principal component analysis and ant colony optimization [Journal Article]. *Journal of Network and Computer Applications*, *64*, 1-11.

Fernandes, G., Rodrigues, J. J., & Proença, M. L. (2015). Autonomous profile-based anomaly detection system using principal component analysis and flow analysis

[Journal Article]. *Applied Soft Computing*, *34*, 513-525.

Fernández, A., Carmona, C. J., del Jesus, M. J., & Herrera, F. (2016). A view on fuzzy systems for big data: progress and opportunities [Journal Article]. *International Journal of Computational Intelligence Systems*, *9*(sup1), 69-80.

Ficco, M., Pietrantuono, R., & Russo, S. (2018). Aging-related performance anomalies in the apache storm stream processing system. *Future Generation Computer Systems*, *86*, 975–994.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144.

García, L., Tomás, J., Parra, L., & Lloret, J. (2018). An m-health application for cerebral stroke detection and monitoring using cloud services [Journal Article]. *International Journal of Information Management*.

Gartner. (2018). *Gartner forecasts worldwide information security spending to exceed 124 billion dollar in 2019* (Web Page No. 15 August 2018). Retrieved from `https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to-exceed-124-billion-in-2019`

Genuer, R., Poggi, J.-M., Tuleau-Malot, C., & Villa-Vialaneix, N. (2017). Random forests for big data [Journal Article]. *Big Data Research*, *9*, 28-46.

Ger, M. (2017). *Autonomous vehicles silicon valley 2017 – the future of transportation will drive huge growth in data* (Tech. Rep.). Hortonworks. Retrieved from `https://hortonworks.com/blog/autonomous-vehicle-silicon-valley-2017-findings-future-transportation-will-drive-huge-growth-data/`

Gonçalves, D., Bota, J., & Correia, M. (2015). Big data analytics for detecting host misbehavior in large logs [Conference Proceedings]. In *Trustcom/bigdatase/ispa, 2015 ieee* (Vol. 1, p. 238-245). IEEE.

Grill, M., Pevný, T., & Rehak, M. (2017). Reducing false positives of network anomaly detection by local adaptive multivariate smoothing [Journal Article]. *Journal of Computer and System Sciences*, *83*(1), 43-57.

Guha, S., Mishra, N., Roy, G., & Schrijvers, O. (2016). Robust random cut forest based anomaly detection on streams. In *International conference on machine learning* (pp. 2712–2721).

Habeeb, R. A. A., Nasaruddin, F., Gani, A., Hashem, I. A. T., Ahmed, E., & Imran, M. (2018). Real-time big data processing for anomaly detection: A survey. *International Journal of Information Management*.

Hafsa, M., & Jemili, F. (2019). Comparative study between big data analysis techniques in intrusion detection. *Big Data and Cognitive Computing*, *3*(1), 1.

Hamamoto, A. H., Carvalho, L. F., Sampaio, L. D. H., Abrão, T., & Proença, M. L. (2017). Network anomaly detection system using genetic algorithm and fuzzy logic [Journal Article]. *Expert Systems with Applications*.

Hamamoto, A. H., Carvalho, L. F., Sampaio, L. D. H., Abrão, T., & Proença Jr, M. L. (2018). Network anomaly detection system using genetic algorithm and fuzzy logic [Journal Article]. *Expert Systems with Applications*, *92*, 390-402.

Hasani, Z. (2017). Robust anomaly detection algorithms for real-time big data: comparison of algorithms. In *Embedded computing (meco), 2017 6th mediterranean conference on* (p. 1-6).

Hashem, I. A. T., Chang, V., Anuar, N. B., Adewole, K., Yaqoob, I., Gani, A., ... Chiroma, H. (2016). The role of big data in smart city [Journal Article]. *International Journal of Information Management*, *36*(5), 748-758.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, *47*, 98–115.

Hayes, M. A., & Capretz, M. A. (2015). Contextual anomaly detection framework for big sensor data [Journal Article]. *Journal of Big Data*, *2*(1), 2.

helpnetsecurity. (2017). *Machine learning in cybersecurity will boost big data, intelligence, and analytics spending* (Web Page No. 21 March 2017). Retrieved from `https://www.helpnetsecurity.com/2017/01/31/machine-learning-cybersecurity/`

Hoffman, M., Steinley, D., & Brusco, M. J. (2015). A note on using the adjusted rand index for link prediction in networks. *Social Networks*, *42*, 72–79.

Jayanthi, M. D., Sumathi, G., & Sriperumbudur, S. (2016). A framework for real-time streaming analytics using machine learning approach. In *Proceedings of national conference on communication and informatics-2016.*

John Hunter, D. D. (2019). *matplotlib installation* [Web Page]. Retrieved from `https://matplotlib.org/l`

Jones, N. (2016). *Gartner identifies the top 10 internet of things technologies for 2017 and 2018* [Web Page]. Retrieved from `http://www.gartner.com/newsroom/id/3221818`

Juvonen, A., Sipola, T., & Hämäläinen, T. (2015). Online anomaly detection using dimensionality reduction techniques for http log analysis. *Computer Networks*, *91*, 46–56.

Kakavand, M., Mustapha, N., Mustapha, A., Abdullah, M. T., & Riahi, H. (2015). A survey of anomaly detection using data mining methods for hypertext transfer protocol web services [Journal Article]. *Journal of Computer Science*, *11*(1), 89.97.

Karami, A., & Guerrero-Zapata, M. (2015). A fuzzy anomaly detection system based on hybrid pso-kmeans algorithm in content-centric networks. *Neurocomputing*, *149*, 1253–1269.

Karim, A., Siddiqa, A., Safdar, Z., Razzaq, M., Gillani, S. A., Tahir, H., ... Imran, M. (2017). Big data management in participatory sensing: Issues, trends and future directions [Journal Article]. *Future Generation Computer Systems*.

Katal, A., Wazid, M., & Goudar, R. (2013). Big data: issues, challenges, tools and good practices. In *Contemporary computing (ic3), 2013 sixth international conference on* (pp. 404–409).

Kerner, S. M. (2016). *Cisco vni: 2.3 zettabytes of ip traffic in 2020* [Web Page]. Retrieved from `http://www.enterprisenetworkingplanet.com/netsp/cisco-vni-2.3-zettabytes-of-ip-traffic-in-2020.html`

Kleppmann, M. A., & Kreps, J. (2015). Kafka, samza and the unix philosophy of

distributed data.

Kokate, U., Deshpande, A., Mahalle, P., & Patil, P. (2018). Data stream clustering techniques, applications, and models: Comparative analysis and discussion. *Big Data and Cognitive Computing*, *2*(4), 32.

Koning, R., Buraglio, N., de Laat, C., & Grosso, P. (2018). Coreflow: Enriching bro security events using network traffic monitoring data. *Future Generation Computer Systems*, *79*, 235–242.

Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2011). An effective evaluation measure for clustering on evolving data streams. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp. 868–876).

Lai, Y., Liu, Z., Song, Z., Wang, Y., & Gao, Y. (2016). Anomaly detection in industrial autonomous decentralized system based on time series [Journal Article]. *Simulation Modelling Practice and Theory*, *65*, 57-71.

Landset, S., Khoshgoftaar, T. M., Richter, A. N., & Hasanin, T. (2015). A survey of open source tools for machine learning with big data in the hadoop ecosystem [Journal Article]. *Journal of Big Data*, *2*(1), 24.

Langone, R., Alzate, C., De Ketelaere, B., Vlasselaer, J., Meert, W., & Suykens, J. A. (2015). Ls-svm based spectral clustering and regression for predicting maintenance of industrial machines. *Engineering Applications of Artificial Intelligence*, *37*, 268–278.

Lavin, A., & Ahmad, S. (2015). Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark [Conference Proceedings]. In *Machine learning and applications (icmla), 2015 ieee 14th international conference on* (p. 38-44). IEEE.

Lin, D. (2013). *Big data analytics for network security monitoring* (Tech. Rep.). Pivotal. Retrieved from `https://content.pivotal.io/blog/big-data-analytics-for-network-security-monitoring`

Lin, W.-C., Ke, S.-W., & Tsai, C.-F. (2015). Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, *78*, 13–21.

Liu, S. (2015). *How do big data analytics enhance network security?* (Tech. Rep.). HUAWEI. Retrieved from http://www.forbes.com/sites/huawei/2015/02/24/how-do-big-data-analytics-enhance-network-security/

Liu, X., Iftikhar, N., & Xie, X. (2014). Survey of real-time processing systems for big data. In *Proceedings of the 18th international database engineering & applications symposium* (pp. 356–361).

Liu, X., & Nielsen, P. S. (2016). Regression-based online anomaly detection for smart grid data [Journal Article]. *arXiv preprint arXiv:1606.05781*.

Lobato, A., Lopez, M. A., & Duarte, O. (2016). An accurate threat detection system through real-time stream processing. *Grupo de Teleinformtica e Automao, Universidade Federal do Rio de Janeiro, Tech. Rep. GTA-16-08*.

Maarala, A. I., Rautiainen, M., Salmi, M., Pirttikangas, S., & Riekki, J. (2015). Low latency analytics for streaming traffic data with apache spark. In *2015 ieee international conference on big data (big data)* (pp. 2855–2858).

Maglaras, L. A., & Jiang, J. (2014). Intrusion detection in scada systems using machine learning techniques [Conference Proceedings]. In *Science and information conference (sai), 2014* (p. 626-631). IEEE.

Marir, N., Wang, H., Feng, G., Li, B., & Jia, M. (2018). Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access*, *6*, 59657–59671.

Mascaro, S., Nicholso, A. E., & Korb, K. B. (2014). Anomaly detection in vessel tracks using bayesian networks [Journal Article]. *International Journal of Approximate Reasoning*, *55*(1), 84-98.

McNeil, P., Shetty, S., Guntu, D., & Barve, G. (2016). Scredent scalable real-time anomalies detection and notification of targeted malware in mobile devices. *Procedia Computer Science*, *83*, 1219-1225.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... others (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, *17*(1), 1235–1241.

Mirsky, Y., Shabtai, A., Shapira, B., Elovici, Y., & Rokach, L. (2017). Anomaly detection for smartphone data streams [Journal Article]. *Pervasive and Mobile Computing*, *35*, 83-107.

Mnz, G., Li, S., & Carle, G. (2007). Traffic anomaly detection using k-means clustering. In *Giitg workshop mmbnet* (p. 13-14).

Muller, S., Lancrenon, J., Harpes, C., Le Traon, Y., Gombault, S., & Bonnin, J.-M. (2018). A training-resistant anomaly detection system. *Computers & Security*, *76*, 1–11.

Muniyandi, A. P., Rajeswari, R., & Rajaram, R. (2012). Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm [Journal Article]. *Procedia Engineering*, *30*, 174-182.

Nadler, B., & Galun, M. (2007). Fundamental limitations of spectral clustering. In *Advances in neural information processing systems* (pp. 1017–1024).

Neumeyer, L., Robbins, B., Nair, A., & Kesari, A. (2010). S4: Distributed stream computing platform. In *Data mining workshops (icdmw), 2010 ieee international conference on* (pp. 170–177).

Nian, K., Zhang, H., Tayal, A., Coleman, T., & Li, Y. (2016). Auto insurance fraud detection using unsupervised spectral ranking for anomaly. *The Journal of Finance and Data Science*, *2*(1), 58-75.

Noghabi, S. A., Paramasivam, K., Pan, Y., Ramesh, N., Bringhurst, J., Gupta, I., & Campbell, R. H. (2017). Samza: stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, *10*(12), 1634–1645.

Ohio, U. (2017). *U.s. federal cybersecurity market forecast 2017-2022* (Web Page No. 21 March 2017). Retrieved from `https://www.marketresearchmedia.com/?p=206`.

Ošep, A., Voigtlaender, P., Luiten, J., Breuers, S., & Leibe, B. (2017). Large-scale object discovery and detector adaptation from unlabeled video. *arXiv preprint arXiv:1712.08832*.

P.R, Z. (2019). *Zeek broids* [Web Page]. Retrieved from `https://www.bro.org/sphinx/intro/index.html`

Promod, K., & Jacob, B. (2016). Mining a ubiquitous time and attendance schema using random forests for intrusion detection [Journal Article]. *Procedia Technology*, *24*, 1226-1231.

Prosak, A., Gangopadhyay, A., & Garg, H. (2019). *A new machine learning approach for anomaly detection using metadata for model training* (Tech. Rep.). EasyChair.

Puggini, L., & McLoone, S. (2018). An enhanced variable selection and isolation forest based methodology for anomaly detection with oes data. *Engineering Applications of Artificial Intelligence*, *67*, 126-135.

Qadah, E., Mock, M., Alevizos, E., & Fuchs, G. (2018). A distributed online learning approach for pattern prediction over movement event streams with apache flink. In *Edbt/icdt workshops.*

Raguseo, E. (2018). Big data technologies: An empirical investigation on their adoption, benefits and risks for companies. *International Journal of Information Management*, *38*(1), 187–195.

Ramamoorthi, A., Subbulakshmi, T., & Shalinie, S. M. (2011). Real time detection and classification of ddos attacks using enhanced svm with string kernels [Conference Proceedings]. In *Recent trends in information technology (icrtit), 2011 international conference on* (p. 91-96). IEEE.

Ranjan, R. (2014). Streaming big data processing in datacenter clouds. *IEEE Cloud Computing*, *1*(1), 78–83.

Rathore, M. M., Ahmad, A., & Paul, A. (2016). Real time intrusion detection system for ultra-high-speed big data environments. *The Journal of Supercomputing*, *72*(9), 3489–3510.

Rettig, L., Khayati, M., CudrMauroux, P., & Pirkowski, M. (2015). Online anomaly detection over big data streams. In *2015 ieee international conference on big data* (p. 1113-1122).

Rivetti, N., Busnel, Y., & Gal, A. (2017). Flinkman: Anomaly detection in manufacturing equipment with apache flink: Grand challenge. In *Proceedings of the 11th acm international conference on distributed and event-based systems* (pp. 274–279).

Sahasrabuddhe, A., Naikade, S., Ramaswamy, A., Sadliwala, B., & Futane, P. (2017). Survey on intrusion detection system using data mining techniques. *International Research Journal of Engineering and Technology*.

Sarker, R. A., Elsayed, S. M., & Ray, T. (2014). Differential evolution with dynamic parameters selection for optimization problems. *IEEE Trans. Evolutionary Computation*, *18*(5), 689–707.

She, C., Wen, W., Lin, Z., & Zheng, K. (2017). Application-layer ddos detection based on a one-class support vector machine [Journal Article]. *International Journal of Network Security & Its Applications (IJNSA)*.

Shiravi, H., Shiravi, A., & Ghorbani, A. A. (2012). A survey of visualization systems for network security [Journal Article]. *IEEE Transactions on visualization and computer graphics*, *18*(8), 1313-1329.

Shu, B., Chen, H., & Sun, M. (2017). Dynamic load balancing and channel strategy for apache flume collecting real-time data stream. In *2017 ieee international symposium on parallel and distributed processing with applications and 2017 ieee international conference on ubiquitous computing and communications (ispa/iucc)* (pp. 542–549).

Singh, K., Guntuku, S. C., Thakur, A., & Hota, C. (2014). Big data analytics framework for peer to peer botnet detection using random forests. *Information Sciences*, *278*, 488-497.

Software, P. (2019). *Python installation* [Web Page]. Retrieved from `https://www.python.org/about/`

Solaimani, M., Iftekhar, M., Khan, L., Thuraisingham, B., & Ingram, J. B. (2014). Spark-based anomaly detection over multi-source vmware performance data in real-time. In *Computational intelligence in cyber security (cics), 2014 ieee symposium on* (pp. 1–8).

Solaimani, M., Khan, L., & Thuraisingham, B. (2014). Real-time anomaly detection over vmware performance data using storm. In *2014 ieee international conference on information reuse and integration* (p. 458-465).

Son, J. G., Kang, J.-W., An, J.-H., Ahn, H.-J., Chun, H.-J., & Kim, J.-G. (2016). Parallel job processing technique for real-time big-data processing framework. In *Proceedings*

of the international conference on research in adaptive and convergent systems (pp. 226–229).

Srikanth, B., & Reddy, V. K. (2016). Efficiency of stream processing engines for processing bigdata streams. *Indian Journal of Science and Technology*, *9*(14).

Stojanovic, L., Dinic, M., Stojanovic, N., & Stojadinovic, A. (2016). Big-data-driven anomaly detection in industry 4.0: An approach and a case study. In *Big data (big data), 2016 ieee international conference on* (p. 1647-1652).

Stripling, E., Baesens, B., Chizi, B., & vanden Broucke, S. (2018). Isolation-based conditional anomaly detection on mixed-attribute data to uncover workers compensation fraud. *Decision Support Systems*.

Su, M.-Y. (2011a). Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers [Journal Article]. *Expert Systems with Applications*, *38*(4), 3492-3498.

Su, M.-Y. (2011b). Using clustering to improve the knn-based classifiers for online anomaly network traffic identification. *Journal of Network and Computer Applications*, *34*(2), 722–730.

Suthaharan, S. (2014). Big data classification: Problems and challenges in network intrusion prediction with machine learning [Journal Article]. *ACM SIGMETRICS Performance Evaluation Review*, *41*(4), 70-73.

Symantec. (2016). *Anomaly detection for automotive- proactive security analytics built for embedded systems* (Web Page No. 20/04/2017). Retrieved from `https://www.symantec.com/content/dam/symantec/docs/data-sheets/anomaly-detection-for-automotive-en.pdf`

Ta, V.-D., Liu, C.-M., & Nkabinde, G. W. (2016). Big data stream computing in healthcare real-time analytics. In *2016 ieee international conference on cloud computing and big data analysis (icccbda)* (pp. 37–42).

Tang, R., & Fong, S. (2018). Clustering big iot data by metaheuristic optimized mini-batch and parallel partition-based dgc in hadoop. *Future Generation Computer Systems*.

Wang, C., Zhao, Z., Gong, L., Zhu, L., Liu, Z., & Cheng, X. (2018). A distributed anomaly

detection system for in-vehicle network using htm. *IEEE ACCESS*, *6*, 9091–9098.

Wasikowski, M., & Chen, X.-w. (2010). Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1388–1400.

Wauters, M., & Vanhoucke, M. (2017). A nearest neighbour extension to project duration forecasting with artificial intelligence [Journal Article]. *European Journal of Operational Research*, *259*(3), 1097-1111.

Wiatr, R., Słota, R., & Kitowski, J. (2018). Optimising kafka for stream processing in latency sensitive systems. *Procedia Computer Science*, *136*, 99–108.

Woolf, N. (2016). *Ddos attack that disrupted internet was largest of its kind in history, experts say* [Web Page]. Retrieved from `https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet`

Xhafa, F., Naranjo, V., & Caballé, S. (2015). Processing and analytics of big data streams with yahoo! s4. In *Advanced information networking and applications (aina), 2015 ieee 29th international conference on* (pp. 263–270).

Xie, S., & Chen, Z. (2017). Anomaly detection and redundancy elimination of big sensor data in internet of things [Journal Article]. *arXiv preprint arXiv:1703.03225*.

Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, *2*(2), 165–193.

Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B., & Vasilakos, A. V. (2016). Big data: From beginning to future [Journal Article]. *International Journal of Information Management*, *36*(6), 1231-1247.

Yasumoto, K., Yamaguchi, H., & Shigeno, H. (2016). Survey of real-time processing technologies of iot data streams [Journal Article]. *Journal of Information Processing*, *24*(2), 195-202.

Yin, C., Zhang, S., & Kim, K.-j. (2017). Mobile anomaly detection based on improved self-organizing maps [Journal Article]. *Mobile Information Systems*, *2017*.

Yin, S., Zhu, X., & Jing, C. (2014). Fault detection based on a robust one class support vector machine [Journal Article]. *Neurocomputing*, *145*, 263-268.

Yuan, D., Jin, J., Grundy, J., & Yang, Y. (2015). A framework for convergence of cloud services and internet of things. In *Computer supported cooperative work in design (cscwd), 2015 ieee 19th international conference on* (pp. 349–354).

Zhao, S., Chandrashekar, M., Lee, Y., & Medhi, D. (2015). Real-time network anomaly detection system using machine learning. In *Design of reliable communication networks (drcn), 2015 11th international conference on the* (pp. 267–270).

Zhao, Y., Ni, Q., & Zhou, R. (2017). What factors influence the mobile health service adoption? a meta-analysis and the moderating role of age [Journal Article]. *International Journal of Information Management*.

Zhou, J., Lazarevic, A., Hsu, K.-W., Srivastava, J., Fu, Y., & Wu, Y. (2010). Unsupervised learning based distributed detection of global anomalies. *International Journal of Information Technology & Decision Making*, *9*(06), 935–957.

Zhou, L., Pan, S., Wang, J., & Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges [Journal Article]. *Neurocomputing*, *237*, 350-361.

Zhou, Y., Yan, S., & Huang, T. S. (2007). Detecting anomaly in videos from trajectory similarity analysis. In *Multimedia and expo, 2007 ieee international conference on* (pp. 1087–1090).

Zolotukhin, M., Hämäläinen, T., Kokkonen, T., & Siltanen, J. (2014). Analysis of http requests for anomaly detection of web attacks. In *2014 ieee 12th international conference on dependable, autonomic and secure computing* (pp. 406–411).