Republic of Iraq

Ministry of Higher Education and Scientific Research

University of Babylon

College of Information Technology

Software Department

Improve Fog Services Performance Using Smart Broker and Controller

A Dissertation

Submitted to the Council of the College of Information Technology, University of Babylon in Partial Fulfillment of the Requirements for the Doctor of Philosophy Degree in Information Technology/ Software

By

Wial Abbas Hanon Ali

Supervised by:

Asst. Prof. Dr Mahdi Abed Salman

2023 A.C. 1445 A.H.

بسم الله الرحمن الرحيم

وَأَن لَّيْسَ لِلْإِنسَانِ إِلَّا مَا سَعَى ﴿٣٩﴾ وَأَنَّ سَعْيَهُ

سَوْفَ يُرَى ﴿٤٠﴾ ثُمَّ يُجْزَاهُ الْجَزَاءَ الْأَوْفَى ﴿٤١﴾

صدق الله العلي العظيم

سورة النجم

Supervisor Certification

I certify that this dissertation entitled " Improve Fog Services Performance Using Smart Broker and Controller " was prepared under my supervision at the Department of Software / College of Information Technology / University of Babylon, by Wial Abbas Hanon as a partial fulfillment of the requirements of the degree of Ph.D. in Information Technology / Software.

Signature:

Name: Dr. Mahdi Abed Salman

Title: Asst. Prof.

Date: / / 2023

The Head of the Department Certification

In the view of available recommendations, I forward the dissertation entitled "" Improve Fog Services Performance Using Smart Broker and Controller " for debate by the examination committee.

Signature:

Name: Dr. Sura Zaki Alrashid

Title: Asst. Prof.

Date: / /2023

Declaration

I hereby declare that this Dissertation, submitted to University of Babylon in partial fulfillment of requirement for the degree of Ph.D. in Information Technology \ Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source are appropriately cited in the references.

Signature:

Name: Wial Abbas Hanon

Date: / /2023

Dedications

To the savior of mankind, Al-Imam Al-Mehdi (peace be upon him), whowill fill the earth with justice and equity, after it was filled with injusticeand oppression.

To the soul of my father. May Allahhave mercy on you, my father.

To my mother. You have always been a source of kindness and compassion; I grew up on the shoulders of your patience. I got this certificate thanks to your prayers. God bless you for us.

To my wife and children who are the secret of my happiness in life. I thankyou for your understanding of my shortcomings during the study period, and I thank your unlimited support, as I live for you.

To my sisters and brother who are the source of tenderness for me.

To every relative or friend who encouraged me and prayed for me from hisheart. All respect to you

Acknowledgements

All praise be to ALLAH Almighty who enabled me to complete this task successfully willingness to undertake this work and the opportunity to contribute a drop in the sea of knowledge and my utmost respect to his last Prophet Mohammad PBUH and to the hero of Islam born in the heart of Kaaba Amir-Al- Mo'mineen Al-Imam Ali Ibn Abi Talib.

All thanks, appreciation, and respect to my adviser Dr. Mahdi Abed Salman, for his invaluable guidance, supervision, and untiring efforts during the course of this work.

I want to say thank you for all the staff of IT College / Babylon University and for the professors who presented the fantastic courses in this PhD program.

Many thanks for the encouragement of my colleagues in Computer Center / Babylon University, and the staff of the College of Information Technology / Software Department.

Last but not least, but most importantly, I would like to thank all the kind, helpful, and lovely people who helped me directly or indirectly to complete this work and apologize to them for not being able to mention them by name here, but they are in my heart.

Wial Abbas Hanon

Abstract

Recently, there has been a revolution in the use of Internet of Thing (IoT) devices, which is expanding into all fields related to human daily life. As in health care, quality of production, weather forecasting, smart cities, and other fields of life. The IoT is one of the important resources that generate massive amounts of data in efficient time. Data is heterogeneous and streaming continuously and widely distributed and needs effective techniques for resource management in storage, processing, and analysis. The data stream generated may contain missing values and anomalies, causing low accuracy and reliability in a system and deteriorating the system's performance. Most IoT devices lack the resources for processing data streams locally. Cloud Computing (CC) and the edge (CC gateway) can provide promising solutions, but they have drawbacks. Such as scalability and remoteness from data resources, heterogeneous data sources, a lack of resources for large processors, and low bandwidth. Subsequently, they lead to problems in security, high latency, low response time, dropping of data, and network overload

This dissertation aims to build adaptive architecture by combining the Smart Controller (SC) and services with the MQTT broker in the Fog Computing (FC). By benefiting from the advantages of the FC of the low latency, high response time, and high throughput the system performance is improved and the reliability is increased. In addition, integrates machine learning models for predicting the best number of services, and proves diagnosis the breast cancer.

The SC makes decisions automatically about adding or removing new services. Moreover, the clean data is produced from functions in the services of the anomalies and missing values.

Experiments have been conducted on three datasets: Smart-Homes-Temperature-time-Series-Forecasting (Smart-homes), Intel Berkeley Research Lab Sensor Data (Intel Berkeley), and Diagnostic Breast Cancer medical dataset (DBC) are used. Accuracy has been used to measure the quality of proposed models (One-Class-Support vector machine), whereby the accuracy of anomaly detection reached 99.44% and 99.42% for the Intel Berkeley and Smart-homes datasets, respectively. The results of the proposed model that integrate PCA and KNN Number of Services Prediction (PKNSP) model reached 0.0043 using the mean square error.

The Prediction of Breast Cancer (BCP) model achieved an accuracy, precision, and F1- Score of 99.12%, 99.13%, and 99.12% respectively. The reduction of the data size by applying BCP model achieved a reduction of 68%. The findings demonstrated the effectiveness of the adaptive architecture by deploying a set of services and SC with an MQTT broker. The improvement in average latency is 91.65%, the response time is 97.34%, the reduction in data loss is 84.49%, and the throughput is 7.07 j/s in the proposed architecture.

Table of Contents

Chapter One: Background and Overview

Chapter Two: Theoretical Background

Chapter Three: Methodology and Proposed System

Chapter Four: Results and Discussions

Chapter Five: Conclusions and Future Directions

List of Tables

List of Figures

List of algorithms

List of Abbreviations

Abbreviation	Meaning
ANN	Artificial Neural Network
BCP	Brest Censer Predict Model
CC	Cloud Computing
CNN	Convolutional neural network
DT	Decision Tree
DBC	Diagnostic Breast Cancer Dataset
FC	Fog Computing
IoT	Internet of Things
ISF	Isolation Forest
KNN	K-Nearest Neighbors
KNNR	K-Nearest Neighbors Regression
KP	Number of Principal Components
ML	Machine Learning
MSE	Mean Square Error
MQTT	Message Queue Telemetry Transport
OCSVM	One-class Support Vector Machines
PCA	Principal Component Analysis
PKNSP	PCA and KNN for Number of Services Prediction Model
QoS	Quality of Services
RF	Random Forest
SB	Smart Broker
SEB	Smart Edge Broker
SC	Smart Controller
SVM	Support Vector Machines

Chapter One

Background and Overview

Chapter One: Background and Overview

1.1. Introduction

The Internet of Things (IoT) is a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities use intelligent interfaces, often communicate data related with users and their environments [1]. The IoT has become a crucial part of our everyday lives. The IoT generates a massive data stream that is continually transmits to (the gateways or fog nodes) and then to the cloud for processing, analysis, and storage [2]. The massive stream requires rapid preprocessing, a high response time, and low latency. In addition, it may contain missing values, and anomalies, or it can lead to increase an overload on the system services [3].

IoT applications are specific use cases or implementations of IoT technology to solve particular problems or provide specific functionalities [1]. These applications exploit the data produced by Internet of Things devices to improve user experiences, streamline workflows, and provide services. Smart cities, healthcare monitoring, industrial automation, smart homes, and agricultural monitoring are a few examples of IoT applications [4]. The IoT applications that use stream processing have high latency requirements in order to be functioned correctly [5].

IoT devices are real-world "things" or physical objects that have been fitted with sensors, actuators, and communication features so they can gather, transfer, and receive data via the Internet. The generation of data by IoT devices is essential for apps to exploit in order to get insights and take action [1], [4].

An IoT system is a network of networked devices, also called "things," that share data and communicate with one another via the Internet. These devices could be commonplace items that have sensors, software, and other technology integrated into them for data collection and exchange. To enable intelligent, automated, and frequently remote monitoring and control of physical things, IoT systems generally integrate hardware, software, communication protocols, and data analytics[4]. IoT system performance may be deteriorated and stop, and data could be lost [6]. With the slow data stream processing in the IoT services, the system could be unstable or degraded; thus, the data would be dropped, reducing the system's reliability and Quality of Services (QoS) [7]. Therefore, the big challenge is how to efficiently fast data processing locally in efficient time with decreased overload on the system services[8].

The "IoT services" refers to the various services and capabilities that are provided in the context of the IoT. IoT services play a crucial role in facilitating communication, data management, and functionality in IoT environments [9].

Cloud computing (CC) is a model for enabling ubiquitous, appropriate, on-demand network access to a shared grouping of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with the least management effort or service provider interaction [10]. CC is considered a promising solution for IoT data through the process of providing resources, storage, and development at the same time. Its use has led to changes in the way live, work, and study since it is announced around the year 2005 [3]. However, CC suffers has limited scalability, and the distance of data centers from IoT devices. Thus, the inability to meet the requirements of applications that need low latency and rapid decision-making with the need to compute large amounts of data transferred to the data center [11].

Therefore, it is needed to find technologies that are compatible with the requirements of IoT applications such as Fog Computing (FC).

FC is a decentralized environment, an extension of CC, and, at the same time, is close to the data source (the IoT platform)[12]. **An IoT platform** is a collection of services and technologies that make it easier to create, implement, and maintain IoT devices and applications. By providing developers and enterprises with a centralized environment, IoT platforms seek to reduce the complexity involved in developing and expanding IoT solutions[9] .

The gateway is a device that acts as an interface between various communication protocols or networks. Interoperability is made possible via gateways, which allow data to move between systems or devices that employ various communication protocols [4].

In the context of fog computing, a **fog node** is a server or computing device that is located closer to the devices and sensors that are producing data, near the network's edge. Fog nodes enable networking, processing, and storage capabilities close to the devices, resulting in faster reaction times and more effective data processing[13]. FC is used to enhance data analysis and processing by working as a gateway for collecting data and making decisions in efficient time. Moreover, it can reduce latency and high response times, facilitate rapid processing, and bring computations closer to the data generation source for systems that need fast response [14].

Undoubtedly, using a broker (publish/subscribe) is a straightforward and quick method to handle the massive data produced by IoT devices and process it close to the data source [15].

This is suitable for multiple and time-sensitive IoT systems, the publisher is unaware of the subscriber, and system failure is avoided if one of the subscribers is not connected to the Internet [16]. However, there are a variety of uses for the broker, such as: i) use a broker for routing the data. ii) dynamic data processing in fog nodes iii) resource management, energy, and content filtering. v) scheduling for load balancing and iv) determining the optimal data destination [17].

This dissertation is going to employ the Message Queue Telemetry Transport (MQTT) broker combined with a Smart Controller (SC) that automatically controls the system's performance by enhancing the fast preprocessing of the data stream for IoT devices. In addition, employing Machine Learning (ML) algorithms inside the components (services and smart controller) of the system to increase accuracy and reliability. Moreover, investment in the FC environment provides the resources required for the applications of IoT and reduces latency, and improves service quality [18]. The FC paradigm have advantages, including a reduction in communication latency between a client's request and the cloud service provider's response time, a decrease in network congestion, the ability to compute at the fog nodes for devices, and a reduction in computational costs at the cloud server [19].

1.2. Problem Statement

The massive data generated by IoT in efficient time leads to an overload on the system's services and might lead to performance degradation in terms of high latency, low throughput, low response time, and increased data loss. The dynamic scalability of IoT devices, massive amounts of data, and a variety of required processes for the data cannot be managed according to predetermined plans. Thus, it is highly required to use distributed and scalable solutions using an adaptive

architecture that has the ability to process different data streams in efficient time . In addition, an improvement in overall system performance is represented by enhancing the QoS measures in efficient time that utilize available system resources.

1.3. Aim of the Study

The main aim of this dissertation is to improve the IoT system performance by automatically making decisions for adding or removing new services for the architecture and to achieve this, the following objectives will be considered.

1. Constructing an adaptive architecture by benefiting from the power of a FC environment and message MQTT broker based on a dynamic model.
2. Building a set of services based on ML algorithms for increasing velocity of data preprocessing.
3. Building SC for making decisions dynamically in efficient time by developing a model integrating several MLs which is used for predicting the best number of services.

1.4. Limitation and Challenges of Study

The proposed architecture is implemented in cases with a huge data stream and efficient time data generation, such as in healthcare, smart cities, and weather systems. However, this need for urgent decision-making automatically in efficient time may not be feasible in situations where the data flow rate is low and there is no requirement for immediate responses.

The main challenge in processing the data stream is how it is possible to quickly receive and process this data, which often has low latency and high response time requirements to extract insights and knowledge from raw data. In

addition to reducing data loss to increase reliability [20]. The variety of data from IoT devices presents a major challenge to the design of scalable IoT applications. Because it requires flexible systems in terms of adding modern IoT devices while taking into account the operation of devices of older models [21]. The system has one opportunity to process the data stream and extracts information from IoT-generated massive data, as in anomaly detection [22], and missing values [23]. Moreover, the system has priorities in processing and require a high response rate and low latency.

1.5. Dissertation Contributions

This dissertation has achieved the following contributions in comparison to previous work:

1. Proposing an adaptive architecture integrating the SC with an Smart broker (IB) to reduce the overload of the services in the FC node and ensuring interaction between IoT devices and services by distributing the data stream on a set of service.
2. Detecting missing values and then anomalies from the data stream in the services, by using ML algorithms such as One-class-SVM and then correction.
3. Making decisions dynamically to add (spawn) or remove (kill) service automatically in the system, according to the outcome of the SC Algorithm by collecting information from services in a screen table.
4. Deploying initial service dynamically based on the data topic name that does not belong to an active set of currently deploying services.
5. Improving overall system performance, QoS in terms of latency and throughput, and the system's reliability and stability by predicting the best

number of services based on the system state.

1.6. Related Works

Most researchers in the field of IoT systems and FC investigate three sides, approved architecture, system performance, and implementation tools.

As for the study in [7] , the authors used a framework based on integrating fog with deep learning in the analysis of heart diseases. It provides health care as a service to users. It is used in applications that are sensitive to response time, high scalability, and load-balancing division of tasks. The authors used the broker for load balancing between services, receiving data from the gateway, determining which service is available, and sending data to the cloud if there is no service available. In healthcare, the task requests and, or input data from the edge are received by the broker component. The purpose of the request input module is resource management and task initiation is carried out on fog broker nodes, where the broker's location is a node inside the FC and within the framework (FogBus). The broker is made up of a security manager, data manager, resource manager, and cloud integrator. Due to the broker's position (within the FC environment or FogBus-based edge environment), it will receive job requests and/or input data from the gateway devices (mobile phones, laptops, and tablets), which serve as a fog device to gather sensor data from various sensors and forward this data to the broker nodes/workers for additional processing. Researchers did not dispense with the cloud, in addition to running a specific set of services to detect heart diseases, using a protocol HTTP to transfer data between IoT devices and services.

The study in [24] has used a Smart Edge Broker (SEB), located at the edge of the network between IoT devices (sensors) and services and datacenter all over the city, the purpose is to receive all the data from the sensors and analyze the request for data and how it is directed to the appropriate, SEB provides intelligent data

routing functions at the edge of smart homes and services, thus transmitting data be more efficient. In smart cities, the distance between services and sensors is more than several kilometers, so it is normal to send and receive data for a long time, and thus the QoS decreases [15] and the larger the range to distribute devices, the more sensors to cover the range, and the greater the burden on the data center for data collection and transmission.

The study in [15] has used FC, which extends the ability of an IoT gateway to handle a group of fog nodes and choose the appropriate fog node for the current data requirements that are within the QoS (latency and throughput). The challenge is in responding fast to key events that affect moving data streams before they are stored in the cloud. The solution uses fog, with a broker which is responsible for scheduling different applications and determining the available resources according to the current state of the system, in other words, determining the fog service that is suitable for applications according to the current state of the resources available to the system.

The authors in [25] have used northbound broker ,the location is in the added edge layer. The purpose of the broker is to manage the lifecycle of advanced IoT services deploy within an added edge layer (the position) of the network between the IoT and the cloud to support processing, communication and storage services near the end user, increasing performance, mobility, security, privacy, reducing the volume of exchanged data, ensuring low latency to access data, and energy saving. Testing of performance and functional results are used to evaluate this method.

The study in [26] has used a lightweight IoT broker to manage a portion of context data, record data, and the dynamic coordination of data processing tasks. Many city services require very low latency and fast response time with significantly high storage and configuration costs if all the data sensors are shared

to the cloud. The fog computing [27] fits with the geographically distributed nature of smart city infrastructure and is mainly based on improving the deployment of tasks across the distributed edges in terms of providing bandwidth and reducing latency[17].

The study in [2] has used a distributed mobile broker based on the publish-subscribe model built using MQTT brokers. The role of the broker is to communicate the information on the appropriate topic to the appropriate subscriber. Initially, the publishers inform the broker of the amount and priority of the data. The priority is thus determined based on the data request. Depending on the value reported, the broker assigns a weight to the publishers, and the broker selectively chooses the data-collection area, positions, and collects data from nearby nodes. In this method, publishers who have massive and important data are collected in the list and then the broker can find them. Thus, The amount of time it takes the mobile broker to gather all of the nodes' available data is the data acquisition response time.

The study in [16] has used the message brokers (RabbitMQ and Apache Kafka) inside the fog node to evaluate the performance. The message broker can also use a publish/subscribe communication model which is suitable for IoT systems that often consist of many connected devices that are published messages over the broker. The performance of each message of the broker is determined based on throughput and response time [17].

The study in [21] has used a dynamic solution to deliver large-scale IoT data with low latency and high scalability by employing the cloud and the fog to address the problem of the huge amounts of data produced by IoT devices which needs to be transported, stored, and analyzed in appropriate places and be close to the source of its generation before sending it to the cloud. IoT devices require low

utilization of system resources, high usability, and high traffic to pass large and continuous data. At the network's edge, the hierarchical Publishing/Subscription broker paradigm is employed to support low-latency data delivery. Data latency will be decreased by enabling intermediaries to exchange data with one another.

The study in [28] has used the integrated publish/subscribe and push/pull methods of a cloud-based and scalable IoT framework are provided for connecting IoT devices and processing efficient time data. The suggested framework transfers data from the device to the cloud using a push-pull mechanism and a publish-subscribe message broker. IoT devices transmit data through a channel that cloud service providers also use. A push-pull method is used to deploy the system and transfer data from the medium to the cloud. The broker makes the necessary computations to choose the cloud service provider in this process. As a result, the gadget is under less load. The system's latency is decreased by the concurrent processing of all calculations. Any number of devices, brokers, and cloud service providers can be accommodated by the system.

The study in [29] has used One-class Support Vector Machines (OCSVM), which can adapt to complicated nonlinear boundaries between normal and novel data, are one of the most recent methods for novelty detection (or anomaly detection) in ML. Due to the variety of IoT devices, traffic patterns, and sorts of anomalies that can appear in such contexts, OCSVM presents a compelling use case for novelty detection in linked building infrastructure, smart homes, and smart cities. As a result, a large portion of earlier research utilized OCSVM to address the novelty of the IoT.

The authors in [30] have used more IoT devices for building control systems, are connected to the internet, both this connected infrastructure and the network itself are vulnerable to new threats and weaknesses. A typical method of network

defense is anomaly detection, which looks for unexpected activity based on the characteristics of network data that can be seen. Due to its flexibility, notably, its capacity to recognize a broad range of nonlinear boundaries dividing classes of data. OCSVM is one of the most cutting-edge methods for anomaly identification in ML. Such adaptability is obviously suitable in situations involving IoT devices and apps, which by their very nature exhibit complexity due to the enormous heterogeneity of devices.

The authors in [31] have used K-nearest neighbors (KNN) regression which is a supervised learning algorithm that can be used for regression tasks. It predicts the target value of a new sample by considering the values of its k-nearest neighbors in the training dataset. The KNN does not require prior knowledge of the distribution or parameters of the underlying dataset because feature distributions are not always known in advance when working with complex data structures, and this makes it a potent tool. This approach turns into useful in situations where there is a dearth of pertinent data regarding a specific problem domain, such as IoT applications.

Utilizing a variety of coping mechanisms, physicians are now able to diagnose breast cancer in women. Numerous data science approaches, in addition to new technologies, make it easier to gather and analyses cancer-related data in order to forecast this potentially fatal condition. The application of ML techniques to the treatment of cancer computationally has proved fruitful. Automatic learning systems, for instance, have been demonstrated in research [32] to boost diagnosis accuracy by 79.97%. On the other hand, ML achieved 91.1% accurate predictions.

The authors in [33] have used a breast cancer prediction model built using an improved deep learning methodology. The authors have provided this improved deep Recurrent Neural Network (RNN) model based on RNN and Keras-Tuner

Optimization approach for the early detection of breast cancer. An input layer, five hidden layers, five dropout layers, and an output layer make up the optimized deep RNN model.

The authors in [34] have used demonstrated how to diagnose breast cancer utilizing an SVM approach and a few selected capabilities. The performance of the model is validated using the DBC dataset. The results of the experiment showed that, in comparison to other ML models, the suggested SVM had the greatest classification accuracy, up to 98.51 percent. Models for predicting breast cancer have been created using ML techniques including RF, LR, and SVM.SVM is more sensitive than other models, with scores ranging from 82% to 88%.

The study in [35] has presented a model of decisional trees for breast cancer detection. The Gini index is employed by the decision tree to establish the qualities' priority levels. The suggested diagnostic method outperformed existing models like adaptive boosting (AdaBoost), SVM, KNN, NB, Artificial Neural Network (ANN), and others with a precision rate of 90.52%.

In [36], the authors have proposed a breast cancer diagnosis and detection model using MLP and Convolutional Neural Network (CNN). The model's effectiveness is measured by how well it can spot cancer in breast cells. CNN is more accurate than MLP by 98.37%.

The study in [37] has used the Feature selection and dimensionality reduction for the DBC dataset using PCA. A Decision Tree (DT) and K-nearest neighbor (K-NN) are trained and tested using the features that are chosen. The K-NN classifier outperformed the DT in all measures, with 95.61% accuracy, 95.95% sensitivity, and a Mathews Correlation Coefficient (MCC) of 90.44%. The DT and K-NN models didn't go through any cross-validation. Overfitting and sampling bias may therefore have an impact on performance.

The study in [38] has used wrapper models such the Bayesian Network, SVM, K-NN, DT, Logistic Regression (LR), and Artificial Neural Network (ANN) and the best first search strategy for feature selection. The number of features that each model chose from the DBC dataset varied. The comparative of model performance is conducted by comparing the accuracy metrics for both models, one with feature selection and the other without. The SVM model without feature selection fared better than other models, obtaining an accuracy of 97.36%.

The study in [39] has used the DBC data set, with SVM, radial basis kernel, ANN, and Naive Bayes are able to identify breast cancer with 98.82% accuracy, 98.41% sensitivity, and 99.07% specificity.

The study in [40] has used combined PCA and KNN had an accuracy of 96.4%. Recently, utilizing the k values and distinct distance functions, two separate breast cancer datasets are used to test the efficiency of KNN. Chi-squared features, linear SVM, and KNN without feature selection have all been used in trials.

A summary of the previous studies with additional details is listed in Table (1.1):

Table (1.1): Summarize the literature review

Previous Stud	Year	Techniques / Location	protocol used	Evaluation metric
[20]	2018	Inside Fog	MQTT	The accuracy and response times.
[35]	2018	PCA+KNN	-	Accuracy
[32]	2019	PCA+KNN	-	Accuracy
[33]	2019	SVM	-	Accuracy
[34]	2019	SVM	-	Accuracy, precision,F1 score
[17]	2020	Inside FogBus based edge/fog computing environment	HTTP	The accuracy and response times.
[18]	2020	In the edge between IoT devices and services with datacenter in the city	MQTT	A latency
[9]	2020	The fog broker decouples applications from fog nodes	MQTT	A latency and throughput
[19]	2020	In the edge and the same network	MQTT	A latency and response time
[10]	2020	In the Fog computing environment.	HTTP	A throughput, response time, and latency
[25]	2020	OCSVM	-	Accuracy
[37]	2020	LR with Area under the curve	-	Accuracy
[22]	2021	In the edge	MQTT	A latency
[24]	2021	OCSVM , anomaly detection	-	Accuracy
[29]	2021	SVM	-	Accuracy
[31]	2021	CNN	-	Accuracy
[2]	2022	Inside the edge with broker	MQTT	A latency
[26]	2022	KNN	-	Accuracy
[36]	2022	PCA And KNN	-	Accuracy
[23]	2023	communicate between IoT devices and the cloud	MQTT	A Latency

The first three works are considered the closest to this dissertation in terms of using a broker and several services in the architecture. They used the same measures for system performance, latency, response time, and throughput. The rest used the broker in different places and for different purposes. In contrast, the works [21,22,23] are identical to the services proposed in the dissertation in terms of anomaly detection and using the same database. The evaluation is based on the accuracy measure by applying the algorithms.

1.7. Dissertation Organization

After this introductory chapter, the rest of the dissertation is structured as follows:

Chapter Two, "Theoretical background": Many concepts will be introduced in this chapter, including datasets used in this dissertation, ML, SVM, RF, ISF, PCA, KNN and evaluation measures.

Chapter Three, "Methodology and Proposed System": This chapter first views the proposed methodology. Then, two parts will be discussed. The first part presents developed adaptive architecture. Furthermore, the second part explains the proposed ML algorithms.

Chapter Four, "Results and Discussion": This chapter presents the results of the proposed models on all the datasets used in this dissertation. Then, the results will be compared with the previous studies conducted on these same datasets and architectures.

Chapter Five, "Conclusions and Future Works": Many conclusions drawn from the proposed architecture, models, and suggestions for future work will be introduced.

Chapter Two

Theoretical Background

Chapter Two: Theoretical Background

2.1. Introduction

This chapter presents some techniques used in the architecture of IoT applications and services. Combining the MQTT broker with the ML, which is used in the services, can provide higher accuracy with increased processor speed and dynamic decision-making in efficient time . The ML is trained offline either through enhancing the ML parameters or building new predict model by selecting the best features. Systems need fast processing and making decisions automatically when increasing the overload on the services of the system without increasing the resources or adding new configurations. It is strongly believed that combining these components (IoT, MQTT broker, smart controller, and services) in the fog environment will have an impact on improving system performance and QoS. Avoiding obstacles, for example, bandwidth overload, data anomalies, stoppage of one of the system services, or loss of data, can lead to complete system failure or loss of reliability.

2.2. Internet of Things (IoT)

The IoT is a technology model that has the ability to influence our daily life where massive of data amount is generated from a huge number of heterogeneous physical devices with the many sources and data formats [41]. IoT devices are connected over the internet allowing collection, analysis, and sharing data such as data these which collected from the smart city or health care [42].

The IoT design architecture includes a server and always connected IoT devices so that the server can handle all complex activities and the IoT devices merely communicate with the server for data [43], as Figure (2.1) [44].

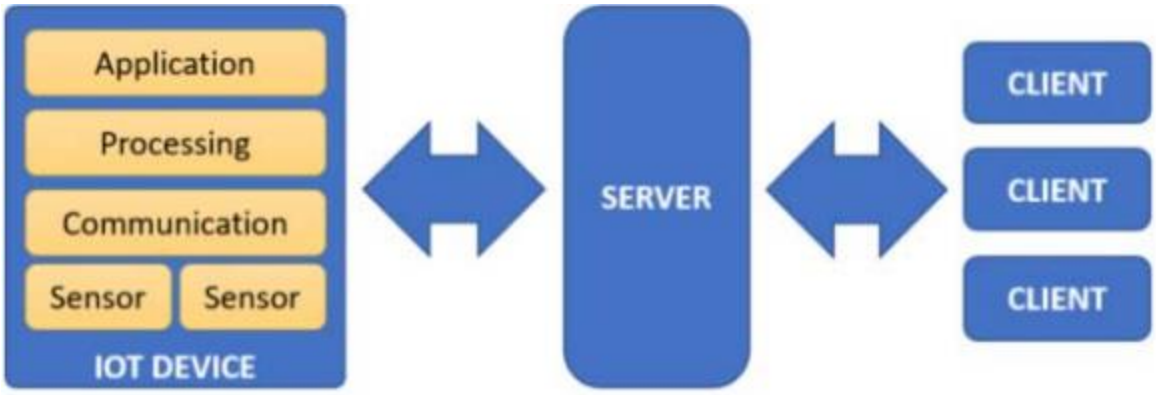


Figure (2.1): Architecture on IoT [44]

The IoT generates a large volume of data flows over time, and continually transmits data to gateways, edge nodes, or fog nodes, and finally to cloud centers[45]. The IoT data stream requires fast preprocessing, ample resources, a high response time, low latency, and high scalability [21]. Therefore, it is necessary to use CC which has the ability to spread resources, analyze and store data via the Internet, and thus perform processing on large amounts of data and make decisions. The cloud plays a role in providing central data analyses and data storage for future processing as well as being primarily responsible for complex, resource-intensive tasks, moreover, CC provides many resources over the network and allows the deployment of a set of applications that provide services in different fields [46].

There are several significant obstacles faced by the cloud; Like the centralized designs which might cause delays and performance difficulties. The distance between the data center and data resources is very large, thus consuming a significant amount of bandwidth on the network, having high latency, and causing the inability to meet the requirements of computing environments based on the central IoT [47].

When data analysis is centralized in the cloud, like when analyzing emergency circumstances, any communication breakdowns or response times will slow down operations and endanger lives [48].

Interdependence between various data sources is necessary to fully utilize the IoT data streaming for analytical purposes. These sources are heterogeneous by definition, making difficult to integrate them. Using ML algorithms for predictive analyses without taking domain expertise into account is a typical strategy for maximizing the potential of flow sensor data. A method like this is simple to be included in various use cases [49]. ML has been widely used in remote sensing because it can provide accurate predicted input-output data with strong correlations. This opens up a lot of opportunities for biophysical parameter retrievals and applications [50], [51].

In order to deal with the data IoT streams in efficient time, it can use window technology. which is a round-the-clock processing method where data is collected with timestamps, and grouped according to specified policies [11]. Thus, it can be preprocessed using ML for anomaly detection, and missing values. The size of the window is variable according to the nature of the data of the IoT, in terms of data transmission times, the number of devices, or the importance of data, as in high-priority health care [7]. FC is a solution to the problems of IoT because of the capabilities of processing, analysis, and storage [6].

The Edge represents any device located along the path between the data source and the cloud data center, such as a home switch or smartphone. The Gateway represents a device that mediates between IoT devices, and their services and is used to collect and transfer data securely [52].

2.3. IoT Devices

An IoT device is a collection of physical devices that are referred to as objects in IoT. They are unique in terms of identity, connected to the Internet, and allow remote sensing and monitoring capabilities [53]. The IoT devices generate big data from various geographic locations [1]. This data is processed and analyzed to produce useful information to direct further actions. These include devices such as thermostats, irrigation pumps, sensors, and other devices [54].

IoT devices are dynamic in configuration, adapting to changing data contexts and taking tasks based on operating conditions or the surrounding environment. As in the surveillance cameras, where they adjust their conditions (night mode, or normal) and depend on whether the time is day or night. There is a possibility of changing the camera resolution when any abnormal movement is detected and alerting nearby cameras to do the same task [55].

IoT devices can be discovered dynamically in the network by another device or the network itself. It can also describe itself and its characteristics to other devices or user applications. For example, a weather observation node can describe its monitoring capabilities to another node connected to the network so that it can communicate and exchange data. The data will be collected from a large number of IoT devices related to weather observation, analysis, and forecasting [56].

The IoT devices are a publisher for heterogeneous data streams over the MQTT broker and the subscribers receive their data of interest [57]. The subscribers and preprocesses of a data stream will be explained in the next section [58]. As shown in Figure (2.2) [59], the hierarchical architecture of IoT consists of several processing layers

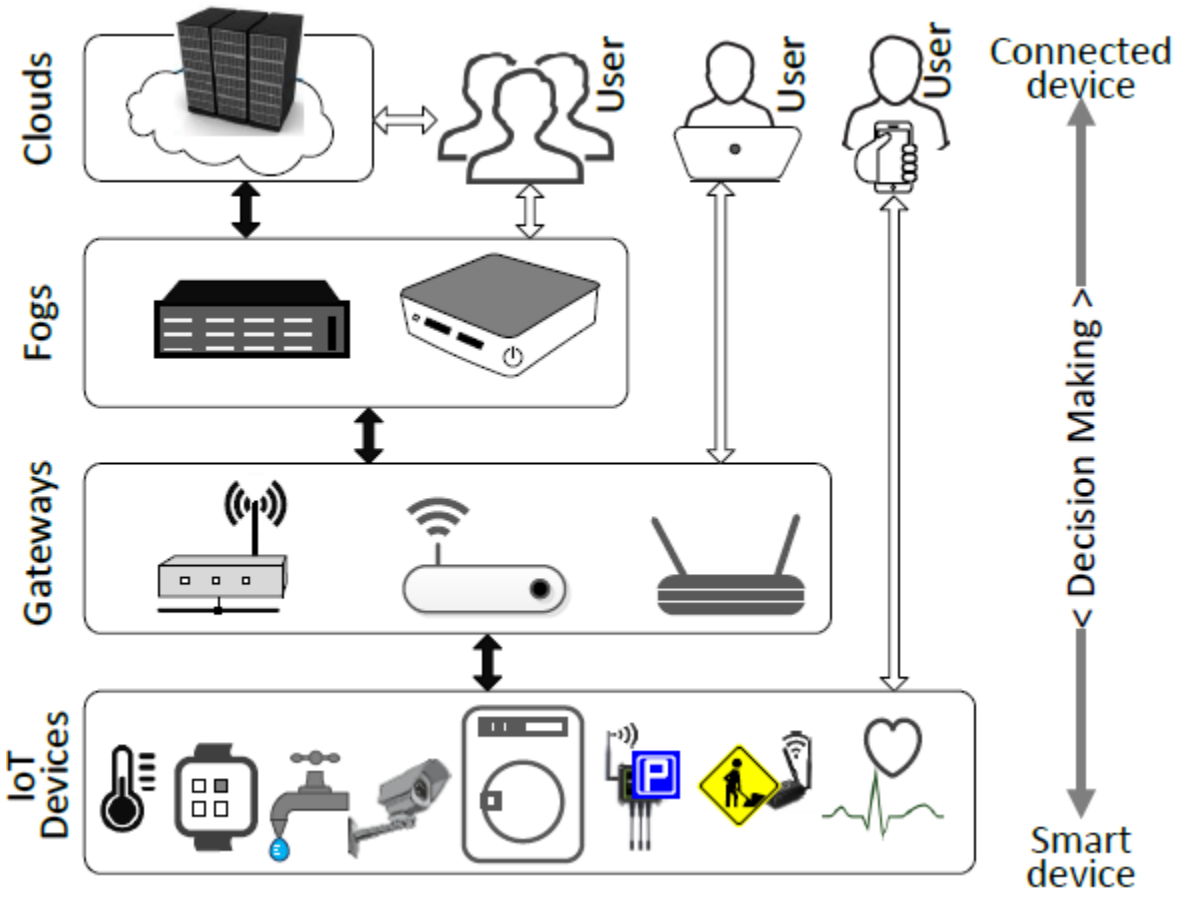


Figure (2.2): Hierarchical architecture of IoT processing layers [59]

2.4. Fog Computing (FC)

FC architectures offer intriguing options that enhance cloud-based systems, particularly for quick emergency response. It is a paradigm for computing that is developed to bring CC, storage, and network services closer to the network's edge [13]. The fundamental tenet of the FC approach is to relocate part of the storage resources and computing to the network's edge, close to the sources of the data[12]. FC alternative approaches to addressing the quality needs of time sensitive IoT applications, such as micro-cloud and information centers, are suggested [60].

FC is an easy solution to preprocess data before it reaches the cloud, which reduces latency and reduces external storage sources for large data by filtering it before sending it to the cloud. In general, fog is a suitable solution for applications and services that depend on the IoT [7]. FC brings storage, networking, and computation services nearer to edge devices. This is illustrated in Figure (2.3) [61], [62]. These end devices are known as fog nodes and can be positioned as a network connection at any place [61].

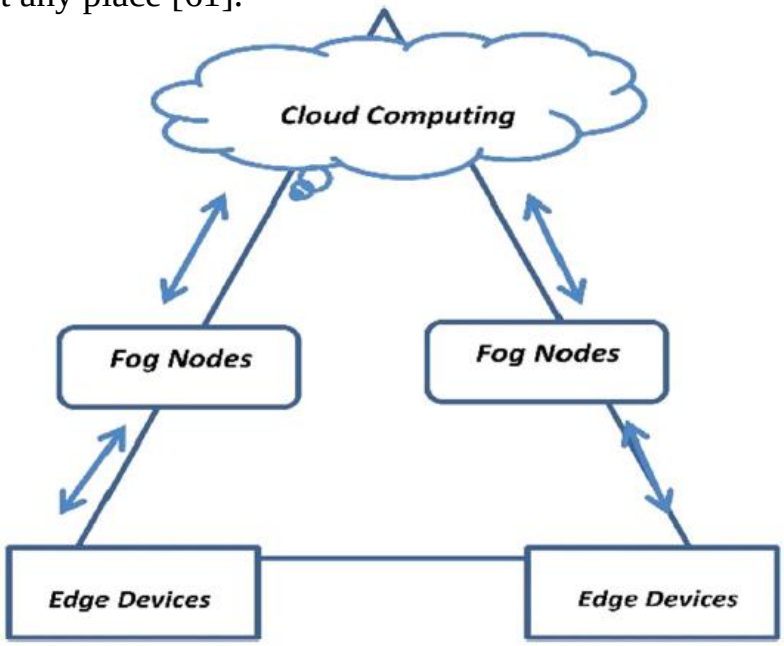


Figure (2.3): Fog computing is an expansion of the CC [61]

Processing of data and making decisions in FC reduces data transfer and ultimately increases computational efficiency on clouds by reducing the need to process and retain huge amounts of redundant data. The need for a FC paradigm is primarily related to the rising rise in IoT devices, where increasing amounts of data are generated from an expanding set of devices (in terms of size, variety, and speed) [52], [63].

The fog data life cycle is comprised of multiple stages, commencing with data acquisition at the device layer where data is generated. Subsequently, data

processing and storing occurs in upper levels, and feedback is sent back to the device layer. Ultimately, commands are executed in the device layer. We take into consideration five essential processes, as Figure (2.4) [13] illustrates: data collecting, study feedback, processing and analysis, lightweight processing, and command execution [13].

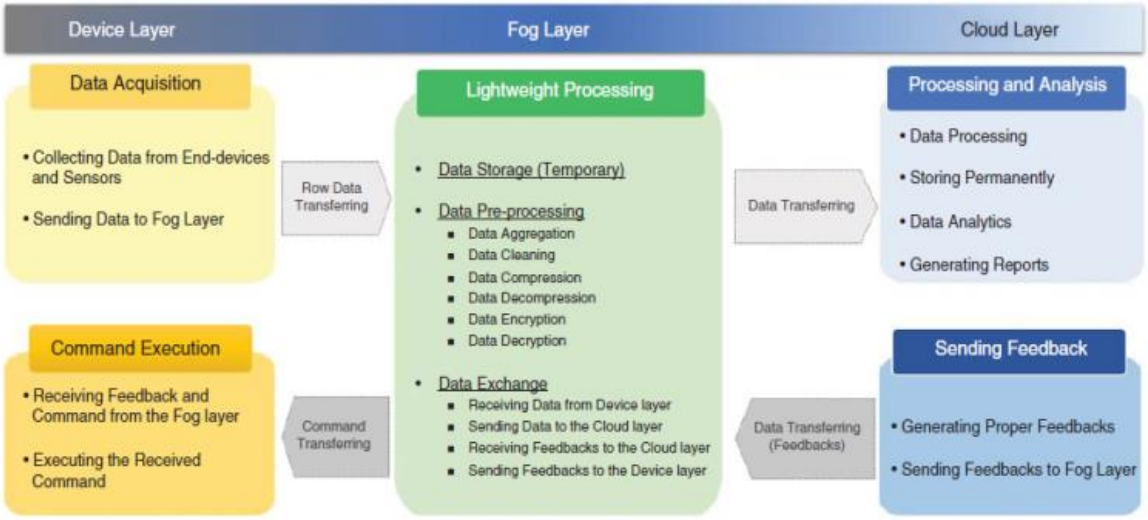


Figure (2.4): Data life cycle in fog computing [13]

The IoT, operational infrastructure monitoring, smart grids, smart cities, and intelligent buildings are some of the most common use cases for FC [64]. The benefits of FC are [13]:

1) Agility: Primarily on how to improve deployment of tasks across the distributed edges in terms of saving bandwidth and reducing latency.
2) Better security: It is possible to secure fog nodes using the same controls, processes, and policies as they are used in other information technology environment areas.
3) The FC in Smart Cities: Fog is considered a suitable approach for applications and services for (IoT) to process the requests in efficient time .

4) Reduced Operating Cost: FC aims to reduce the transfer amount of data from the sources (IoT devices) to the cloud and reduce the latency.
5) Bringing CC capabilities closer to the data source.
6) Enhancing the performance of the overall system.
7) Ensuring resource reliability and availability.

Fog is used to help with low-power facilitated computing and storage for time-sensitive applications like data analysis that employ IoT devices to complete their task. For the following causes fog is used:

1) FC has storage and processing capabilities are the same to the capabilities of the CC.
2) It makes it easier to bring these services closer to the edge near lower-powered peripherals and reduce the amount of data that travels to the cloud.
3) There are end-user applications that require low latency in accessing data.

Because of the inability of devices of IoT, and the limitations of cloud storage services, the role of Fog appears here as it improves the process of bringing the cloud closer to the end-user [65].

FC provides performance enhancement and resource savings, where both of which are necessary in many IoT applications and are closely tied to the development of technologies that cut down on bandwidth usage and processing time. when there are significant changes in the monitored system, the processing to transmit and receive messages is carried out; resource reductions are feasible when using FC [66].

The system needs light protocols to transition data, such as the MQTT protocol which ensures the transfer of data between IoT devices, and subscribers (services or applications) without dropping [67].

2.5. Message Queue Telemetry Transport (MQTT) Protocol

MQTT is a lightweight protocol for efficient communication between devices in constrained networks, such as those with low bandwidth, high latency, or limited computational power [68]. MQTT uses a publish-subscribe model for communication. "Topic" in MQTT indicates the name of the information [69]. Through a broker, the subscriber of the receiving device and the publisher of the transmitting device can communicate. This model allows for one-to-many and many-to-many communication [70]. It is suitable for communication with unreliable connections (suffering from interruptions) [71], and is well suited for IoT as it does not impose continuous sessions [72]. It can deliver the message from the source to the destination over several stages. The message can wait at each stage for a certain time before being re-sent (forwarded) to the next one, as in Figure (2.5) [72] the typical MQTT architecture can be divided into two main components

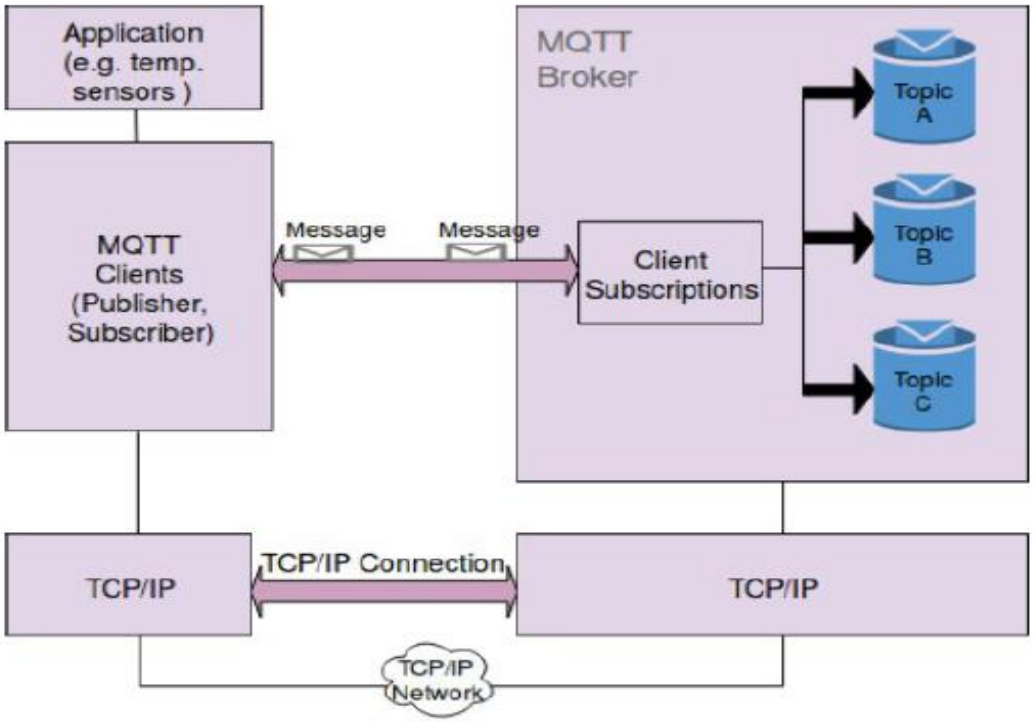


Figure (2.5): MQTT Architecture [72]

The MQTT protocol is used instead of the Hypertext Transfer Protocol (HTTP) [7], as it consumes less energy and bandwidth and is more appropriate for the development of IoT [73].

The HTTP (Hypertext Transfer Protocol) protocol is the most well-known in internet history. In this complex internet system, the HTTP protocol has grown to be crucial for data transfer [44]. Effectively establishing a connection between an IoT and the internet may not be as straightforward as it first appears to be because of limitations include header size restrictions, latency, completely connection-oriented design, and others [57]. The topic is condensed by examining the MQTT Protocol and all of its applications [74].

The main difference between HTTP and MQTT is in the communication model. HTTP uses the request response communication technique, while MQTT uses the publish subscribe architecture. Sensor, dashboard, and mobile app installations are examples of generic IoTs deployments. The first thing is a publisher gets data from a variety of sources, including run sensors in machines or wearables, embedded mobile sensors, and more [75]. The second thing is a subscription. The topic that the subscriber chooses, for which the publisher provides data. The subscriber could be a mobile application or a user dashboard. The broker is the third and most crucial item. The broker's primary responsibilities are to obtain data from the publisher and transmit it to the subscriber. The broker will be equipped with innovative and resourceful tools that can manage multiple challenges at once [76].

Constrained Application Protocol (CoAP) is an application layer protocol for machine-to-machine (M2M) applications, intended for restricted networks and devices [1]. CoAP is a web transmission protocol that follows the same request-response model as HTTP, although it operates over UDP rather than TCP. CoAP employs a client-server architecture in which clients use connectionless datagrams

to exchange information with servers. CoAP is made to interface with HTTP with ease. CoAP supports GET, PUT, POST, and DELETE commands, just like HTTP does [9]. Determining a protocol that works with an IoT system depends on the nature of the data, the problem, and the environment [77]. The CoAP operates over UDP, which suffers from unreliable, no flow control, and no ordered delivery [78]. In addition the CoAP protocol suffers from the following [79]:

1. Reliability and Ordering: CoAP doesn't guarantee ordered delivery of messages. In scenarios where the order of messages is critical, additional mechanisms need to be implemented at the application layer.
2. Scalability Challenges: CoAP has challenges in highly scalable environments. The protocol may require additional mechanisms to handle large-scale deployments effectively.

One of the components required for each publish or subscribe messaging is a message broker. There is a message broker, a publisher client, and a subscriber client in the diagram of the MQTT signal flows. To exchange messages using MQTT's publisher and subscriber protocol, the client must connect to the broker [80]. To guarantee continuous data flow, distribute data on the services based on certain topic names, and avoid system failure, the next section explains the properties of MQTT broker.

2.6. MQTT Broker

The MQTT broker is a dynamic message distributor that provides the continuity of transferring messages between the publishers and the subscribers [81]. The broker does the publishing and subscription process between publishers and consumers then the publishers are not aware of the consumers [68]. Consumers

subscribe to the topics which are managed by the broker [1]. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers, in other words, delivering the right topic information to the right subscriber [2], [82].

The broker is located between IoT devices network and service or application server in FC to make decisions efficiently, thus that can receive blocks of data with the specified topic of required processing and deliver data with reformed in new structure and topic to interested subscribers [21]. Some IoT services require simplified mechanisms in networks in order to communicate reliably and with low latency[64].

The broker often uses the MQTT protocol, which utilizes a lightweight subscription-publish messaging mechanism. It is preferred by IoT devices since it uses little bandwidth and can serve as a finite resource [74]. A client (such as an IoT device) connects to a server (also known as an MQTT broker) in MQTT's client-server architecture and sends messages based on specific topics.

Customers who have subscribed to topics receive messages from the broker. As illustrated in Figure (2.6) [1], [15] which shows the general architecture for MQTT broker and the publisher not aware of the subscriber.

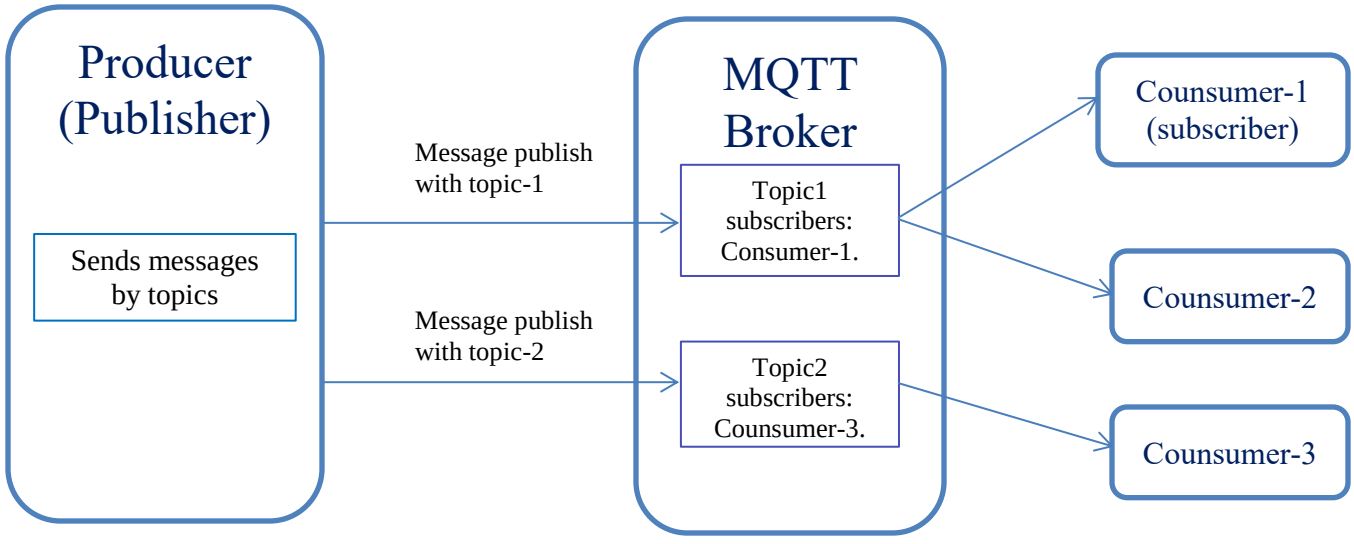


Figure (2.6): General form MQTT Broker [1],[15]

2.6.1. The Services/ Process Subscriber

The publish-subscribe service is a practical decoupling approach between the publisher and subscriber, where subscribers (services) can subscribe to data of their interests from data publishers (IoT devices), and the publisher is not aware of subscribers[83].

The service is a standalone component capable of managing specific activities and functions. The subscribe services are a group of running services in the system for performing specific tasks. It can use ML algorithms to preprocess data and have subscribed to the broker according to the interesting topics [84].

They receive, collect, and preprocess data in blocks and republish clean data back to the broker for the benefit of consumer services, cloud, or applications. Any service can be added to the system by the administrator. The IoT platform provides instance services (similar to the original service but run when needed) to meet user requirements such as cleaning, analysis, and decision-making. The number and type of these services vary according to the required data quantities and preprocessing [85].

In general, message patterns are dealt with using the pub-sub paradigm. A publisher-subscriber arrangement involves three parties: publishers, subscribers, and a broker. A subject can be created by publishers and subscribers in several pub-sub formats. It is sent to the broker by the subject creator. Following the creation of a topic, the subscriber uses the broker to create a subscription with the necessary details[85].

Inside the services, the data is received, collected, preprocessed, and republished cleanly back to the SB to benefit other services, like the cloud or applications. A service has numerous instances, each works on a distinct data block, runs the same algorithm, and may be executed anywhere. Examples of

services are anomaly detection, data missing, and data correction. Service may contain all of them. As illustrated in Figure (2.7) [68] which shows the broker model with publishers (IoT devices) and subscribers (services).

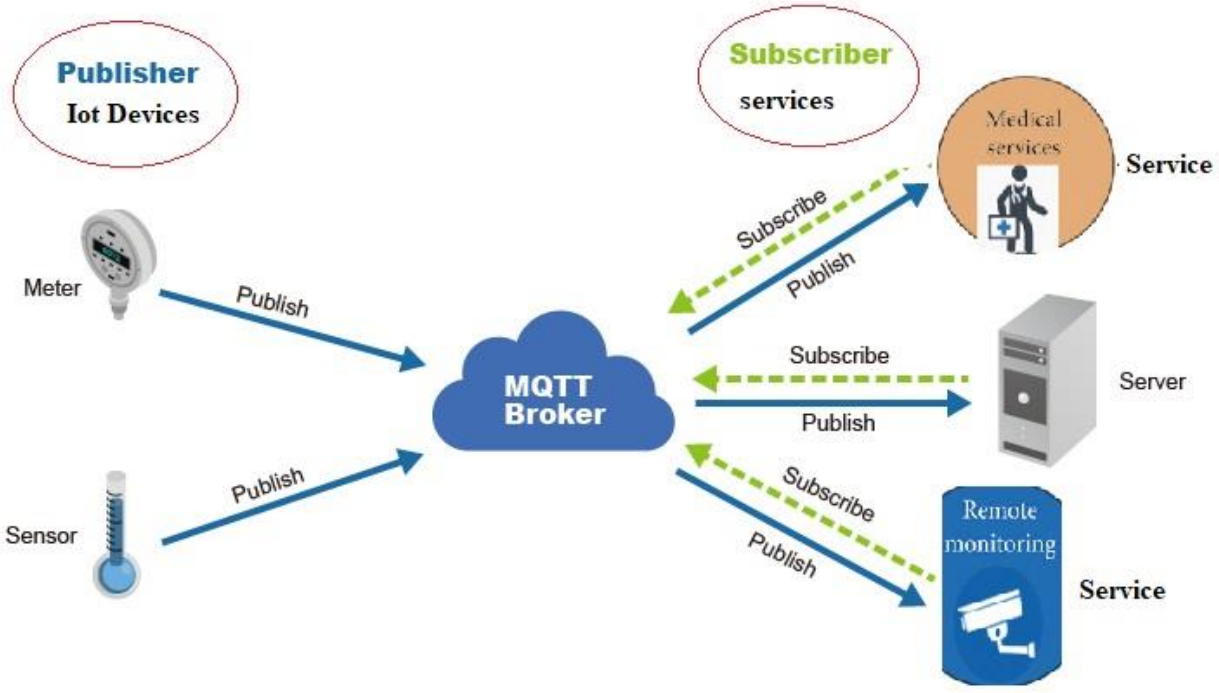


Figure (2.7): The Publish Subscribe an MQTT Broker [68]

The data received by the service that are subscribed to the SB is based on a specific topic. One of these services is the data collector and detector. For each sensor (or group of correlated sensors), there is a specific collector subscribed to their specific topic or global one. It collects a series of readings into blocks. The size of the block is determined experimentally. Then, data is passed to the part of the system that performs anomaly detection for the purpose of knowing whether there are anomalies or not. This service republishes the data block to the SB with a new topic (normal or abnormal). Using a method to analyze data streams inside of the services (subscriber) with high accuracy, processing speed, and reliability by benefiting from the key advantages of ML in IoT situations [86].

A set of measures are used to assess the proposed model. They are widely used and well-known by researchers [87]–[89]. The most used measures can be summarized as follows:

1. Throughput: The quantity of jobs (tasks) a system can perform in a specific period of time measured in jobs per second (J/s).

   Throughput = *#jobs/ duration in s* ***(2.1)***

2. Latency: The period from submitting a first bit of a job to starting execution time.

   Latency= *starting execution time - 1st message submitting time* ***(2.2)***

3. Execution time: the difference between starting and finishing time of a job.

4. Data rate: refers to the speed at which data is transmitted or processed over a communication channel, or computer network. It is measured in bits per second (bps).

   Data rate = *Total-Data / Total-Time* ***(2.3)***

   Total-Data = *Max-Data-Min-Data* ***(2.4)***

   Where Total-Data is all data sent during the experiment

5. Response time: the duration from submitting a job until receiving the first response from the destination (finish time in batch processing).

   Response time= *finish time submitting* ***(2.5)***

6. Total time: the difference between submitting time and finishing processing time.

   Total-Time = *finish time - Start-Time* ***(2.6)***

7. Data Loss : the difference between the total amount of data submitted by the source and the actual data received by the service.

   Process-Data =Δ*T * size-Block (size bloks in the services)* ***(2.7)***

   Data-Loss = *Total-Data – Process-Data* ***(2.8)***

   ΔT: *a period size to aggregate the information from the services.*

2.7. Machine Learning (ML)

Artificial intelligence (AI) is widely used in various industries in today's society, as demonstrated by its applications in financial trading, medical diagnosis, and customer service. Furthermore, AI is utilized more and more in time-sensitive and mission-critical applications like self-driving cars, cybersecurity, and remote surgery. AI also has a significant role to play in developing and future applications. Artificial intelligence (AI), which has gained popularity by using resource-intensive cloud infrastructure, must function well in resource-constrained contexts at the network edge in order to meet the latency, security, and privacy requirements that such applications demand [90].

The ML is a type of Artificial Intelligence (AI) [91], a branch of computer science that provides machines with the ability to learn without explicit programming [92]. The challenges of latency, scalability, and privacy for the edge can be overcome using ML. An effective ML algorithm, such as SVM, One-Class-SVM, KNN, PCA, Random Forest (RF), and Isolation Forest (ISF), can be deployed on an edge server that has the ability to receive and collect data from IoT devices and sent to services or the cloud.

By developing prediction models, it may be possible to identify diseases earlier and provide patients with more effective treatment. ML models have demonstrated significant performance when used to diagnose breast cancer in earlier research [93], [94]. It is possible to use predictions or facts derived from experience. To determine the most precise link between variables, a variety of application methods can be applied, such as early breast cancer prediction, forecasting jobs, and time-series techniques [95].

The features can be exploited to implement artificial intelligence algorithms on devices with limited resources, allowing local data processing [96]. Bringing

ML to the edge to meet the latency requirements, different fast processing. Most ML developers focus on designing models with few parameters, thus reducing memory and execution latency while maintaining high fidelity. On the other hand, crucial to remember that the qualities of a dataset and the nature of problem should guide algorithm selection. In order to find the algorithm that performs best for given task, it is frequently the smart practice to test out a variety of them and evaluate their effectiveness using cross-validation.

Additionally, some of these restrictions can be reduced with the aid of preprocessing, feature engineering, and hyperparameter optimization [97].

2.7.1. Support Vector Machina (SVM)

A Support Vector Machine (SVM) is a supervised machine learning algorithm that is primarily used for classification tasks, and regression (SVR) and outlier detection as well. The SVM ability excel in high-dimensional spaces and can handle complex decision boundaries, making them particularly well-liked in the field of machine learning. The SVM is primarily used for classification tasks. It is used to find a hyperplane (a decision boundary) that best separates the data into different classes. SVM classifies data points into one of two or more classes based on the side of the hyperplane they fall on. It outputs class labels. The SVM objective is to maximize the margin (distance between the hyperplane and the nearest data points of each class) while minimizing classification errors.

To find a nonlinear decision boundary, the SVM can use a kernel (e.g., linear, polynomial, radial basis function) to transform the data into a higher-dimensional space [98]. SVR (Support Vector Regression) is used for regression tasks. To make precise predictions on continuous numerical data, it seeks to identify a hyperplane that best fits the data. Since the output is a continuous numerical number, it can be

used to forecast outcomes with real values. The SVR seeks to maintain a margin that permits some prediction error while minimizing the error between the projected values and the actual target values [99].

Limitations of usage: Due to SVM is sensitive to the scale of input characteristics, feature scaling is frequently necessary. Large-scale SVM training can be computationally expensive, especially when nonlinear kernels are used. SVM is intended for classification in binary form. The use of one-vs-all or one-vs-one approaches is necessary for multi-class classification.

2.7.2. One-Class-SVM (OCSVM)

The One-Class Support Vector Machine algorithm is an unsupervised system uses to detect anomalies the examples of training or negativity are very few as in the anomalous events. It uses a single classifier without prior learning which has many advantages. It can deal with non-linear solutions as well as complex decision limits by using the kernel to solve the linear data problem by finding the appropriate curve for the data instead of the hyper level. It is a suitable technology in the environment of various and heterogeneous devices, such as IoT devices, with massive data streams generated in efficient time . It is considered as a very powerful algorithm when both the number of features and training samples are small. It has the objective to creating a boundary that enables us to distinguish the normal data from those that are abnormal [100].

In this dissertation, the anomaly detection [101] model will be used based on the OCSVM algorithm which given its ease of implementation and its proven robustness to outliers during the training. It works by finding the optimal separation boundary, which is called a hyperplane, between two classes [102].

When using the kernel trick to apply the One-Class SVM (OCSVM) in a non-linear feature space, the decision function becomes dependent on the kernel function and the kernel parameter (gamma). The kernel function implicitly computes the dot product between the data points in the higher-dimensional space. The OCSVM optimization problem with a kernel (commonly used with the Radial Basis Function (RBF) kernel) can be defined as follows:

Given a training dataset with data points $\{x_1, x_2, ..., x_i\}$ where x_i is a d-dimensional vector, the OCSVM aims to find the hyperplane in the feature space defined by the kernel that maximizes the margin while capturing most of the data points. Let Φ be a feature space, mapping space X to F, $X \rightarrow F$, i.e. a transformation into a dot product space F such that the dot product is calculated using:

$$\boldsymbol{k(x_i, x_t) = (\Phi(x_i).\Phi(x_t))} \quad \textbf{(2.9)}$$

This means that $\boldsymbol{k(x_i, x_t)}$ is achieved by the dot product of $\boldsymbol{\Phi(x_i)}$ and $\boldsymbol{\Phi(x_t)}$. In our case, $\boldsymbol{k(x_i, x_t)}$ is a Gaussian kernel and $\boldsymbol{x_i}$ and $\boldsymbol{x_t}$ are sample patches [103]:

$$\boldsymbol{K(x_i, x_t) = e^{(-\gamma * ||x_i - x_t||^2)}} \quad \textbf{(2.10)}$$

where $\boldsymbol{\gamma}$ controls the width of the Gaussian kernel. In the context of One-Class Support Vector Machines (OCSVM), the parameter γ is often associated with the RBF (Radial Basis Function) kernel. The γ parameter determines the width of the Gaussian distribution in the kernel function and influences the shape of the decision boundary.

A smaller γ results in a wider distribution and a smoother decision boundary, while a larger γ leads to a narrower distribution and a more complex, finer-grained decision boundary [29].

The goal is to find function $\boldsymbol{D}$ that outputs the value +1 in a "small" region where data distribution of the target class exists, and −1 elsewhere. The solution is

to map the data into the feature space corresponding to the kernel and to separate them from the origin with a maximum margin by calculating a separating hyperplane. Assigning a new input point x to the target class is determined by evaluating which side of the hyperplane (in the feature space) it belongs to. The decision boundary is determined by the support vectors and their associated weights. The decision function for OCSVM is given by [102]:

$$\boldsymbol{D(x) = sign(\sum_{i=1}^{nsv} \alpha_i \; K(x, x_i) - \rho)} \qquad \textbf{(2.11)}$$

Where:

- $\boldsymbol{D(x)}$ is the decision function for a data point $\boldsymbol{x}$.
- $\boldsymbol{nsv}$ is the number of support vectors.
- $\boldsymbol{\alpha i}$ is the weight associated with the support vector.
- $\boldsymbol{K(x, x_i)}$ is the kernel function (in this case, the RBF kernel) between the test point $\boldsymbol{x}$ and the support vector $\boldsymbol{x_i}$.
- $\boldsymbol{\rho}$ is the offset term.

OCSVM has a parameter called $\boldsymbol{C}$ that represents the upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors. It controls the trade-off between having a smooth decision boundary and classifying points as abnormal. The steps to compute the decision boundary using the RBF kernel in OCSVM [29]:

- Compute the RBF kernel between $\boldsymbol{x}$ and each support vector $\boldsymbol{x_i}$: $\boldsymbol{K(x, x_i).}$
- Multiply each kernel value by the corresponding weight ($\boldsymbol{\alpha_i}$) and sum them using equation (12).
  $$\boldsymbol{\sum_{i=1}^{nsv} \alpha_i \; K(x, x_i)} \qquad \textbf{(2.12)}$$
- Subtract the offset term ($\boldsymbol{\rho}$) and take the sign of the result equation (13).
  $$\boldsymbol{sign \sum_{i=1}^{nsv} \alpha_i \; K(x, x_i) - \rho)} \qquad \textbf{(2.13)}$$
- The sign indicates whether the point $\boldsymbol{x}$ is classified as normal or an anomaly.

Performance evaluation of the model and accuracy by using four main parameters that are used in computing the accuracy of the models of ML.

1. TP (true positive) predicts positive if the correct answer is positive;
2. TN (true negative) predicts negative if the correct answer is negative;
3. FP (false positive) predicts positive if the correct answer is negative;
4. FN (false negative) predicts negative if the correct answer is positive.

Accuracy: is a performance measure used in classification problems. The classification aims to determine the class of new instances within the predefined classes. Accuracy measures the proportion of correctly predicted instances out of the total number of instances[104]. **Table (2.1): summarizes the following explanations** [105]**.**

Table (2.1): Confusion matrix

	Positive for Predicted Value	Negative for Predicted Value
Positive for actual value	TP (True Positive)	FN (False Negative)
Negative for actual value	FP (False Positive)	TN (True Negative)

The proportion of accurately fitted data points in all the data is referred to as accuracy. It ranges from 0 to 1, and the nearer it is to 1 the better. The relationship can be expressed as:

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad \textbf{(2.14)}$$

$$\textbf{Precision} = \frac{TP}{TP+FP} \qquad \textbf{(2.15)}$$

$$\textbf{Recall} = \frac{TP}{TP+FN} \qquad \textbf{(2.16)}$$

$$\textbf{F1 Score} = \frac{2*(precision*recall)}{precision+recall} \qquad \textbf{(2.17)}$$

Limitations of usage: It can be difficult to select the proper kernel and tune hyperparameters, and the performance of the model may be affected by these decisions. When the data has complicated or irregular distributions, OCSVM cannot perform adequately.

2.7.3. K-Nearest Neighbors Regression (KNNR)

The KNN algorithm is a supervised machine-learning tool that can be employed for sorting, classification, and regression. In the current study, regression is used to determine the relationship between the input and output data. An output numerical value has been predicted based on the given input. It is imperative to use the KNNR for estimating the exit parameters and evaluating the performance of the model because this tool produces great accuracy in a variety of prediction situations [106]. It is primarily a classification algorithm, but it can also be used for anomaly detection by leveraging the concept of nearest neighbors.

The basic idea is that anomalies are often distant from the majority of data points in the feature space [107]. Here are the steps to perform KNNR:

1. Data Preparation: Ensure have a labeled dataset with input features and corresponding target values. Split the dataset into a training set and a test set to evaluate the performance of the model.
2. Choosing the Number of Neighbors (k): Decide on the number of neighbors, k, to consider during prediction. A larger k value smoothens the regression line, but it might lose local patterns, while a smaller k value might leads to overfitting. A common approach is to cross-validate the experiment with different k values and select the one that gives the best performance.

3. Distance Metric: Choosing an appropriate distance metric (e.g., Euclidean distance) to measure the similarity between data points. Euclidean distance is commonly used for continuous numeric features.
4. Training: During training, the algorithm simply memorizes the entire dataset.
5. Prediction: For each data point in the test set, do the following:
    - Calculating the distance between the data point and all the points in the training set using the chosen distance metric.
    - Selecting the k-nearest data points (data points with the smallest distances).
    - Taking the average (or weighted average) of the target values of these k-nearest data points to get the predicted value for the test data point.
6. Evaluating the Model: Calculating the performance metrics (performance evaluation by equations 4,5,6, and 7) on the test set to assess the model's accuracy.
7. Hyperparameter Tuning: Experiment it with different values of k and choose the one that gives the best performance on the test set.
8. Predicting New Data: Once the model is trained and evaluated, it can be used to predict the target values for new unseen data points.

Remember that KNNR is computationally expensive, especially on large datasets, as it requires calculating distances between each test point and all training points. Additionally, scaling the features might be necessary to avoid undue influence from features with larger scales [108].

Regression deals with continuous numeric values; classification deals with presorted class labels. The aim of regression is to predict the continuous values. Therefore, it is preferable to use the Mean Square Error (MSE) in measuring the

performance of the model rather than the accuracy [109].

MSE: Measure the mean squared difference between predicted values and actual values. It is used as a function to evaluate the performance of regression models. It predicts a continuous numeric value. In order of importance, it gives more weight to the larger errors that are undesirable when solving regression problems [110].

$$\mathbf{MSE}=\frac{\sum(\boldsymbol{y}_i-\hat{y}_i)^2}{\boldsymbol{N}} \qquad \textbf{(2.18)}$$

Where:

y_i: represents the actual values.

$\hat{y}_i$: represents the predicted values.

N: represents the total number of points.

Limitations of usage: Due to its reliance on the close proximity of data points, the KNN algorithm can have several drawbacks, such as being susceptible to noisy data and outliers. Predictions can be dramatically affected by outliers.

Given that every forecast necessitates calculating the distances between data points, it can be computationally expensive, especially when working with huge datasets.

2.7.4. Principal Component Analysis (PCA)

In machine learning, there is a need to know models and ways of working. In fact, machine learning is not only related to models because models are like the brain that needs to learn about new things. Data is what teaches the brain. Model learning is best when the data provided is well represented. Therefore, processing and analyzing data is an important part of machine learning.

The outcomes (data output) of the models need to be interpreted. It is observed that if the used data set contains unbalanced data for each category, especially in classification problems, it leads to poor learning of the model. On the contrary, the accuracy of the model is good because the test data will be strongly represented and affect the category that is predicted. Here, it has a good classification but no skill. Therefore, the output must be analyzed before being published. Data analysis is of two types: exploratory Data Analysis EDA and Principal Component Analysis PCA. The second method will be explained in this work.

Feature selection and dimensionality reduction are two techniques used in machine learning and data analysis to reduce the number of features or dimensions in a dataset. While their goals are similar.

Feature selection is a critical stage in building ML classifier models, it has a big effect on the model's performance, training time, and interpretability. It is a widely used data preprocessing technique that enhances the performance of the data analysis process while lowering the number of features [111].

The features that are redundant and irrelevant are eliminated during feature selection since they are of little value in describing the data. When there are intra-class differences between objects, choosing a single feature subset to represent the entire dataset may reduce performance [112].

In ML, there are various crucial feature selection techniques, such as Filter Methods, Wrapper Methods, Embedded Methods, and PCA is a dimensionality reduction technique that can be used for feature selection by selecting a subset of principal components that capture most of the variance in the data. Dimensionality reduction is the method of minimizing the number of variables taken into account. It can be applied to decrease data while preserving structure or to extract latent properties from raw datasets. When you have a lot of features and wish to cut

down on computational overhead, this can be helpful [113].

PCA is primary premise is the separation of feature groups, with the goal of reducing reciprocal correlation and sorting in accordance with a dropping eigenvalue and subsequently a declining variance. Principal components are another name for eigenvectors. They are initially subject to standard normalization because of the different feature domains[114].

Each column of the database (the feature set without the target) is used as a vector of values in the model. To relate the two feature values, use the dot product of the two feature vectors, which gives the covariance among the feature vectors. The eigenvalue and the correlation of the initial variables are closely related. The number of characteristics that can be decreased rises as the correlation strength grows because larger eigenvectors are possible. Weakly correlated input variables will cause r to be similar to p and an increase in the loss of initial variance of a set of features. The number of samples may be maximally equal to a reduction in dimensions [115] . Steps of the operation of PCA:

1. Data Preparation: Assume have a dataset with multiple features (dimensions). For simplicity, consider a 2D dataset with two features: X and Y.
2. Data Standardization: If the features have different scales, it's often recommended to standardize the data (subtract mean and divide by standard deviation) so that all features contribute equally to the analysis.
3. Computing the Covariance Matrix: The next step is to calculate the covariance matrix, which shows how the features vary in relation to each other.
4. Calculating eigenvectors and eigenvalues: The eigenvectors and eigenvalues of the covariance matrix represent the principal components of the data. The eigenvectors are orthogonal (perpendicular) to each

other and point in the directions of maximum variance in the data. The eigenvalues indicate the amount of variance explained by each principal component. Higher eigenvalues correspond to more important principal components.

5. Sorting eigenvalues and eigenvectors: Sorting the eigenvalues in descending order to identify the most important principal components.
6. Choosing the Number of Principal Components (KP): Decide how many principal components (KP) to retain. Typically, researchers select the top (KP) components that explain a significant portion of the data variance, often using a cumulative explained variance threshold.
7. Projection Matrix: Create a projection matrix by selecting the top (KP) eigenvectors corresponding to the chosen principal components.
8. Projecting the data onto the new subspace : Multiply the original data with the projection matrix to obtain the transformed data in the new coordinate system.
9. Visualization: Plot the transformed data in the new coordinate system, where the axes represent the principal components.
10. Interpretation: Interpret the results. The first principal component explains the most variance, the second explains the second most variance, and so on.

The covariance matrix provides information about the relation between features. It consists of covariance and variance for each feature[116]. As illustrated in Table (2.2), the covariance matrix uses to find the eigenvector (data difference in the original feature space) and eigenvalue (the principal component).

Table (2.2): Covariance Matrix		
	Feature 1	Feature 2
Feature 1	Variance	Covariance
Feature 2	Covariance	Variance

Using PCA, the dimensionality of the data can be reduced while preserving the most critical information. This is particularly useful for visualization, noise reduction, and speeding up machine learning algorithms that might struggle with high-dimensional data [117].

A set of projection vectors are acquired by conventional PCA in order to extract global features from supplied training patterns. It operates directly on an entire pattern represented as a vector [118].

Limitations of usage: The major components that are produced after using PCA might not be easily understood in terms of the original features.

2.7.5. Isolation Forest (ISF)

The ISF is an unsupervised ML algorithm for finding anomalies effectively in massive data streams. Based on the idea that anomalies are easier to spot because they contain few and varied observations, the ISF is worked. A set of isolation trees for the provided data points is used by ISF to isolate anomalies. As in illustrated in Figure (2.8), it shows that the classification the anomaly and normal data.

DT is the foundation of ISF. By randomly choosing a feature from the available features and then a split the value between that feature's maximum and minimum values, it separates the outliers. The anomalous data points will have shorter trajectories in trees as a result of the random feature partitioning, making them stand out from the rest of the data.

Creating a profile of what is "normal" is typically the first step in anomaly identification. Anything that does not fit this profile should then be reported as abnormal. The ISF algorithm does not, however, operate according to this theory; it does not compute point based distances or define "normal" behavior first. The ISF randomly selects a feature and then a segmented value for the feature to produce divisions in the dataset. Anomalies require fewer random partitions to be isolated than "normal" points in the data set [111].

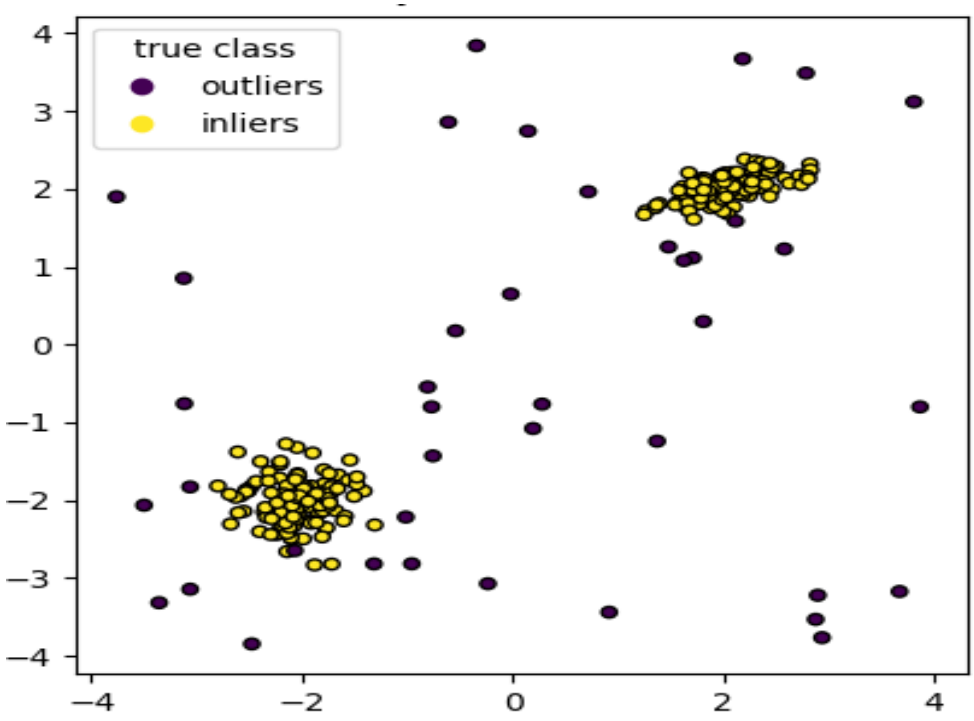


Figure (2.8): Classification Using ISF [111]

Limitations of usage: Due to its reliance on random sub-sampling, ISF may struggle to handle high-dimensional data or data with complicated dependencies. Limited interpretability entails the potential inability to offer insightful explanations for why a data point is seen as an outlier.

2.7.6. Random Forest (RF)

The Random Forest is a decision tree-based integrated method, widely used in a variety of fields. It works on the premise of building several decision trees by randomly choosing samples and attributes, then, integrating these decision trees for integrated prediction. As a result, a RF may analyze the value of different features by calculating how often each feature is used throughout all decision [111]. The RF algorithm requires training several decision trees. The results of each separate tree are combined to produce the algorithm's ultimate result. Each decision tree independently creates its output, and for regression tasks or classification tasks, the final prediction is arrived at by averaging the responses or by majority vote.

For regression tasks, it integrates the ideas of both decision trees and ensemble learning. It is a robust and adaptable regression approach, which is used to generate predictions on continuous numerical data [119].

Limitations of usage: Although RF is renowned for its accuracy, the collection of decision trees it employs can be more complex to understand than a single decision tree. Although less prone to overfitting than individual decision trees, it is still possible to overfit noisy data. Due to RF's ensemble structure, inference (forming predictions) may be slower compared to simpler models.

Chapter Three

Methodology and Proposed System

Chapter Three: Methodology and Proposed System

3.1. Introduction

This chapter presents the methodology and proposed system that starts from IoT applications and services to combining the MQTT broker with the ML, which is used in the services and can provide higher accuracy with increased processor speed and dynamic decision-making in efficient time . IoT systems (Healthcare, weather, Smart city) need fast processing and making decisions automatically when increasing the overload on the system's services without increasing the resources or adding new configurations. In the same context, the services (subscribers) have been subscribed on the broker under a specific topic name as the initial service used to process data streams published from IoT devices (publishers) that have the same topic name in the fog node.

The SC is used to overcome the problems of overload, high response time, and low latency by dynamically making decisions, which are either adding a new service (spawn) or removing one of the currently deploying services (kill) after completing the current task, depending on the information messages (containing the performance measures, topic name, and timestamp) that came from the deploying services and saving them in the screen table for making suitable decisions.

3.2. The Architecture of the Proposed System

The main objective of this dissertation is to enhance system performance and reduce congestion on the services by making decisions to spawn or kill them automatically. All services in the system have been subscribed to the broker based on a specific topic. These services collect and preprocess data before sending it to the cloud as clean data (does not contain missing values and anomalies). The SC is controlling the processing speed-up by predicting the number of services that avoid data congestion on the service. By integrating the algorithms PCA and KNN, accuracy and prediction efficiency can be increased.

IoT devices publish data messages to the broker, which distributes the data to services according to the subscription topic. The SC monitors the services states that are currently deploying. The services collect data to perform anomaly detection and data correction. The SC is aware of any increase in the latency of preprocessing services. So, if there is a congestion on one of the current services, the SC deploys a new service. As it is illustrated in Figure (3.1), represent the general proposed architecture and main tools.

In this dissertation, in the SC integrating the PCA and KNN are proposed, which are the PKNSP model for predicting of the number of services, and applying the proposed model on the different dataset for predict of the breast cancer. To prove the effectiveness of these models, they evaluated by the accuracy, precision, f score, and MSE on the different datasets and fields.

To evaluate the proposed architecture, databases containing real values that collected from sensors are used as IoT devices. The data are published as messages over the broker to subscribers under the specific topic name.

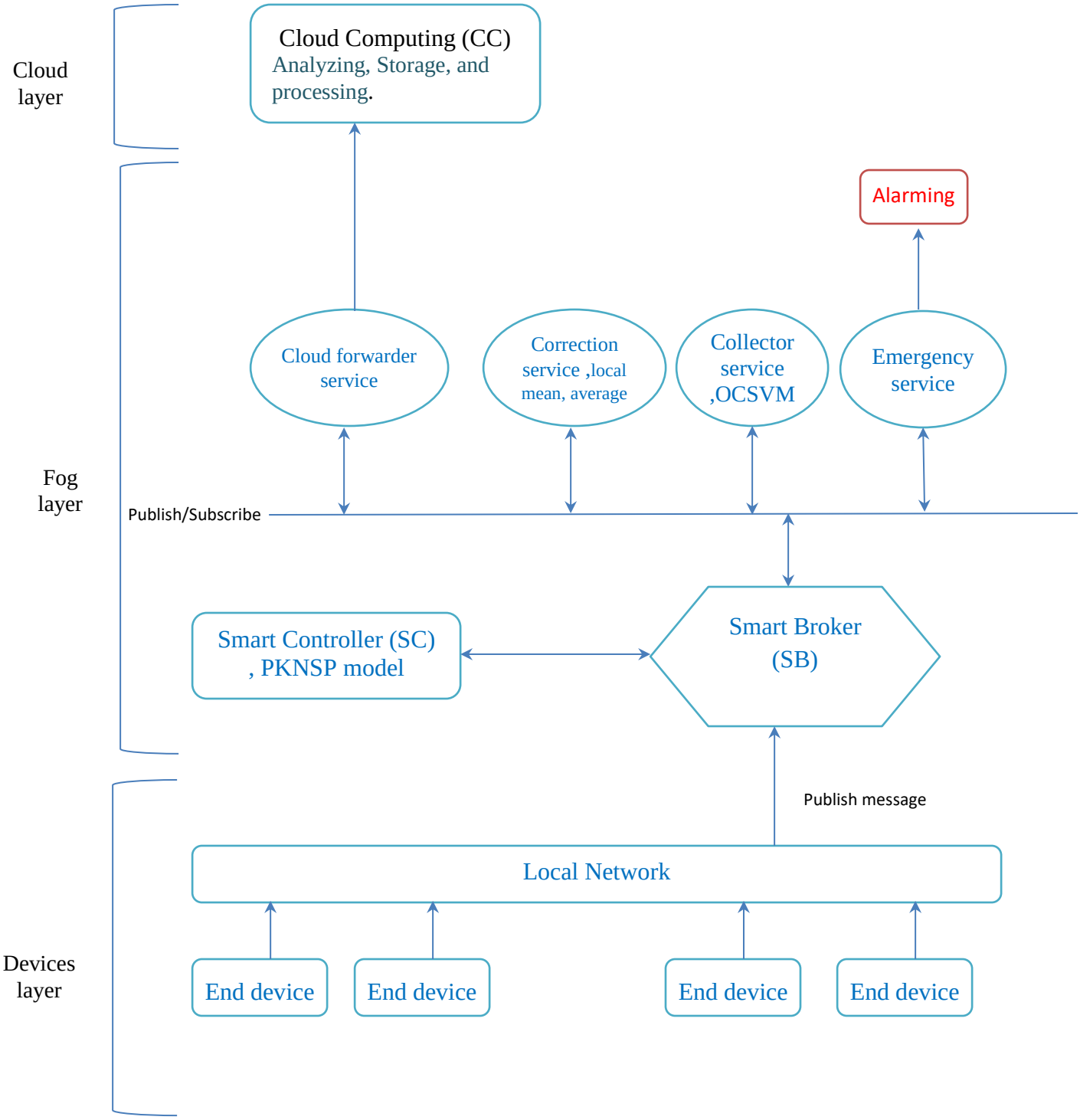


Figure (3.1) : Proposed Architecture Diagram

3.2.1. Devices Layer

The devices can be sensors, actuators, and devices that publish messages to the SB.

3.2.1.1. End devices

Edge devices, for example, various sensors, generate and publish data across the SB under a specific topic. Messages are connected via a reliable and lightweight communication protocol, such as the MQTT protocol. In an IoT system, several sensors should be installed (four sensors temperature, light, humidity, and voltage.) to monitor various situations. The missing values and outliers, which are widely considered useless, are present in the raw sensor data obtained from sensors. To extract pertinent information from the sensor data, data preprocessing must be done on the raw sensor data before data analysis can be done. Furthermore, in a limited IoT sensor network, an abundance of unwanted and worthless data might result in high processing costs and unnecessary resource utilization.

3.2.1.2. Local network

Communication protocols are popularly used for the information exchange process. These communication protocols work to ease the flexibility and scalability of sensor deployments. It makes use of the publish-subscribe paradigm, which is essential to an Internet of Things system since it allows several services and devices to exchange vast amounts of data with one another. This makes the model appropriate for IoT systems, which are mostly made up of numerous linked devices. One benefit of publish-subscribe is that, because publishers and consumers are separate entities, systems do not crash in the event that one of them experiences an outage.

3.2.2. Fog Layer

This layer receives messages published from the devices layer via the communication protocol and sends them either to the cloud or an alarm for the administrator about the presence of a critical situation that is a priority.

3.2.2.1. Smart Broker (SB)

The publish-subscribe approach serves as the foundation for the lightweight communications protocol known MQTT. The client (like an IoT device) connects to the server (also known as the MQTT Broker) in MQTT's client-server architecture and publishes messages under designated topics. The message is forwarded by the broker to the customers who have topics subscribed in the same topic. It works effectively in circumstances when there are few processing and memory resources available on the devices, and the network bandwidth is modest.

The SB's work is supported using machine learning algorithms in two places, in the services and in the making of decisions in SC, Some of the ML algorithms in the services within the fog node. In addition, the use of the SC in the fog node helps making dynamic decisions in efficient time without the need for human intervention. whereas the availability of the human element is not guaranteed and requires many resources to present and audit the details of the data in order to make decisions in the event of an increase in the congestion on services.

The SB notifies the services of the existence of data on a specific topic when it becomes available for publishing. Thus, the services can receive the data for which they have subscribed. An SB between IoT devices and services will allow it to make efficient decisions to distribute the data according to the published topics for the services in which it is interested without data redundancy. The services that have subscribed to the SB receive the data stream for preprocessing, such as collecting the data and verifying it for any problems, then data will be cleaned and

corrected so that it is available without problems to applications and users who request those data.

One of the important capabilities of this SB is its ability to distribute data streams to more than one subscribed service while ensuring that the data is not duplicated in other subscribers' services. Also, it's ensuring that the system does not fail or stop in the event of a service failure or it is not connected to the Internet, as is the case in edge or fog servers. In addition to being inside the fog node, it has strong features in processors, storage, an abundance of resources, and proximity to the data source, which increases the continuity of data flow on services and interaction with the devices that generate that data. The services used and the SC, which will manage and direct the operations of adding and removing these services from the system, will be explained consecutively.

3.2.2.2. Smart Controller (SC)

In the proposed adaptive architecture, SC is unique and typically implemented as a software application component in the fog node. It monitors the overall system performance, which will be explained in section 3.3 by the proposed PKNSP model and combined with an SB, leads to the following benefits:

1. Manage IoT services more efficiently.
2. Dynamic deployment of the services based on the published topics on the SB.
3. Scalability of the architecture without the need to reconfigure it manually and individually.
4. Automate and fast decision-making in the efficient time, avoid human error.
5. Improving the reliability and flexibility of the system in terms of high response time, high throughput and low latency.

6. Optimizes the use of resources and the information that covers the work of all the services currently implemented.

When building the proposed model of the work, it appears a challenge represented by how to deploy initial services in the IoT system, and there are several possibilities that followed:

The First scenario: Initial Run of Several Services is starts by deploying a set of services (for example, 10), and they are stored in a matrix containing a counter starting from (1...10) inside the SC, with the service Status Indicator (SIN) field (0 not executed and 1 executed). By using a process to check the numbers of the currently executed services, the SC sends a command message to remove a service from the system with a status indicator in the array changing from (1 to 0), as in Table (3.1). Services start with a SIN (1) and the number of deploying services (10). When a decision is made to remove a service, the SC changes the SIN to (0), announces the number of services from the (Number of Services) field, and stores it in a variable of the service counter (Counter). This is how the system continues to work. The topic name or type topic means the data published under a certain name and refers to the IoT that published data, such as (Temperature, humidity, etc.). This case is not feasible in the proposed work because the dynamic and heterogeneous nature of the IoT will not be achieved with the consumption of system resources by deploying a static number of services (10).

Table (3.1): Initial Run of Several Services

Topic Name	SIN	The initial state (1)	Change SIN	Number of service
Temperature	1	*Remove*	0	1
Temperature	1	*Remove*	0	2
:	1	*Remove*	0	:
:	1	*Still deploying*		:
Temperature	1	*Still deploying*		10

The second scenario: starts with a number of services that are not activated in the idle stage, which is the operation of a group of services with a condition

within the service to work upon receiving a status message from the SC (true or false). Here, the SC does not specify the number of the service that will be activated but rather sends a status indicator through the broker. Thus, any service that receives the message checks the status indicator to see if it is true and is executing the tasks in it. If the indicator is false, it will enter the idle phase. As for the SC, there is a counter for the number of services currently being implemented, and it changes according to the system's needs (adding or deleting). This case is lacked a dynamic and heterogeneous solution and consumes resources in the system.

3.2.2.3. Run Services Dynamically in the Proposed System

At the beginning of work, the SC and emergency service deploys. The emergency receives all messages that pass from SB and send from IoT, it publishes a message to the SC to deploy the service dynamically if the topic name is new (a service that handles a specific type of data, such as deploying a service for each of temperature, voltage, humidity, and light). The SC receives status messages from the IoT system with the subject to which the data is posted. It performs a test on the topic field to see if the topic currently received is new or not.

The message underlining certain topic name which stored in the screen table has a service currently deploying, and it starts to perform the tasks of the current publisher. After testing the topic, the SC stores the new topic, adds one to the topic counter, and finally sends a message to deploy the service interested in the new topic name.

The initial state is deploying a service with a topic name and then SC starts to deploy services sequentially according to new topic names. The service deploys in the system and starts to send information messages to the SC.

In each iteration, the SC verifies if there are new topics received. As illustrated in Figure (3.2), deploying new services is based on the topic name in the information messages that are received from services.

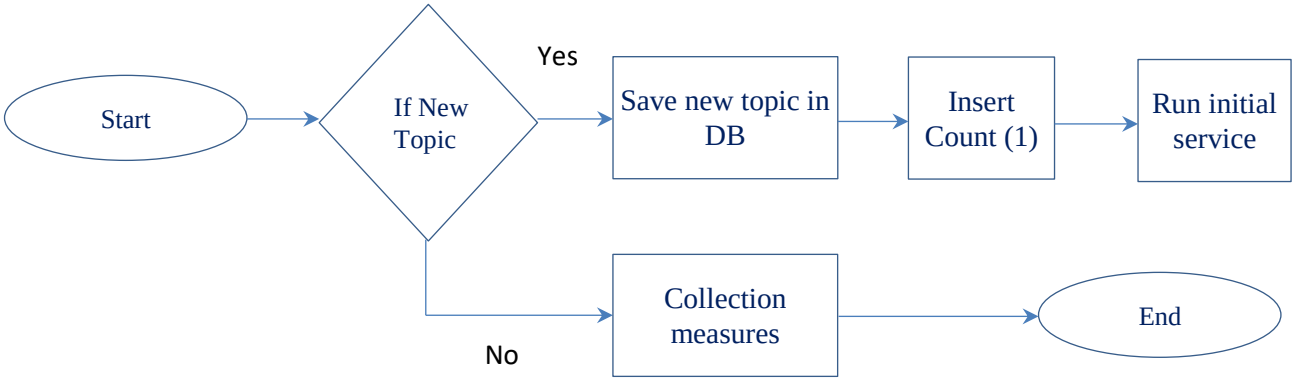


Figure (3.2): Deploy Service dynamically

In this case, it has a dynamic deploying of system services, and this reduces the consumption of system resources, as there is no deploying service if there is a published topic for, and it needs processors. In addition, it can deal with heterogeneous data from the IoT by knowing the SC of the types of services that perform the processing on the published data according to a pre-defined topic through the system administrator (such as a service for each data published under the topics of temperature, voltage, humidity, and light). Therefore, when adding a new publisher to a topic defined by the SC, there is no need to change the system settings or services.

3.2.2.4. Smart Decisions Making

There are two types of decisions that the SC can make: either deployment or stop the service, with the help of a controlling algorithm that is based on the measures which are sent from each currently deploying service.

This information is collected in a screen table, which stores all of the transmitted details that the algorithm will use to generate appropriate decisions to operate the current system with high efficiency.

The addition of service, which is represented by (spawn a service), depends on the continuous improvement of measures such as the latency, response time, data loss, throughput, and data rate, a new service is added of the system, and increment the services counter by one (++1). In contrast to the removal of a service, which is represented by killing a service, if there is an increase in the measures from the previous reading, a service is removed and the service counter is subtracted of one (--1). The system arrives at the starvation (the number of services more than the system needs). Throughput is enhanced with the increase in the number of services that process the current data stream.

The screen table consists of the set of details sent from the currently active service, as shown in Table (3.2). The first column (Timestamp (ts)) represents the times of receiving the details from the currently active services, which will be referred to it as (ts), which starts from (ts0, ts1, ts2,... tsn). The screen table contains of the service name (Temp), along with latency values, throughput, data loss, and the number of services. The SC publishes control messages on the SB, which represent the decisions from the SC algorithm. One can spawn services or kill them based on the decision of the SC algorithm.

Table (3.2): Screen Table Example

Timestamp (ts)	ID-Service	Numbers of services	Latency(s)	Throughput J/S	Data lost percentage
0	T1	1-Service	6.76	2.96	0.88
1	T1	2-Services	5.98	3.34	0.57
2	T1-inst	2-Services	4.83	4.14	0.46
3	T1	3-Services	3.18	6.28	0.14
4	T2-inst	3-Services	3.02	6.64	0.46
5	T1-inst	3-Services	3.31	6.63	0.57
6	T1	3-Services	3.21	6.62	0.46
7	T3-inst	4-Services	4.01	4.99	0.14

The T1-inst in the column ID_Services represent the new services add to the architure by SC's decision.

The Figure (3.3) shows an architecture for SC and there is at least one service of a certain type currently deploying and processing data semantically. There are several services under the same topic name of currently deploying service, and it is normal for them to be stopping and awaiting the decision of the SC in order to enter the service and operate with the original service.

The ***decision taken*** from the SC algorithm based on the information measures receive from currently service that data preprocessing. The data will be published on the SB under a predefined topic heading for this purpose with a status indicator, positive to add a service or negative to stop a service from the architecture in the currently running system. The decision is made by applying the proposed model (PKNSP) to measures that collected in the screen table.

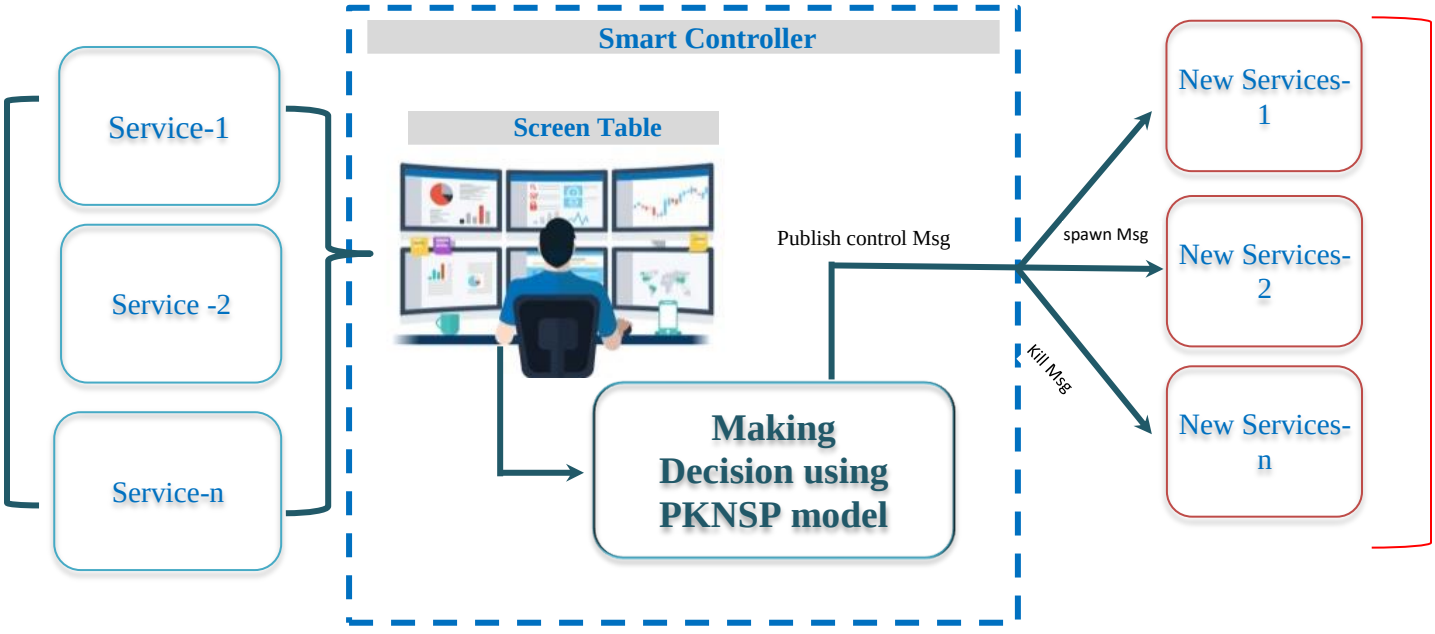


Figure (3.3): Architecture for a SC in the Fog Layer

3.2.2.5. SC Behaviour

The SC behaviour makes decisions based on the evaluation of performance measures according to Algorithm (3.1). where the algorithm begins to receive input from the screen table in the SC. Then it determines the size of the window in which the algorithm will collect data in the form of jobs of data (ΔT). In our experiment, the value of ΔT equal to 20 is chosen, where four values (5, 10, 20, and 30) were tested, and the stability of latency and throughput was observed at this value more than at the previous values. Then it extracts the amount of latency and throughput for the current window and sends it to the decision-making step. The SC algorithm evaluates the performance by reading the metrics collected in the screen table, which is the backbone of the SC's decisions. The algorithm computes average latency, response time, throughput, and the difference between the current number of deploying services and the number of services that predict it by using the proposed PKNSP model. The output of the ***function predicts number of services*** is an integer number of services based on the inputs, which are latency, throughput, execute time, and train data that load from the proposed PKNSP model. The steps of the number of services prediction function are as follows:

1. Reading parameters (latency, throughput, execute time, and train data that load).
2. Determining the samples (X) and targets (y) from the training dataset with the best K (best_K).
3. Applying the normalization and PCA on the parameters (latency, throughput, execute time) for standardized values with training data.
4. Applying the KNNR ML on the (X,y) with (best_K) , then predict the number of services.
5. Returning the number of services.

Algorithm (3.1): SC Algorithm

Input: Data collected in the screen table

Output: decision add service(spawn) or remove service (kill)

Begin

1. ***Counter = 1 // the number of active services***
2. ***ΔT = 20 // number of recorded in screen table***
3. ***MaxData ; MinData // the max and min numbers of messages in the services***
4. ***Msg-Counter = 0 //The number of blocks in the SC***
5. ***Latency = 0.1***
6. ***Respo=0.1***
7. ***Execut=0.1***
8. ***Throughput = 0.1***
9. ***train data = load PKNSP model***
10. ***$\mathscr{E}$ = 0 // predict the numbers of service based on PKNSP***
11. ***Diff = 0 // The difference between the current number of service and $\mathscr{E}$***
12. ***While connection = true***
13. ***Deployment Services Based on the Topic name(raw data, deploy services)***
14. ***Start-Time = current time // The start time of the collecting metrics***
15. ***While (Msg -counter <= ΔT)***
16. ***Block-message = a block receive from the service***
17. ***Latency = + Block-message[‘Latency’]***
18. ***Respo = + Block-message[‘Response’]***
19. ***Execut = + Block-message[‘Execute’]***
20. ***MaxData = max[Block-message[‘MaxID-Msg’]]***
21. ***MinData = min [Block-message[‘MinID-Msg’]]***
22. ***Ms -counter ++***
23. ***End***
24. ***Msg-counter = 0***
25. ***Total-Time = current time - Start-Time***
26. ***Total-Data= MaxData - Min-Data***

```
27.         Process-Data = ΔT * size-Block(size bloks in the services)
28.         Data-Loss   =  Total-Data – Process-Data
29.         Data-rate    =  Total-Data  / Total-Time
30.         Av_Latency =   Latency / ΔT
31.         Av_respo =   respo/ΔT
32.         Av_execut =     execut  /ΔT
33.         Throughput  =  ΔT  / Total-time
34.         & =  Function predict number of services( Av_Latency , Throughput  ,
                                                    Av_execut ,train data)
35.         diff   =   Counter - &
36.         While (diff >0)
37.             Add-service (spawn, Topic name)
38.             Counter   ++
39.         While (diff < 0)
40.             Remove one service (kill, Topic name)
41.             Counter   --
42.         End
43.         Update screen table
44.     End
End
```

Furthermore, the SC has been subscribed to an SB based on status message topics for each service in the fog node (ensuring the continuity of data flow so that the system does not fail when one of the services fails and receives a complete data stream). The behavior of the general algorithm of the SC from the beginning of receiving the data to making the final decision has been illustrated in Figure (3.4) for one topic.

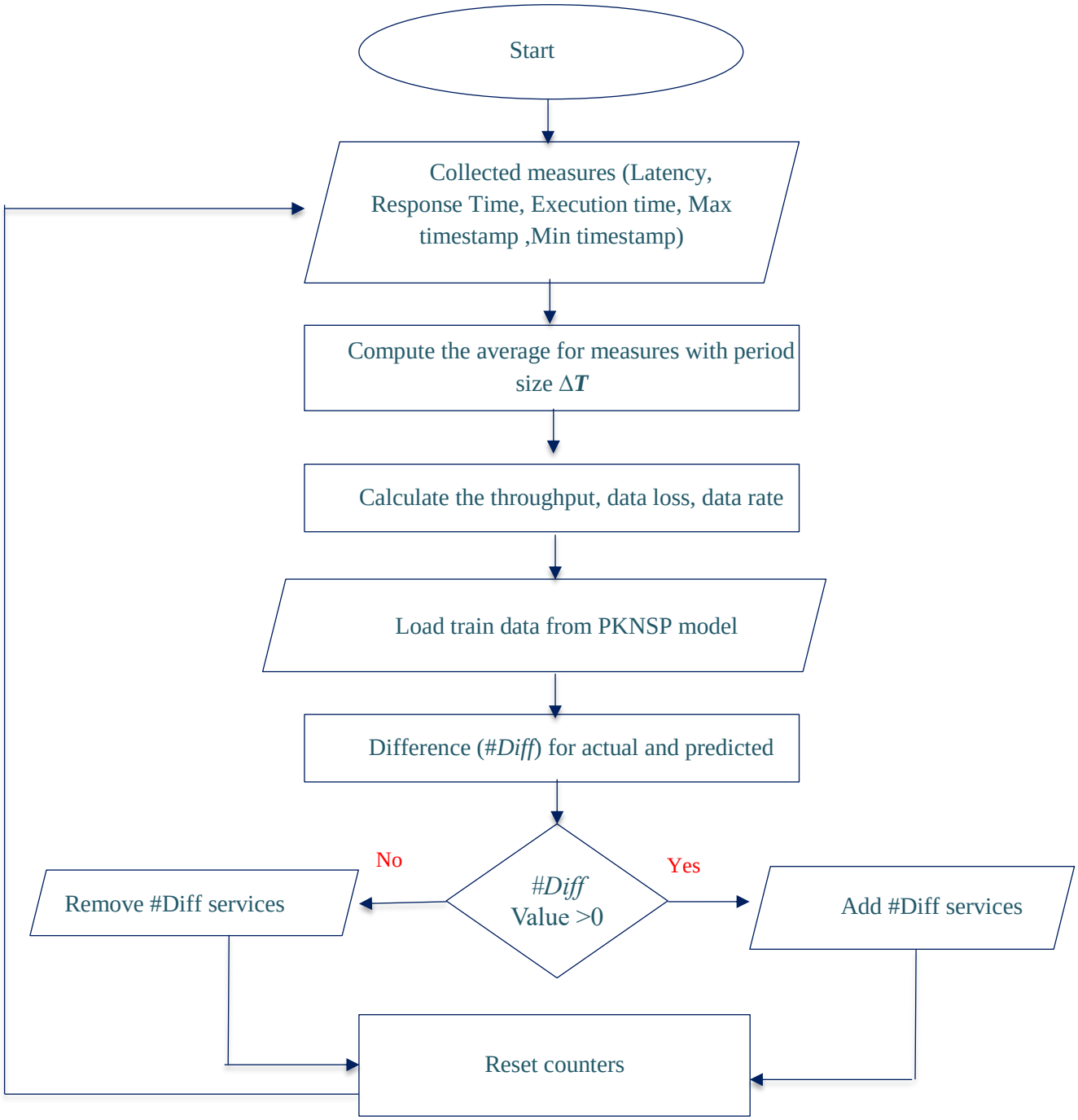


Figure (3.4): Flow Chart of SC Behavior for Each Topic

3.2.2.6. Collector Service

The data received by the services that are subscribed to the SB is based on a specific topic. One of these services is the data collector and detector. For each IoT device, there is a specific collector subscribed to their specific topic name or global one. It collects a series of readings into blocks. Then, data is passed to the part of the system that performs anomaly detection for the purpose of knowing whether there are anomalies or not.

This service republishes the data block to the broker with a new topic (normal or abnormal). The One class SVM (OCSVM) algorithm is used for the detection of anomalies on each block. Before applying the algorithm, the data is normalized to avoid different scalings of the data vector.

The process of normalizing the data and detecting anomalies is run together in the collector service. One of the ML algorithms used is OCSVM (The results of comparing the algorithm with a set of algorithms are discussed in section 4.7) because of its wide reputation, speed, and strength in performance. The result obtained from the collector service is a list of values containing anomaly indices of the original value blocks. The output from the collector service is either normal data (doesn't contain missing values and anomalies) or abnormal data. Maybe the collector service publishes abnormal data to the correction service if there are separate services (detection and correction). The Figure (3.5) shows the data flow in the collector service.

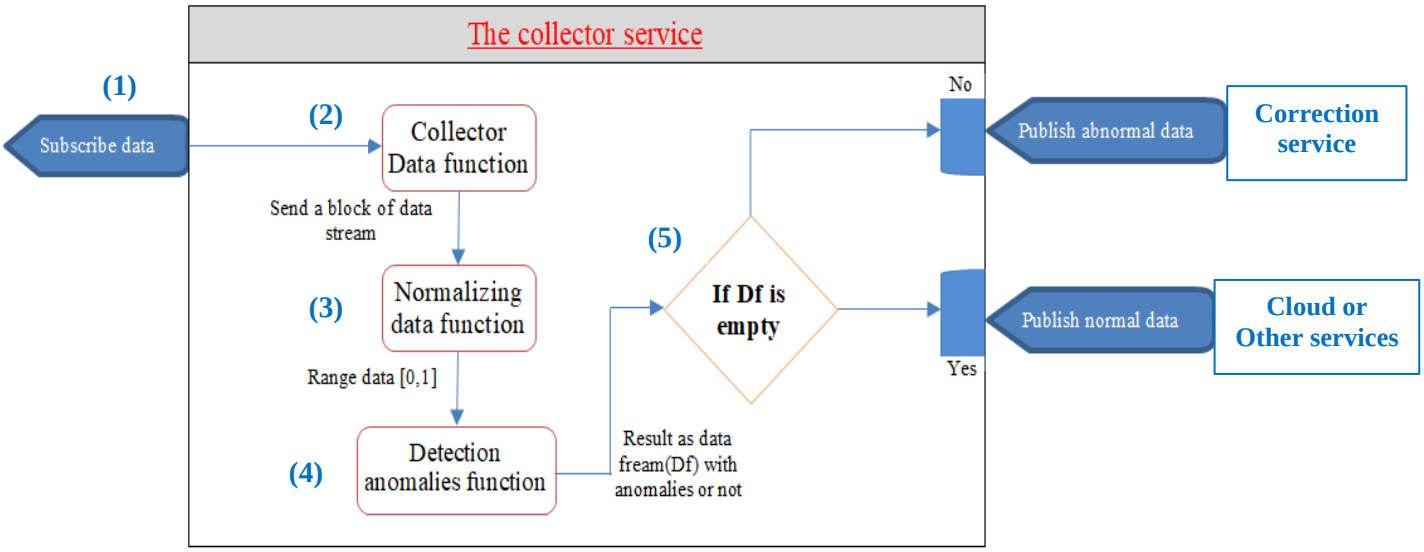


Figure (3.5): The Data Flow for Collector Services

The resulting list with the index of anomaly values is re-published to the broker under the topic of "abnormal" to have the correction service and take action on the data block. In the proposed architecture, a given service can be replicated several times and subscribed to a given topic with the broker for doing the same job but on different data blocks. Hence, a group of services (Collector and Detector [1, 2, 3..., n]) can act on the same topic to enhance the system's performance when enormous amounts of data are supplied for preparation. The capabilities of FC to process data locally with low latency and high response time and reduce data loss. A fog node hosts the services SC and SB.

As illustrated in Figure (3.6), a service has numerous functions; each work on a distinct data block deploys the same algorithm and can be executed anywhere. Examples of services are anomaly detection, missing data, and data correction. The service may contain all of them.

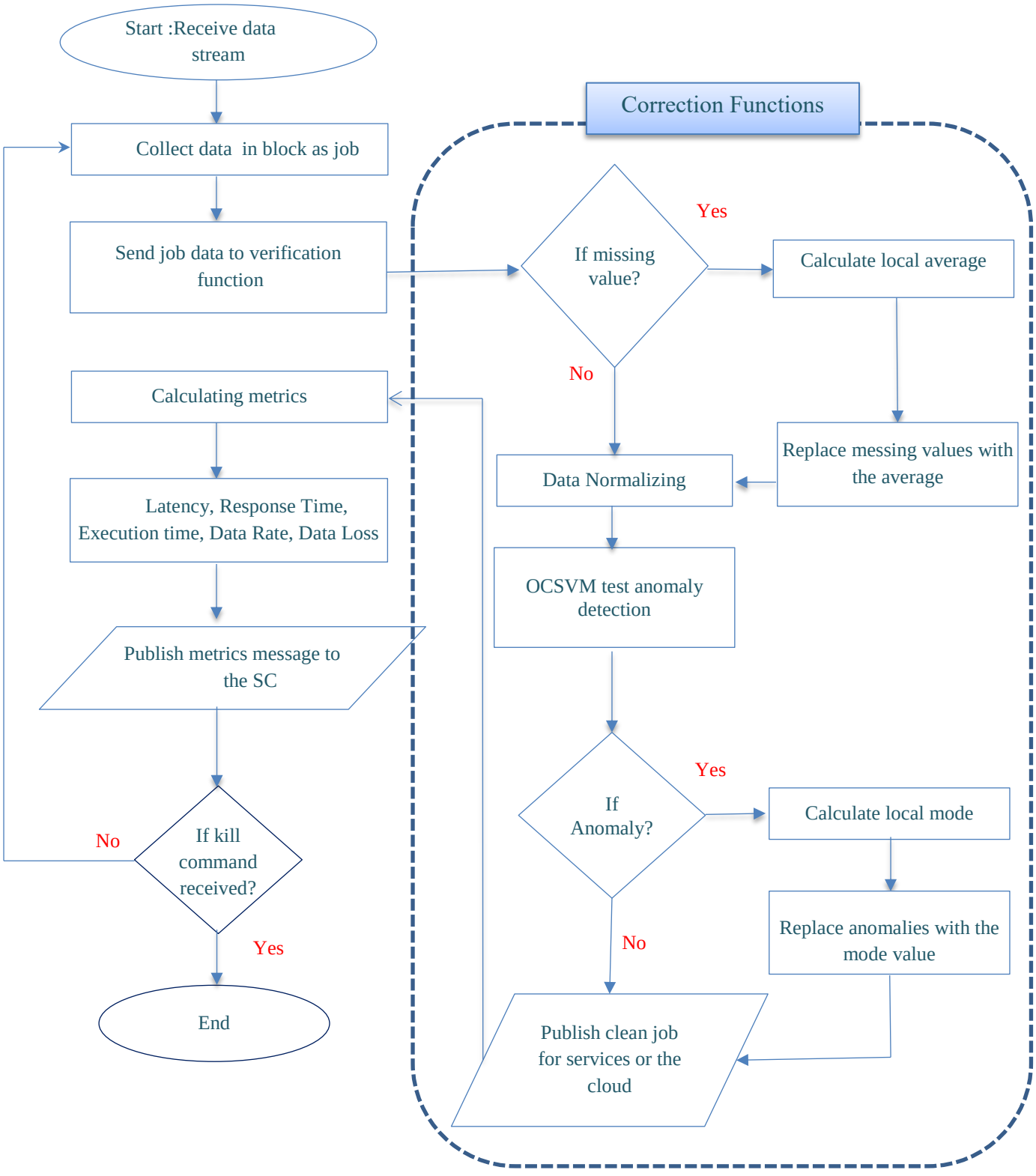


Figure (3.6): Flow chart of Data Processing in the Collector Service

3.2.2.7. Correction Service

The service can be either a function within a collector service or it can be a separate service that corrects the data containing anomalies or missing values. The output will be published as clean data. Tow techniques are used for correcting anomalies and missing values, such as the **local average** or **local mean**, and replacing them with abnormal values. The correction process on the missing values is done by calculating a local average, whereas anomalies are done by the local mean.

The correction of *missing values* by calculating the average of nearby data points around each missing value is called the local average. The following general strategy steps can be used:

1. Identify Missing Values: Find the locations of the data block's missing values first. The placeholder value NaN (Not-a-Number) is frequently used to represent missing values.
2. Determine the nearby data points: Decide how many neighboring data points you want to consider when calculating the local average. This will depend on the window size (size data block).
3. Calculate Local Average: For each missing value, calculate the local average by averaging the values of data points in the block.
4. Replace Missing Values: Replace the missing value with the calculated local average.

The local mean for *anomaly values* typically involves filtering the data to reduce noise and highlight underlying trends. Here, A block means is used to replace abnormal values. Finally, normal data is republished to the SB with the new topic “normal”.

The local mean for anomaly values can be calculated in the following steps:

1. Choose a Window Size: Decide on the size of the window or neighborhood around each data point within which you want to calculate the local mean. The window size determines the number of data points that will be included in the calculation.
2. Define the Anomaly Values: Identify the dataset containing the anomaly values you want to analyze.
3. Calculate the Local Mean:
    i. For each data point in the block (starting from the first data point), calculate the mean of the values within the chosen window size around that data point.
    ii. Move the window to the next data point and recalculate the mean.
    iii. Continue this process until to calculate the local mean for each data point in the block.

3.2.2.8. Emergency Service

The emergency service in the system has the ability to send alarms based on topics priority specific by the system's administrator according to Algorithm (3.2). where it can monitor all the data flow in the system and compare critical topics that are saved in it. It can decide to inform the system of a critical situation that needs immediate treatment. It collects all critical cases according to the critical topic and verifies that the critical case is repeated more than once (let it be five times). If the times are close, the case needs to be declared in the system and send the **alarm** and vice versa, meaning the times are spaced and scattered due to a malfunction in the device or an interruption of the Internet. This system also checks if data of a topic is received for the first time, by examining the membership of the topic of every

received message to a determined set of active topics.

The topic given does not belong to that set, the emergency service publishes to the SC to deploy initial service for a given topic. From analyzing Algorithm (3.2), the input of the algorithm is a set of messages published from data stream generated by IoT over SB, and the output is a decision to send an emergency message, which means indicates that the current data has priority in the processing and making a decision.

The collection of the received data begins with verifying for each message that the priority topic has not exceeded the limitation of threshold, with no repetition more than once during the time period specified by the system administrator or determined by the nature of the IoT devices.

When the priority cases are grouped into a temporary table, the table consists of (the topic name, value, time, and a number of repeats of the value). Every time the value and the topic are checked. The new data is stored in the table (Data-Coll). It is verified that the schedule data for the current topic is equal to the threshold for each value with range, and apply the action, a message is sent to the SC that there is a need to process the current data as soon as possible. On the contrary, the processors are proceeding normally.

Algorithm (3.2): Emergency Algorithm

	Input : Message from data stream
	Output : The decision to alarm
	Begin
1.	***Topic-name = string // Topic name from message.topic.***
2.	***Value = 0 // data value from message.value.***
3.	***Check table // List of tuples per topic, each tuple determined (range, action) .***
4.	***Data-Coll // Table to store topics, value, time, and topic repeat.***
5.	***Time-max // The maximum time allowed to repeat values.***

```
6.  While connection = True ()
7.      Create data block ()
8.      if Check table ['Topic-name'] is new then
            Publish to SC (Topic-name)
9.      Endif
10.     if Check table ['range'] <= Value then
11.         Action      =  Check table ['action']
12.         Data-Coll  =  insert  (Topic-name, value, time())
13.         Time-Sum =   Data-Coll ['time']
14.         If  Time-Sum >=Time-max then
15.             Publish-emergency (Alarm )
16.             Update Data-Coll
17.         Endif
18.     Endif
19.     Return (Alarm)
    End
```

3.2.2.9. Cloud service

The cloud service is used that contains the final addresses to store data or cloud services. This service receives the final data free of anomalies and missing values and stores. Cloud data centers may have multiple servers, and services can be deployed across various nodes for redundancy, scalability, and performance. The cloud data center or services are often accessed in a cloud environment via designated endpoints or URLs. To send or receive data, fog nodes which are usually located closer to end users or data generating devices can connect to the cloud data center by employing the proper addressing and communication protocols. Fog nodes communicate with the cloud data center using the standard MQTT protocol.

3.2.3. Cloud Layer

This layer receives messages published from the fog layer and storage or analysis. When using cloud computing, all data that is finished by the end device at the network's edge is delivered straight to the data center. The cloud data center is the final data center for most data streams after the preprocessing operations or the final results of data analysis in the fog. In the proposed architecture. A data center store can collect all data after cleaning it so that it is ready for use again by users interested in this type of data. In other words, the final data is stored, processed, and analyzed.

3.3. Implementation The Proposed Architecture

In this section, we will explain the implementation processes on the components of the proposed architecture, mentioning the layers in which they are implemented.

3.3.1. Implementation Publisher Data

Data is pulled from the database after specifying column data such as (temperature, humidity, voltage, and light) as messages in JSON format. The data messages are published on the SB according to the type of topic specified in the IoT device configuration by a user in order for that to be available to the services that are interested in the type of sensors published on the broker. For example, the format of data messages looks like:

1. *{"Topic name": "Temp", "ID-sensor": 4, "value": "18.4792", "Timestamp": "02/11/2022 11:06:39.306390"}*
2. *{"Topic name": "Co_2", "ID-sensor": 1, "value": 201.835, " Timestamp ": "02/11/2022 09:08:57"}*

The steps of publish data from datasets to services over the broker are :

1. Establish connection with the broker (IP, Port).
2. Read data from datasets as a IoT devices.
3. Create JSON (format 1,2) from values that read from datasets columns.
4. Create session on the broker open one time connection with broker A logical link that permits message exchange between publishers (entities that send messages), and subscribers (entities that receive messages), is sometimes referred to as "creating a session".
5. Publish JSON (with given Topic name)
6. End session()

3.3.2. Implementation SC

The SC collects details for each job of data that is being executed within the currently executing service, including the average latency and average executing time. where the SC monitors these measures values with high accuracy and then makes a decision in the case of increasing the overload on the service, which is currently being implemented by deploying the service automatically. The user intervention to reduce the latency in this service does not exist. The SC aggregates performance information (latency, response time, and data loss) from active services periodically.

The screen table is used to save these measures and update them periodically. The SC calculates the average latency, data loss, data rate, and throughput for the given period based on the topic type. The decision is made by using the PCA and KNN model with beat K and predicting the best number of services. Moreover, calculate the difference (***diff***) between the number of actual services deploy currently and the number of services predicted from using the

model. If the diff is positive, the SC will publish a message of the deploy (spawn) for the instance of the service; otherwise, it will stop (kill) service after finishing the preprocess.

3.3.3. Implementation SC Algorithm

The system starts with a service of a certain type deploying automatically, in addition to the spawning indicator being positive (True) with one number for the services counter (counter=1). As in Figure (3.7), which explained SC Algorithm action. The stages consist:

6. Receiving the metrics values from the services for each job on which the preprocessing operations are carried out.
7. Collecting the values of the metrics according to the size of the period (ΔT) requires collecting job information, which includes the latency and the execution time for each job received from the services.
8. Calculating the average for the measurements received during the specified period (ΔT), where the average latency and throughput are calculated.
9. Define an initial value for the decision pointer (Spawn) where it is either added service (spawn) and the pointer is equal to (True) or stop(kill) and the pointer is equal to (False).
10. If the pointer is equal to (True), the SC publishes a message to add a non-deploying service instance to the architecture, otherwise delete a currently deploying service instance.
11. Receiving the values of the measures from the services.
12. Collecting the values of the measures according to a specific period.
13. Calculation of the average for measures received during a specified period
14. Using PCA and KNN model for predict.

15. Calculate the difference (Diff) between current and predict services.
16. If the Diff is positive, the status indicator is equal to (True), otherwise, the indicator is equal to (False).
17. Set the previous average latency value equal to the current one.
18. Continue checking the values of the decision pointer (spawn).

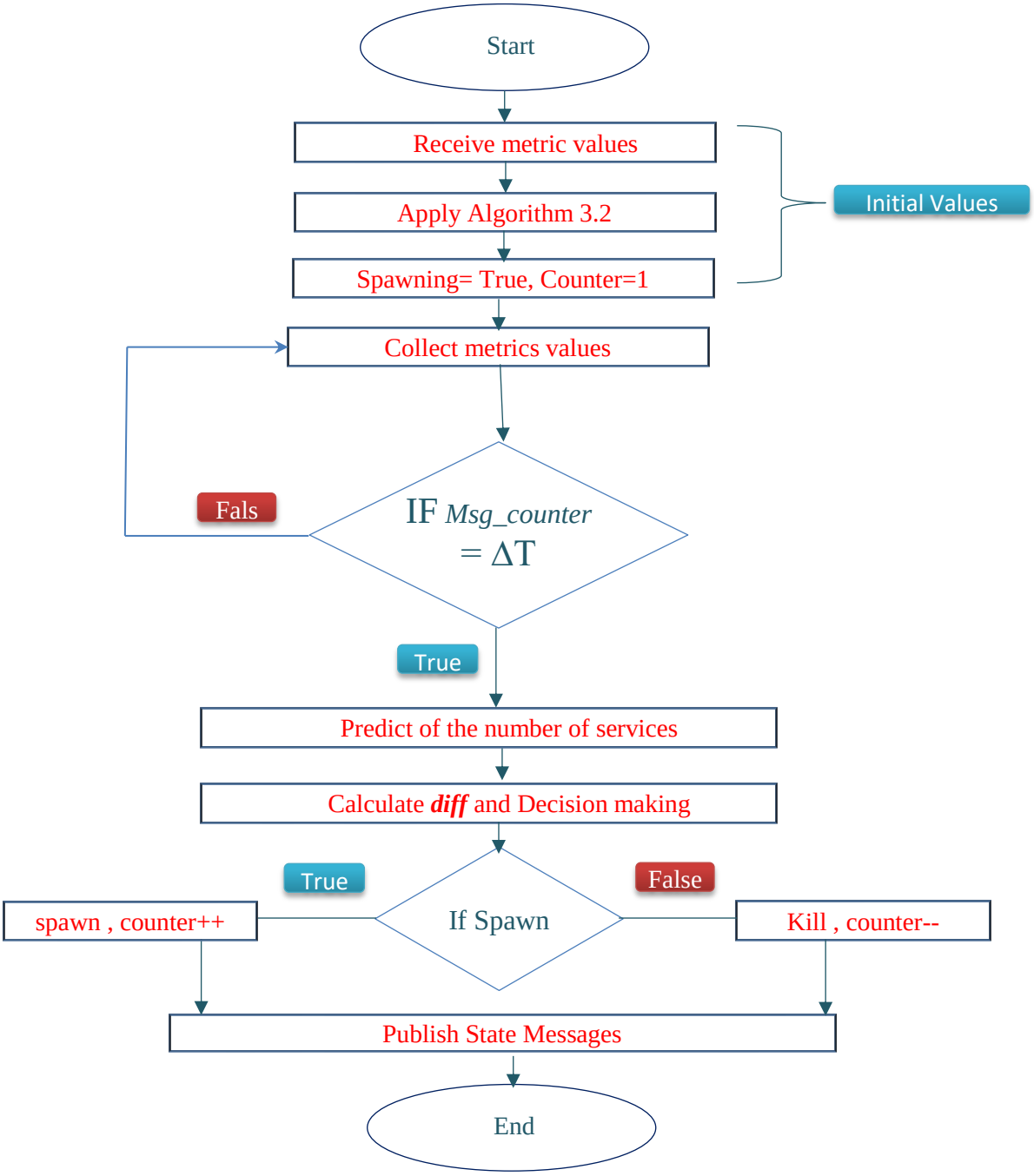


Figure (3.5): Spawn and Kill Actions

3.3.4. Implementation Data Processing Services

An IoT system provides services and applications to meet user requirements, such as health care services, sleep care, weather forecasts, etc. In this dissertation, a set of services that work in the suggested adaptive architecture have been built (collector, correction, emergency, and cloud services). These services can be functioned within one service to save resources and increase the rapid preprocessors within fog nodes. In general, ML algorithms are used to extract knowledge from raw data that is generated from IoT devices or any other sources. In the services, ML is used in the preprocessing of the data stream that is published from IoT devices over the SB. ML provides the quickness and accuracy of the data process within services, and extracts results that are used to make decisions in the SC algorithm. These services are subscribed to the SB inside the fog node. The number and type of these services vary according to the required data quantities and preprocessing.

The service is a standalone component which is capable of managing specific activities and functions. A service uses ML algorithms to preprocess data, and it has subscribed to the SB according to the interesting topics. Inside the services, the data is received, collected, preprocessed, and republished cleanly back to the SB to benefit other services, like the cloud or applications.

The system services receive a data type that interests them, which is published on the SB by IoT devices. On the other hand, they have subscribed to an SB according to the topic type that correlates with certain a sensor, they are ready for data process from faults and problems.

The data are re-published data as messages on the SB in two topics:

i) As a block of clean messages to other services, the cloud, or applications requesting the data and

ii) As status messages to the controller.

The active services contain a set of functions of interest to the user and process data in an efficient time. The missing values and anomaly detection are important issues facing the processing of IoT data streams and lead to the degradation of the system performance or data loss. The Figure (3.8) shows the data flow for a service and the functions in the service for increased accuracy and rapid processing.

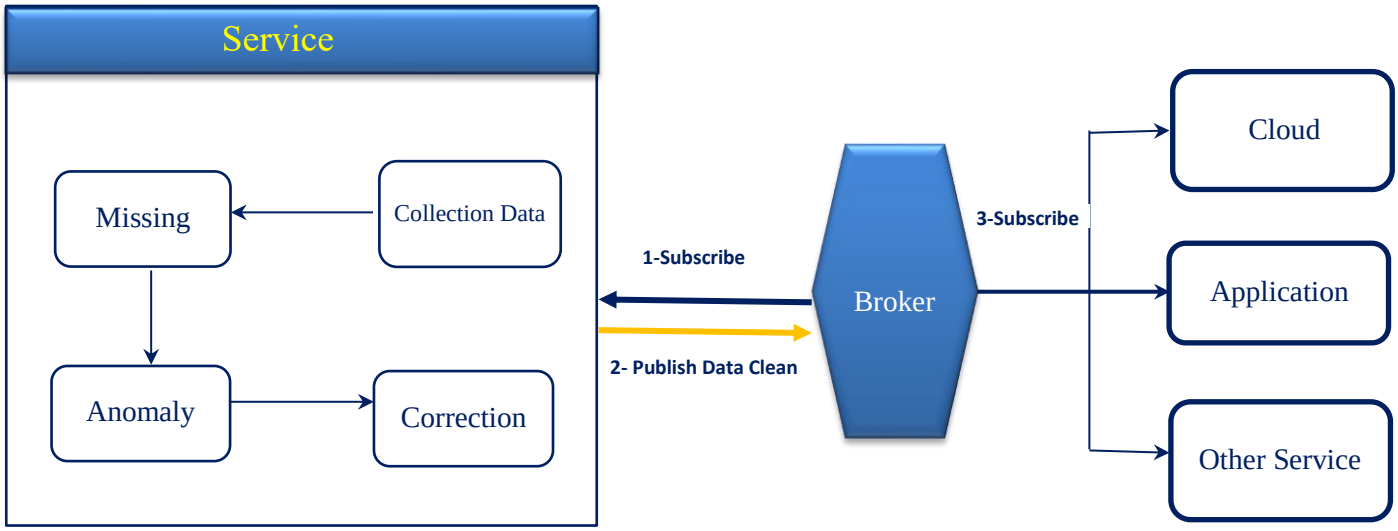


Figure (3.8): Action for a Particular Service

3.3.5. OCSVM for Anomaly Detection

The OCSVM algorithm evaluates parameters (gamma and nu) based on the concept of cross-validation, as it employs an anomaly detection model on data blocks. Using the OCSVM, this proposed method assigns the best parameters to the algorithm according to its contribution to increasing accuracy.

This is achieved as follows:

First, apply the OCSVM to all values in the column and keep the anomaly values in the table based on the parameter value. Then the accuracy of the model is calculated for all parameters in the table by distributing values on the TP, TN, FP, and FN coefficients. Finally, apply the equation (2.14) that computes the accuracy from coefficients. Algorithm (3.3) which shows the steps of the OCSVM.

Algorithm (3.3): Anomaly Detection Algorithm

Input : *Dataset based on topic name*

Output : *The best parameters*

Begin

1. ***DB*** = *dataset*
2. ***Split DB x_trian, x_test, y_trian, y_test***
3. ***Nu*** : *[0.02, 0.02,0.001,0.002,0.01,0.02,0.02], # Control the number of anomalies*
4. ***Kernel*** : *['Rbf'], # The kernel function to be used*
5. ***Gamma*** : *[0.01,0.001,0.001,0.001,0.001,0.002,0.003]*
6. ***For i=1 to increment***
7. ***OneClassSVM*** *(nu[i], kernel , gamma[i])*
8. ***y_Predict = OneClassSVM. Predict()***
9. ***DB_anomaly_values = where(y_Predict == anomaly value)***
10. ***Save DB_anomaly_values***
11. ***Distribte values on the coffecients(TP, TN, FP, and FN)***
12. ***Accuracy*** $= TP+TN \,/\, TP+TN+FP+FN$
13. ***End***
14. ***Return (Accuracy)***

End

3.4. Number of Services Prediction Model

The proposed ***PKNSP*** model focuses on the properties of the KNNR and is concerned with modifying its behavior to improve the prediction of number of services by integrating with PCA. The popular supervised machine learning technique KNN has been utilized extensively in a number of disciplines. KNN does not require prior knowledge of the distribution or parameters of the underlying dataset for classifying or predicting the value using the K closest data points to the test point in feature space and their labels or values. This makes it a powerful tool since, when working with complicated data structures, feature distributions are not always known beforehand. When data is presented in smaller amounts (blocks), the KNN performs better. The results of applying KNN the algorithm without PCA exist in section 4.6.

A supervised learning approach that can be utilized for regression applications is KNNR. By taking into account the values of its KNNR in the training dataset, it may predict the target value of a new sample. The KNNR algorithm is used in the SC to predict the best measures that lead to making decisions automatically in real time

Principal Component Analysis (PCA) is a dimensionality reduction technique that can be used to identify the most informative features in a dataset. It transforms the original features into a new set of uncorrelated variables called principal components.

The best features are obtained by applying the PCA algorithm to the dataset. The features are latency, throughput, execution time, data rate, data process, total data collected and data loss .The reducing model uses the split data to train and test. It is used for selecting the best (k), which is evaluated by MSE on the KNNR prediction model. Finally, keep the best splitting and K and save it in the PKNSP

as a model of results for integrating the PCA and KNNR. Algorithm (3.2) which shows the integrate the PCA and KNN.

The PCA can be used as a preprocessing step to reduce the dimensionality of data, then applied feature selection by using a cumulative explained variance threshold to the transformed data to select the most informative features. In other words, the feature selection is embedded in the PCA algorithm.

Feature selection involves choosing a subset of relevant features from the original set, pointing to retain the most informational ones. Dimensionality reduction, subjects to transform the data into a lower-dimensional space, capturing important information while reducing the overall number of features. In short, feature selection keeps some features and discards others, while dimensionality reduction creates new features from the original ones.

Algorithm (3.4): Proposed PKNSP Model

	Input: multi-features from DB.
	Output: The best model and K of neighbors
	Begin
1.	X_Data = The features without target
2.	Y_Data = The target
3.	X_Data_Norm = Apply Normalization on (X_Data)
4.	X_pca, n_components = Apply The PCA algorithm (X_Data_Norm)
5.	X_reduced = X_pca [:n_components]
6.	***Best_K*** *= 1*
7.	**Last_MSE = Max Double**
8.	***PKNSP*** = ***Split data (***X_reduced, Y_Data, test=0.2***)***
9.	***For K from * to increment***
10.	Apply KNNR model(***K***)
11.	MSE = mean_squared_error (KNNR predict)
12.	***If*** **Last_MSE <** ***MSE then***
13.	*Best_K* = ***K***

```
14.             Last_MSE  = MSE
15.         End if
16.         K  ++
17.     End
18.     Return ( PKNSP , Best_K)
End
```

Integrating PCA and KNN algorithms allows for effective features selection, handling of multicollinearity, improved generalization, and computational efficiency. By leveraging the strengths of both techniques, you can enhance the performance and efficiency of the regression model.

Unsupervised ML techniques like PCA aim to retain as much information as possible while reducing the dimensionality (number of features) of a dataset. To do this, a brand-new set of features known as components is found. These components are composites of the original features and are not associated with one another. Following the loading of the dataset, you can perform these steps to incorporate KNN and PCA for dimensionality reduction. As illustrated in Figure (3.9):

1. Starting by normalizing the data until unifying the data within a specific range.
2. Constructing the covariance matrix involves determining the relationship between pairs of variables and the amount of variance.
3. Computing the eigenvectors and eigenvalues.
4. Sorting the eigenvalues in descending order and selecting the top k eigenvectors corresponding to the highest eigenvalues.
5. Transforming the feature matrix using the selected principal components by multiplying the original feature matrix by the selected eigenvectors to obtain a reduced-dimensional feature matrix.
6. Splitting the dataset into training and testing sets by dividing the transformed

feature matrix and the target variable into a training set and a testing set of 80% of the data and the remaining for testing.

7. Train the KNN model. Fit the KNN model using the training data. Specifying the number of nearest neighbors (k) and any other relevant parameters, such as distance metrics.
8. Evaluate the model: Use the trained model to predict the testing data. Calculate the appropriate regression evaluation measure, such as mean squared error (MSE).
9. Iterate through different values of k (the number of principal components) to find the optimal number of features that provides the best regression performance.
10. Selecting the best-performing k and finalizing the model: Choose the value of k that yields the best regression performance based on the evaluation metrics. Train the final KNN model using this optimal number of principal components.

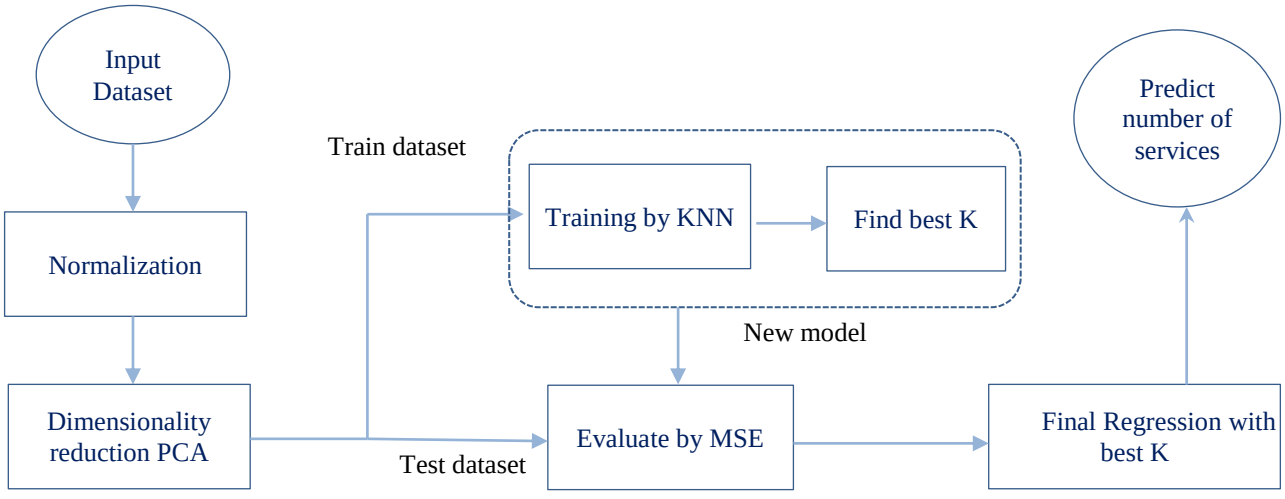


Figure (3.9): Proposed PKNSP model

The Figure (3.10) shows the evaluation of the PKNSP model with other MLs in predicting the number of services and breast cancer. The real data flows after the normalization as inputs to these MLs. The output is MES, accuracy, precision, recall, and f1-score, which is then compared with the result from the PKNSP model.

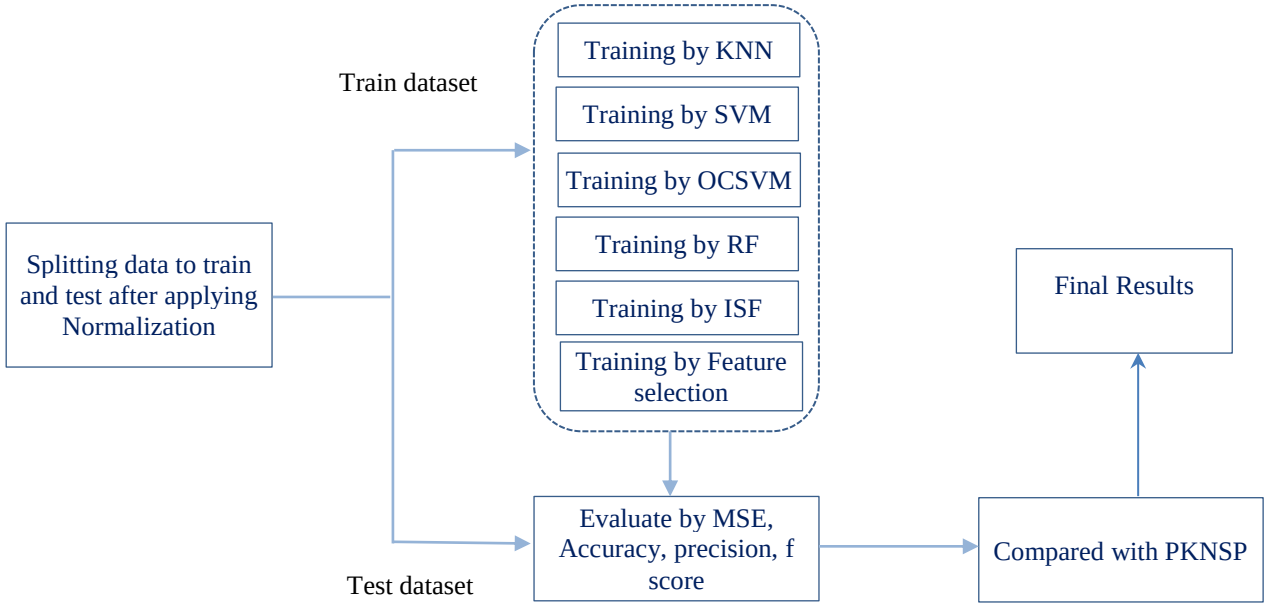


Figure (3.10): Evaluate Proposed PKNSP model with other MLs

The Figure (3.11) shows the construction of the model for the process of predicting the number of services required to deploy in the system. According to the current state of the system in terms of performance measures, those are the inputs of the PKNSP . The output is the best number of services.

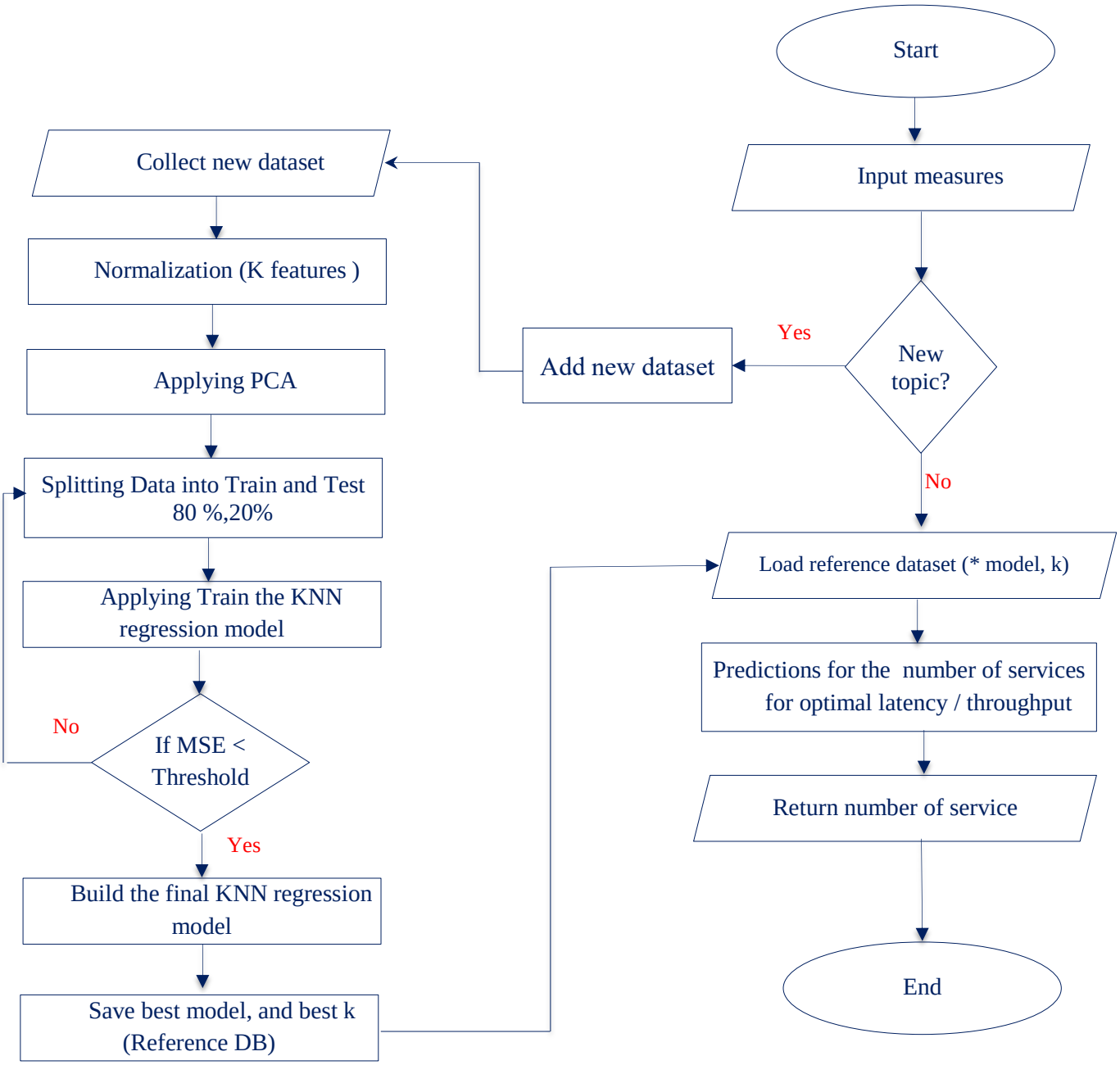


Figure (3.11): Flow Chart of PKNSP Model

3.5. Summary of the Chapter

This chapter discusses the main improvements in IoT system performance that can be obtained from implementing the proposed architecture and investing in the architecture of FC. One of the important advantages is running the system with simple resources. In the same context, integrating the SC and the broker in one architecture and within the fog node led to increased reliability, scalability, saving of resources, rapid processors, high response time, increased throughput, reduced latency, and reduced data loss.

The services which have a subscription in the SB and are organized according to pre-determined topics in the fog node, added to the system's abilities the monitor the performance measures that affect processing operations and data transfer between IoT devices and applications on the one hand and the cloud on the other. In addition, the SC algorithm has the ability to help make a dynamic decision in efficient time . The SC controls a set of services that have the same topic name for the currently active service that processes the data stream. These services can be functioned by the SC deciding to use them and incorporating them into the architecture of the currently operational system after applying the *PKNSP* model for predicting the best number of services based on the information messages.

There are two kinds of decisions obtained from applying the SC algorithm: i) spawning a new service in case the currently deploying service is overloaded. Thus, helping services that process the stream to improve response time, latency, and throughput for services that are currently active in the architecture. ii) Killing a service that is currently deploying, where the service is stopped after processing all the data it has, because the system now has enough processing services and good

standards, and there is no need to add another service, meaning that there is a long waiting time to receive data from the broker to start processing.

By integrating KNNR with PCA for dimensionality reduction, you can reduce the dimensionality of the dataset and improve the model's performance by focusing on the most informative features. Note that the optimal number of principal components (KP) and the number of nearest neighbors (***k***) in KNNR are separate parameters. The optimal KP for PCA may not be the same as the optimal ***k*** for KNNR, so it's important to perform this two-step process.

Integration principal component analysis (PCA) is used in the dimensionality reduction method with K-nearest neighbors (KNN) algorithms are used to BCP based on reduced data size by selecting the best features that capture most of the variance in the data. To enhance the performance of the model, a subset of the most critical features is selected from the initial collection of features.

Chapter Four

Results and Discussions

Chapter Four: Results and Discussions

4.1. Introduction

This chapter is on how the proposed architecture achieves a good system's performance in terms of quick response time, low latency, increase throughput and reduce data loss in addition to improving significantly the QoS in the fog node.

The broker is Mosquitto, which deploys data under certain topic names to subscribers (services). There are many MQTT brokers such as Mosquitto, HiveMQ, EMQ X Broker, RabbitMQ, ActiveMQ, etc. Mosquitto is characterized by being self-hosted and managed, TLS/SSL support for secure communication, a publish-subscribe model, and scalability. The MQTT protocol is version 3, which is light and secure and does not contain additional handshake burdens as in the HTTP protocol. It will explain and discuss the results in the following sections.

4.2. Experimental Setup and Performance measures

To conduct experiments, a computer is used with the specifications mentioned in Table (4.1) using the Python language and multiple libraries for programming of publishers, subscribers, and the SC. IoT devices publish data on the broker (mosquitto version 2.0.14). Where The data is sent at a rate of approximately 1 GB/s and many IoT devices publish data messages.

Table (4.1): Experimental Setup

Item	Description
CPU	2.70 GHz intel Core i3
RAM	4 GB
VRAM	1 G
OS	Windows 10 pro 64bit
IDE	Python 3.10
Broker	Mosquitto 2.0.14
MQTT protocol	V3

4.3. Description of Datasets

Three datasets are used to demonstrate the feasibility and efficacy of the proposed model. The lack of sensors that generates the data stream in an acceptable amount and the lack of laboratories appropriate to the nature of the work of IoT system causes dependence on the datasets that contains real data from sensors.

Intel Berkeley is a real-life dataset that contains a set of IoT sensor readings from (https://www.kaggle.com/datasets) named “Intel Berkeley Research Lab Sensor Data” (**Intel Berkeley**). This file includes a log of about 2.3 million readings collected from 54 sensors deployed in the Intel Berkeley Research Lab between February 28th and April 5th, 2004. It is collected and with timestamped topology information, along with humidity, temperature, light, and voltage values once every 31 seconds. Temperature in Celsius degrees. Humidity is temperature-corrected relative, humidity, ranging from 0-100%. Light is in Lux (a value of 1 Lux corresponds to moonlight, 400 Lux to a bright office, and 100,000 Lux to full sunlight.). Voltage is expressed in volts, ranging from 2-3. There are some missing or outlier values and anomalies.

Smart-homes is named "smart-homes-temperature-time-series-forecasting". It consists of data collected from 16 sensors. Each column contains 2765 readings. The dataset is a multivariate time series collected from the monitoring system mounted in a solar house. The dataset is equivalent to about forty days’ worth of monitoring data. The objective is to forecast the temperature of a space in this operation, which is the bedroom, so that a decision may be made on whether or not to turn on the HVAC (heating, ventilation, and air conditioning) system. Missing values could be present in the dataset. Every minute, data is sampled, computed, and smoothed with 15-minute means before being uploaded. Dates, data from

additional sensors, weather data, and other details are all included in the dataset. A multivariate time-series dataset is being used.

The DBC is named **Diagnostic Breast Cancer** medical data sets. The objective is to diagnose and predict early breast cancer. It exists in the Kaggle datasets at (https://www.kaggle.com/datasets). There are 569 records in the collection. 212 cases of malignant breast cancer and 357 (62.7%) cases of benign breast cancer. An ID number, a diagnosis (dataset label: "B" indicates benign, "M" means malignant), and 30 real-valued input attributes are all included in each record. These characteristics are real-valued characteristics that are assessed using a digital image of a breast mass aspirate. 30 real-valued input attributes are stored in a database for 569 cases. In the dataset, there are no missing values.

The DBC database is used to prove that the process of integrating ML algorithms can give promising results in terms of accuracy. Therefore, the results of the PKNSP model are compared with previous works on the same database. More than one algorithm was also combined with the PCA algorithm and the results were compared in Table (4.7, 4.11).

The **reference dataset** is system performance metrics. The objective is to predict of the best number of services that can be deployed in the system. It was created manually by monitoring performance measures that impact on the stability and reliability of the system. There are 385 records in the collection. seven real-valued input attributes are stored in a database for 385 cases. In the dataset, there are no missing values.

4.4. Results and Discussion

Two scenarios have been applied to evaluate the proposed model's behavior; namely, manual service deployment and automatic service deployment. In both scenarios, the ML (OCSVM, RF, KNN, and SVM) are used for missing values, anomaly detection, and predicting the best number of services that the system requires to avoid overload. The model PKNSP is used to improve predictions of the number of services and diagnose breast cancer by reducing the features and data size that are transmitted to the cloud. The next subsection summarizes the results of using these scenarios.

4.4.1. Results of Scenario One

The main objective of this section is to show the measures taken when increasing the number of services that process the data at high data rates by using several publishers using only the Smart-Homes dataset because of its small data size and easy monitoring and tracking of the results.

Four publishers (4-IoT devices) to one service are used in ***the first experiment***, with the same type of data, and under the one topic (Temp). In the initial state, the services in the system are manually deploy by a system administrator which monitors the system performance indices and makes the decision to deploy new services or not.

In one deployment service, the processed data from the four publishers led to overload data in the service and make an increase in the latency from 4.16s to 13.56s based on the data rate from 2499.95bps to 7927.40bps, as shown in Table (4.2). This increase in measure readings is an incentive to find a way to process data and reduce the overload on services.

One of the solutions is to use a set of services that have the same functions for the same topic name.

Table (4.2): Overload on the Service for First Experiment		
Number of publishers	Latency(s)	Data rate (bps)
1 – IoT	4.16	2499.95
2 – IoTs	8.32	2486.2
3 – IoTs	11.13	2740.7
4 – IoTs	13.56	7927.4

To prove that performance is improved when increasing the number of services that perform preprocessing in terms of response time, latency, total time, and data loss percentage, the next explains that.

Multi-publishers and subscribers under the same topic name as shown in Table (4.2), which is called temperature data from the Smart-Homes dataset are used in ***the second experiment***. The steps and settings are the following:

1. Published messages using smart-homes dataset from four IoT device on the SB to the five services (subscribers).
2. Within the service (Collector), it collects data in a block with a size of 503 messages to verify the data stream for missing values, anomaly detection, and then correct data, and that is considered as one job.
3. Compute latency, response time, total execution time, average throughput, and data loss percentage.

Table (4.3) presents the results for latency, data loss, and throughput obtained from data preprocessing in the collector service. All measures are summaries of the total average performance measures when deploying a set of services manually.The average latency, throughput, and data loss are the true standards for evaluating system states and determining the appropriate number of services based on the current state. **When three services are deployed** in the

system, the performance readings are the best possible, and when they increase, system performance begins to deteriorate.

Table (4.3): Summery the Results Second Experiment				
Number services	Average Latency(s)	Average Response (s)	Data Loss %	Throughput J/s
1-Service	2.36	2.59	6.77	17.37
2-Services	2.26	2.44	4.49	18.58
3-Services	**2.17**	**2.28**	**2.22**	**19.82**
4-Services	3.07	3.17	2.22	14.01
5-Services	4.5	4.61	2.22	9.56

The main goal is to reduce latency and data loss and improve throughput. The primary function of the collector service is to collect a data stream, which will receive all messages published on the SB under a specific topic, regardless of the service's knowledge of the IoT device that publishes the data. The received data is collected as blocks of a predetermined size (depending on the type of IoT device and the nature of the data) because the data stream cannot be handled more than once, so needs to process the data blocks with high speed and accuracy. The blocks are re-published on the SB to benefit the other services.

The number of services that process data starts from one to five in efficient time . The Figure (4.1) shows measures of performance for deploying one service. The readings are not stable due to several variations in the operating system environments and the capabilities of machine hardware. Therefore, the operation takes a long time to process data and consider calculating the average latency in the final results. The x-axis represents the time of an operation (timestamp), and the y-axis represents the average latency at a given time. As a result, the latency is from 1.16 to 3.14 using equation (2.1), the average throughput is 17.37 J/s using equation (2.2), and the data loss is 6.77%. Unstable performance, lowest throughput, and data loss using equation (2.7, 2.8).

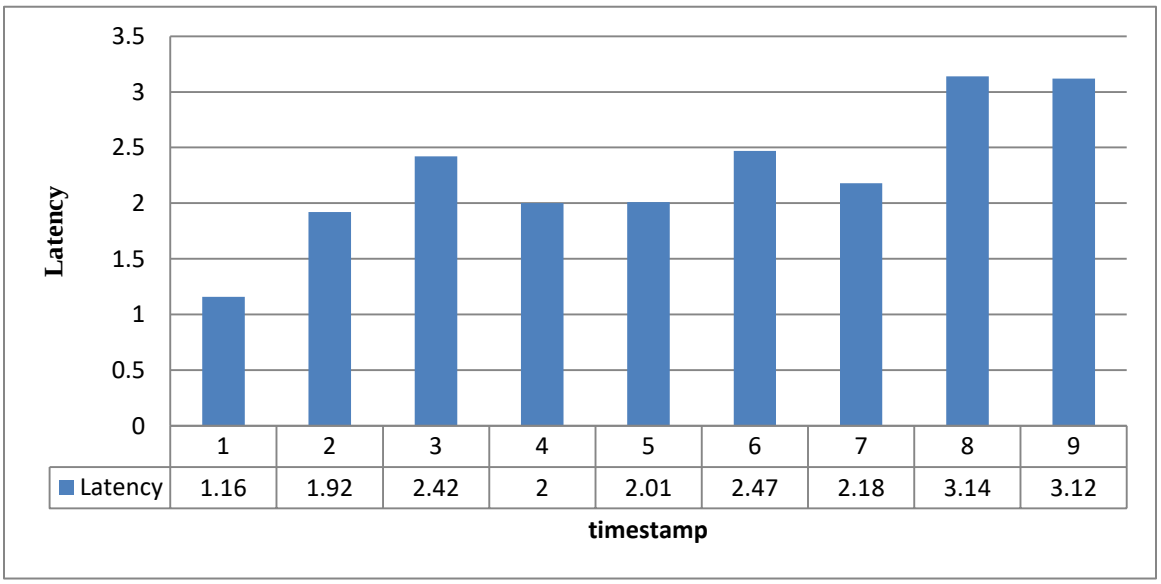


Figure (4.1): The Initial Operation with One Service

The Figure (4.2) shows measures of performance for deploying two services with the same topic name. The readings have been improved because the increasing in the velocity data process in the services. The x-axis represents the time of an operation (timestamp), and the y-axis represents the average latency at a given time. As a result, the latency is from 1.89 to 2.86, the average throughput is 18.58 J/s and the Data loss is 4.49% using equation (2.7, 2.8).

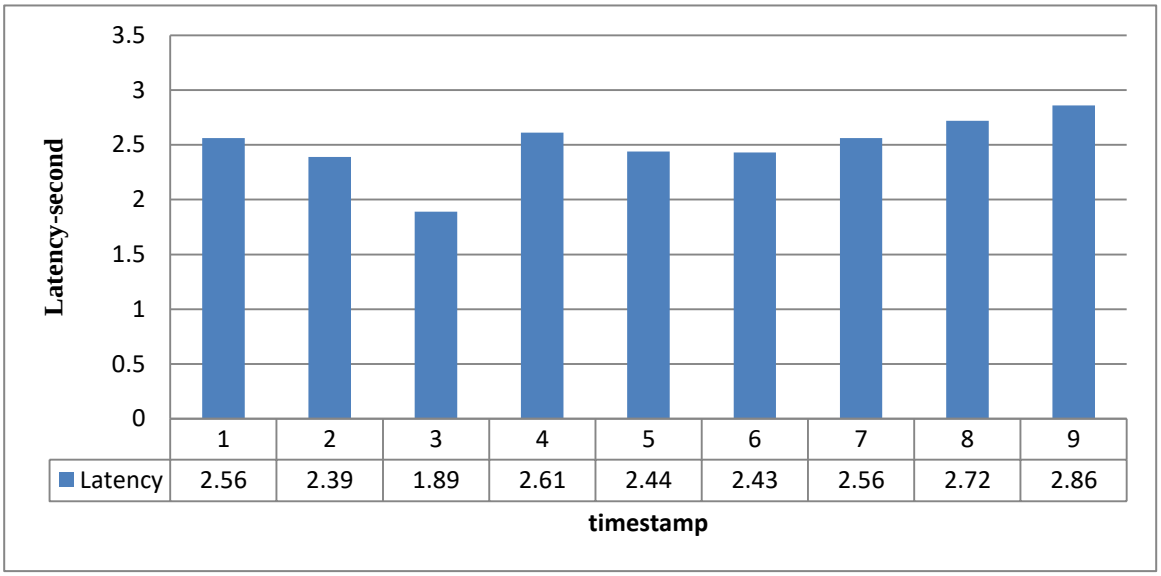


Figure (4.2): The Operation with Two Services

The Figure (4.3) shows measures of performance for deploying three services with the same topic name. The x-axis represents the time of an operation (timestamp), and the y-axis represents the average latency at a given time. As a result, the latency is from 1.86 to 2.53, the average throughput is 19,82 J/s, and the data loss is 2.22%. The performance measures have been better improved.

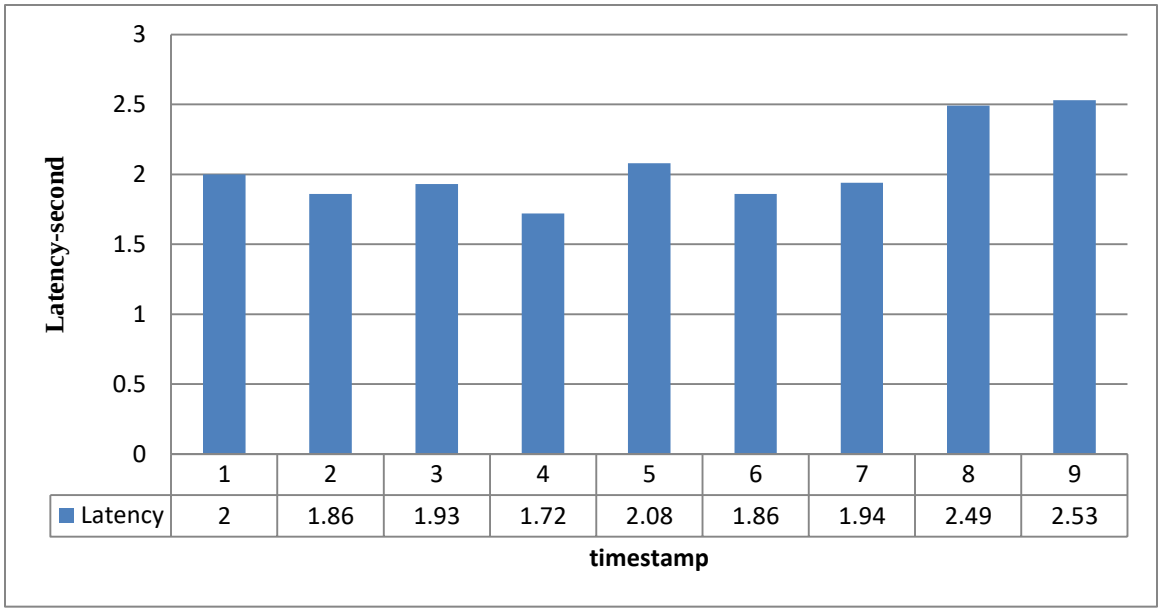


Figure (4.3): The Operation with Three Services

The Figure (4.4) shows measures of performance for deploying four services with the same topic name. The x-axis represents the time of an operation (timestamp), and the y-axis represents the average latency at a given time. As a result, the latency is from 2.26 to 3.62, the average throughput is 14,01 J/s, and the data loss of 2.22% does not change because the time for the process is greater than the collection time. All performance measures have been degraded because the deployed services are more than required in the system; this operation is called starvation.

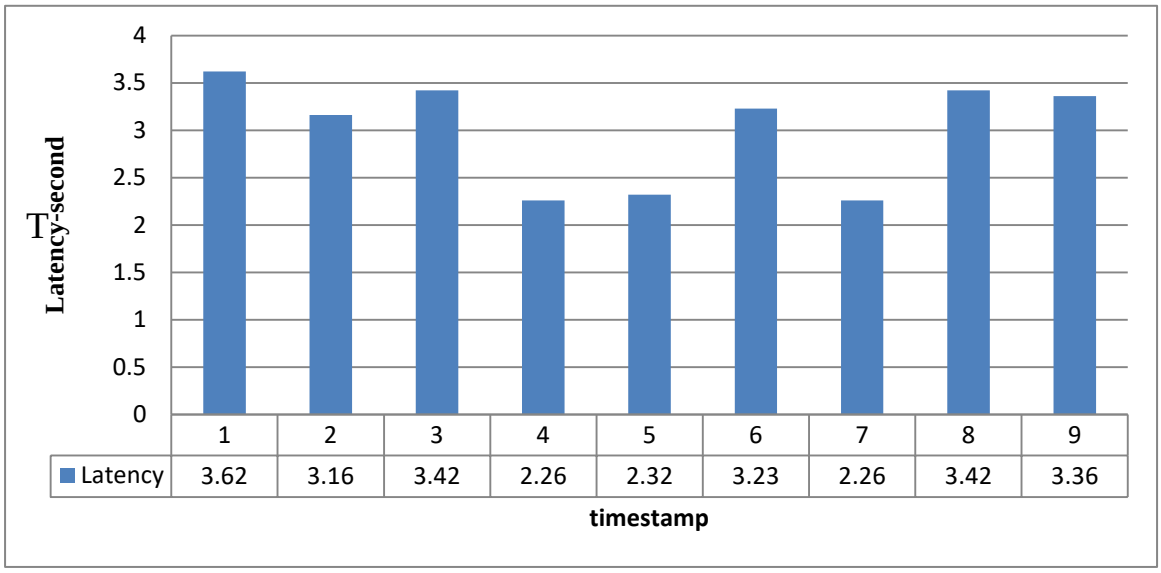


Figure (4.4): The Operation with Four Services

The Figure (4.5) shows a confirmation of the measure's performance degrading when deploying more than three services with the same topic name. As a result, the latency is from 2.26 to 3.62, the average throughput is 9.56 J/s, and the data loss of 2.22%. Except for data loss, all measures are more degraded. Thus, more idle time for service, after three service instances there is no congestion issue causing data loss.

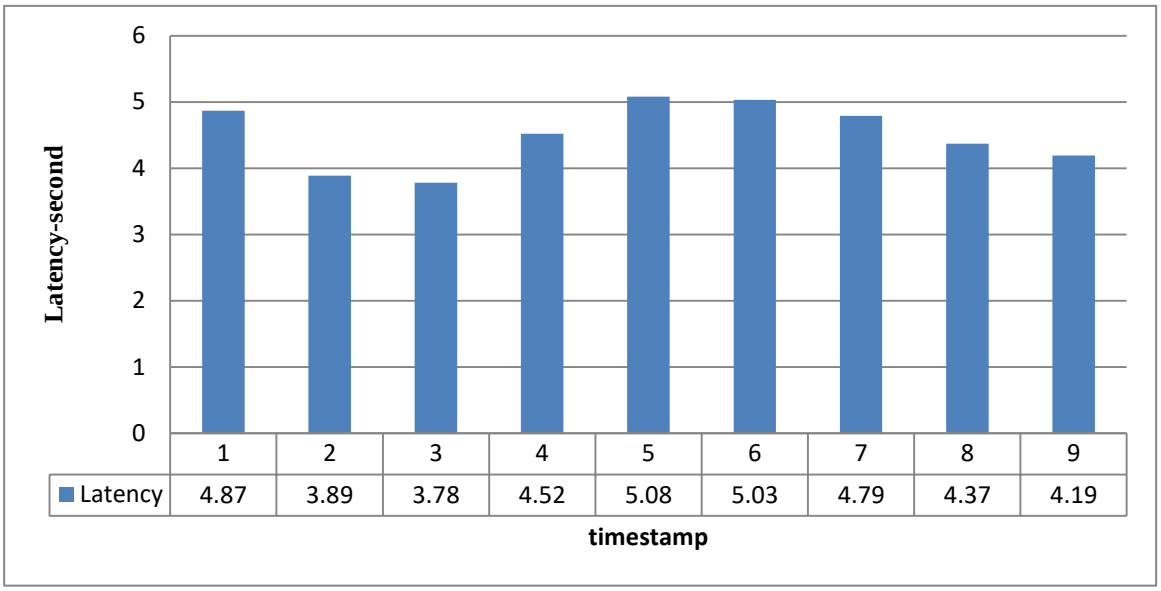


Figure (4.5): The Operation with Five Services

Increasing the rate at which a system can process or handle activities, data, or transactions within a specific timeframe are referred to as improving throughput. The system's capacity to produce results or perform tasks quickly is measured by its throughput. The Figure (4.6) shows an enhancement in the measured throughput when deploying three services with the same topic name. As a result, the average throughput ranges from 9.56 J/s to 19.82 J/s. Improving throughput reduces waiting times, lowers latency, and improves overall system performance.

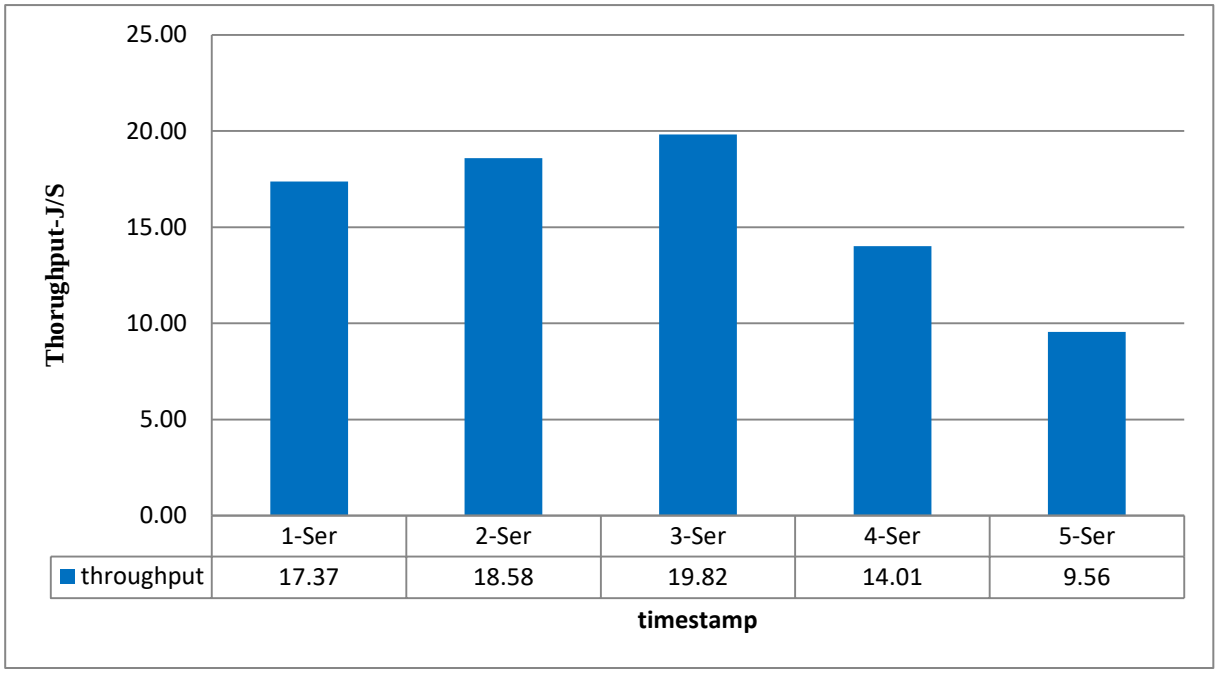


Figure (4.6): The Throughput of Different Number of Services

The third experiment aggregates the results from multiple publishers and subscribers. The deployment of a different number of services has an impact on the total time for processing all data streams as well as response time and latency. The results show an adverse relationship between the increased number of services and the performance measures. A comparison is made on total time, response times, and latency when changing the number of services from 1 to 6 over time. Four services are run to show the best results.

Enhancement in the results starts when increasing the number of services, even reaching four services with the best results; however, it is also affected by other restrictions such as data rate and computing resources. Therefore, when deploying five or six services, there is an increase in total time again because there is a long waiting time for services to collect data and then perform their assigned tasks.

A more accurate definition of latency in the proposed architecture is the time that starts from receiving the first message in the collector service (subscriber) and put in the block until the completion of the last message received and placed in the block, as illustrated in Figure (4.7).

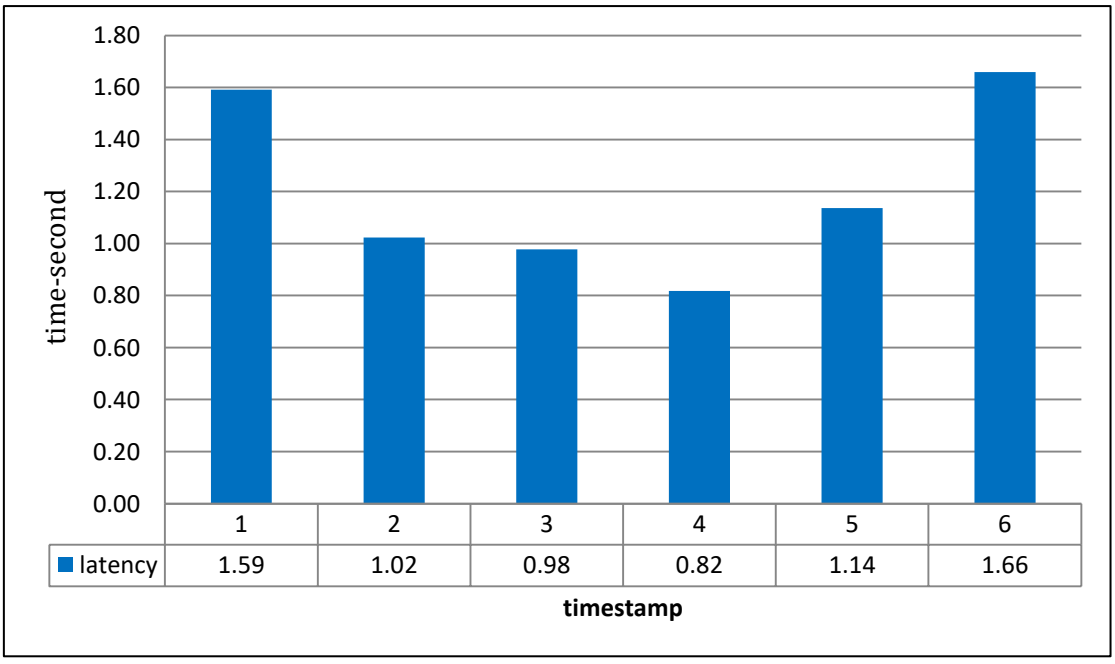


Figure (4.7): The Latency of Different Number of Services

In order for the system to process and react to requests or information from users or other components, response time must be improved. It guarantees that a system can manage a lot of requests without becoming slow or unresponsive.

The response time is from the beginning of receiving the first message in the block until the completion of the detection and correction process for each block and that is illustrated in Figure (4.8).

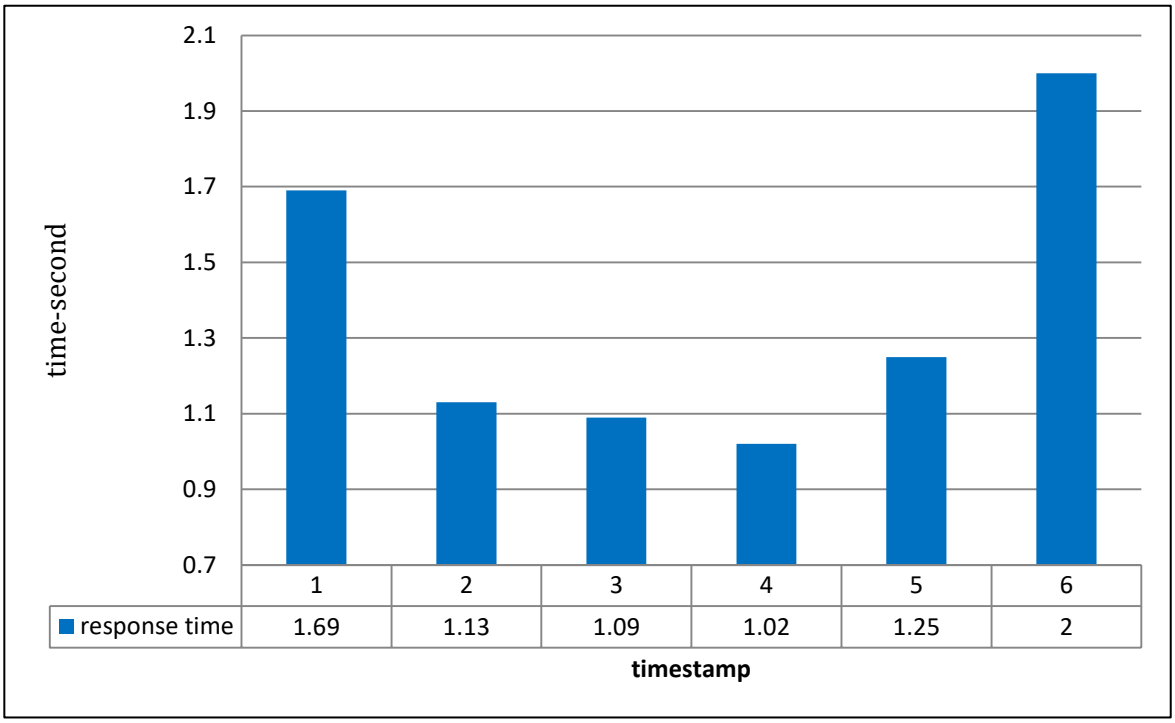


Figure (4.8): Impact the Increasing of Service on the Response Time

Results indicate that the total time for preprocessing all data decreases relative to the number of services received until some point when the time begins to increase again because the available services are larger than the amount of data to be processed (starvation). i.e., more waiting to receive data blocks. The increase in the number of services leads to a reduction in the time for preprocessing. Moreover, to ensure the continuity of data flow to the services and distribute data to the services (subscribers) without redundancy, SB is used, so even if one of the IoT devices is shut down, the system is not stopped as illustrated in Figure (4.9).

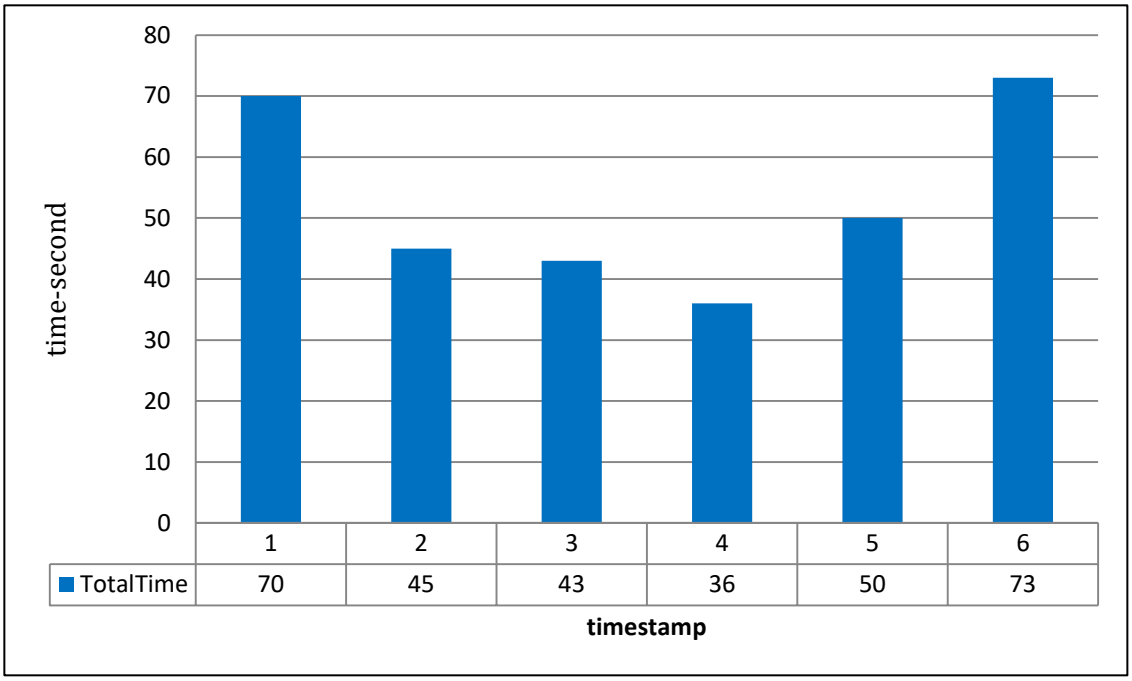
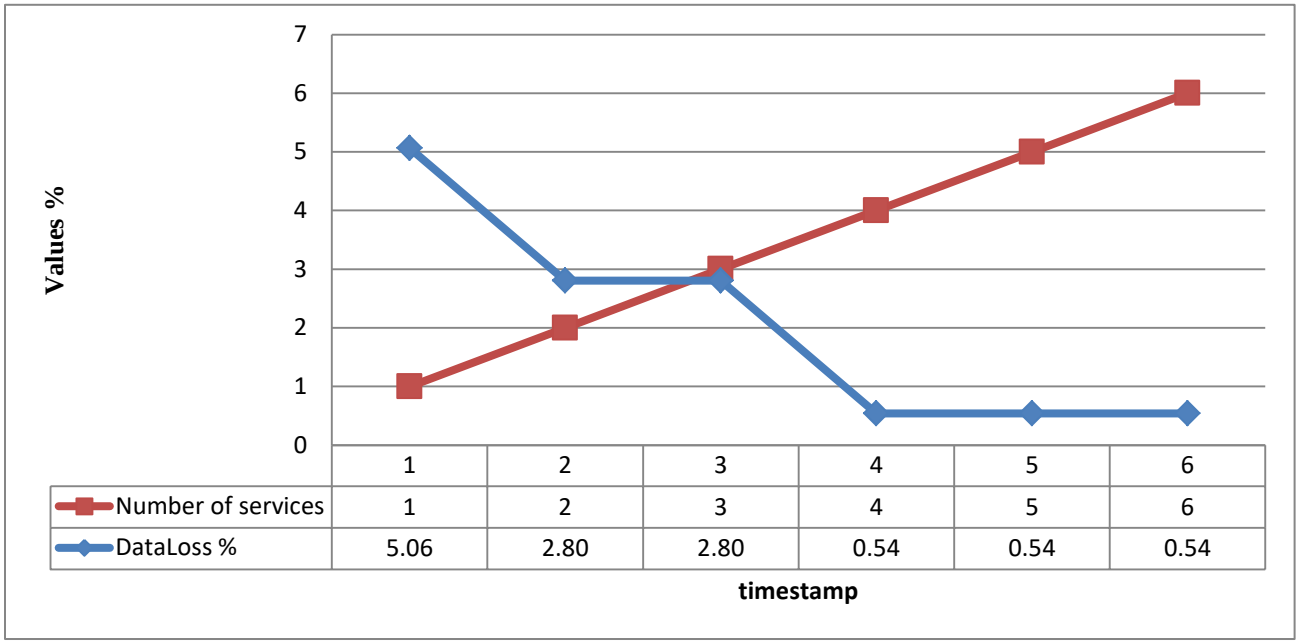


Figure (4.9): The Total Time for Different Number of Services

The latency, response time, and total time can be compared by comparing the values obtained as illustrated in Figures (4.7,4.8,4.9). It is noted that there is an adverse relationship between the increased number of services and the time value. As the number of services that process data increases, processing times decrease until they reach the extent that the number of services is greater than the flow that needs to be processed, thus starting to increase again. The redundant services, after completing the tasks assigned to them, are killed.

The difference between the transmitted data and the received data that has been pre-processed in collector service is the amount of data loss. Data is lost due to the delay in withdrawing messages, which have time to expire, from the SB. As illustrated in Figure (4.10), the x-axis represents the number of services that currently deploying (title the number of services) and deploy service over time (title the data loss). The y-axis represents the data loss values of percentage.

The data loss is high when deploying three services (1, 2, and 3) because there is congestion in the services and a high data rate. In other words, the number of services that are currently deploying in the system is not enough to process all data streams. When deploying four services, the improvement in receiving messages begins because the collector service can process data in time without data loss. Data loss is stable when deploying more than four services because the services take a long time for data collection and processing (collection time); this operation is called starvation.



	1	2	3	4	5	6
Number of services	1	2	3	4	5	6
DataLoss %	5.06	2.80	2.80	0.54	0.54	0.54

Figure (4.10): The Number of Service Reduced Data Loss

Enhancement in reducing data loss continues even after the best stable state of the system (when the number of services is three) because the number of services processing data currently executing in the system is greater than the amount of data deployed on the medium. In addition, the services have enough time to receive and process the data without data loss.

Viewing data as resources for services is emphasizing the strategic value of data in the current technological. Utilizing and managing data effectively can help businesses develop useful services, make better decisions, and gain competitive advantages. But it also has obligations in terms of data administration, security, and privacy. It is possible to have some resource loss when the following causes: i) the buffer cannot handle all arrived data ii) increased resources, and iii) a high transfer data rate.

The congestion in the service is a high transfer rate and its inability to manage and process the amounts of data arriving in a certain period of time results in the loss of some data.

The starvation in the service is an increase in waiting time for tasks and functions within the service to perform the data processing. In this study, a low transfer data rate in a certain period of time or a number of services that are deploying more than data processing needs leads to decreased performance measures and data processes.

4.4.2. Results of Scenario Two

The experiment is applied by using the set of data (temperature, humidity, light, voltage) from the **Intel Berkeley** dataset. The SC is a module/service within the fog node. Its purpose is to monitor and collect information in the screen table for a given service of a particular type. It can also make a decision whether to remove one service or add a new one according to the information collected about the performance of that service. The screen table consists of the information sent from services and is used to evaluate the system's performance containing a timestamp to represent the time of receiving information from the service. The counter of the number of services that are deployed over the timestamp.

When the congestion is on a particular service, the SC algorithm makes a decision to add a service and increments the counter for that service.

The number of services are increased automatically without the intervention of any human element by using the SC. The makes decisions about deploying services or stopping services based on the system service state. When there is an overload on the currently deploying system services (depending on the type of topic published on the IB), the decision is to spawn services to avoid congestion or, on the contrary, kill services to avoid starvation.

The Figure (4.11), shows the changes in the number of services automatically generated by using SC. The x-axis represents the timestamp of an operation ΔT, and the y-axis represents the number of services at a given time for each job based on the topic name. As a result, the number of services varies between 2 and 4 on average. Notice that the number of services (3) is the most used number over the timestamp. It is concluded that SC changes the number of services dynamically.

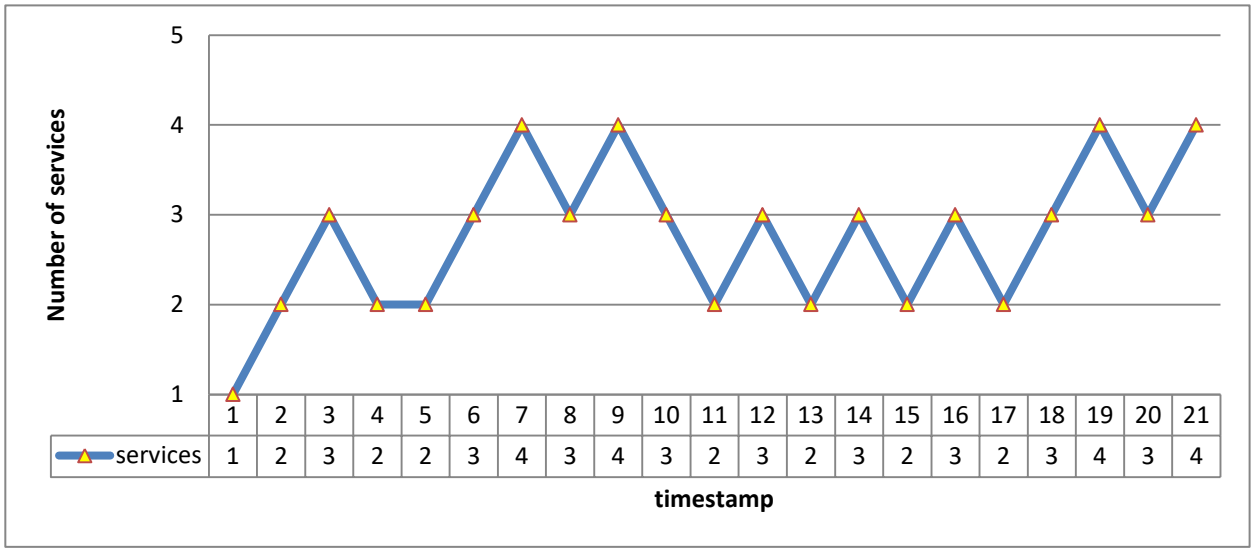


Figure (4.11): Dynamic Number of Service

To Implementation SC that collects details for each block of data that is being executed within the currently executing service in the system. In this experiment one publisher (the temperature data) and four publishers, including the latency, data rate, data loss and throughput are used. The rate for average latency have been calculated instead of a single value (little difference in latency values).

The period (ΔT) that represents the duration for aggregation of received jobs information from services in the SC is used, and the average latency for the given period is calculated. The throughput is calculated by dividing the number of jobs on the total time. By conducting a set of experiments on the optimal size of the window size aggregation of the jobs, the value of (ΔT = 20) is chosen after four values (5, 10, 20, and 30) are tested, based on the stability of latency and increased throughput at this value over the previous values.

The decision is made by comparing the current number of services with the predict number for using PKNSP model. If the difference is positive the SC deploy (spawn) the services that is currently being implemented otherwise stop (kill) one service after finishing the preprocessing.

The results are obtained by publishing the data messages (temperature column only) to the broker and using preprocessors services for publishing details to the SC. Within the service, messages are collected in the form of a block of size=2500 rows. It is found that the real data reading time is approximate.

By **using *one publisher*** (an IoT device), the Table (4.4) presents the measures that are received by the SC and aggregated into the screen table. The data contain the name of services (Temp the original and T1-inst the add new services) that currently represent the data preprocessing. The data values are collected from one publisher with data rate is 1 GB/s, and the topic is temperature (Temp). In addition, the best results are using three data pre-processing services (with instance

Table (4.4): Screen table with one publisher					
Timestamp (ts)	ID-Service	Number of services	Latency(s)	Throughput J/s	Data lost %
0	Temp	1-Service	6.76	2.96	0.88
1	Temp	2-Services	5.98	3.34	0.57
2	T1-inst	2-Services	4.83	4.14	0.46
3	Temp	3-Services	3.18	6.28	0.14
4	T2-inst	**3-Services**	**3.02**	**6.64**	**0.46**
5	T1-inst	3-Services	3.31	6.63	0.57
6	Temp	3-Services	3.21	6.62	0.46
7	T3-inst	4-Services	4.01	4.99	0.14

services).

By **using *three publishers*** with the same topics (Temp), Table (4.5) presents the average latency that is calculated to impact the overload on the service and the same topic. Which is the temperature (Temp). When the number of services is five, the latency values are low although the data transmission value is high, because there are three publishers. The data preprocessing in the services is proceeding at a balanced pace and without high delay in data collection. Enhance in the latency is 50.09 %, and increase the data rate is 42.80%. At the number of services seven and six, the latency is increased again because of the starvation operation.

Table (4.5):Impact Overload on the Latency		
Number of services	Latency (s)	Data rate (bps)
1 - service	10.31	2736.35
2 – services	8.09	2720.51
4 – services	5.09	2603.35
4 – services	5.5	3533.40
5 – services	**4.94**	**3907.45**
7 – services	5.42	4913.85
6 – services	6.9	4807.25

By **using *four publishers*** with different topics (temperature: real, humidity: real, light: real, voltage: real), Table (4.6) presents the average latency that is calculated to impact the overload on the service. When the number of services is three for each topic, the latency values are low. The best number of services leads to the best latency. The average data rate is 1 Gbps.

Table (4.6): The Best Result from Four IoT Devices

Number of services	Latency of Light	Latency of Voltage	Latency of Hum	Latency of Temp
1 - service	10.31	7.32	8.96	8.5
2- services	8.09	7.17	7.63	7.68
3- services	7.27	6.6	7.71	7.74
2- services	6.62	6.96	9.43	9.24
3- services	6.05	6.44	7.39	7.23
4- services	6.65	6.69	7.77	7.45
3- services	5.24	5.22	5.6	7.42

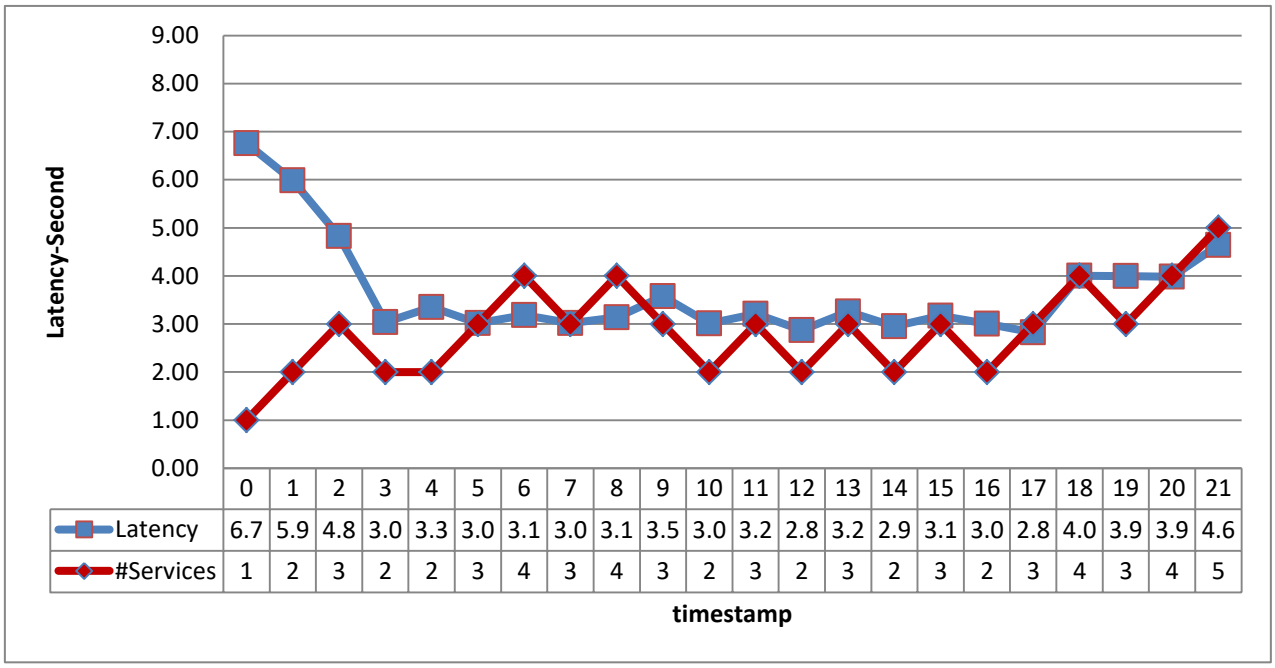
The counter of the number of services are changing automatically, due to the impact of decisions made by the SC. Adding services or deleting services are made according to the predicted results from using a model that integrates the PCA and KNNR algorithm. The results from the implementation of the dissertation idea by using SC with SB in the fog environment tend to:

i) Reduce the latency in processing the data and re-publishing it as clean data to other services (cloud services, etc.) or applications (health care, smart cities, etc.) that require this data.
ii) Improve the throughput of our architecture.
iii) Ensure that data loss is reduced to the greatest extent possible.

The Figure (4.12) shows the impact of the changed dynamic number of services with the same topic name in terms of latency. The x-axis represents the timestamp of an operation ΔT, and the y-axis represents the average latency at a given time for each job based on the topic name. As a result, the latency starts at 7s

and is stable at 3s.

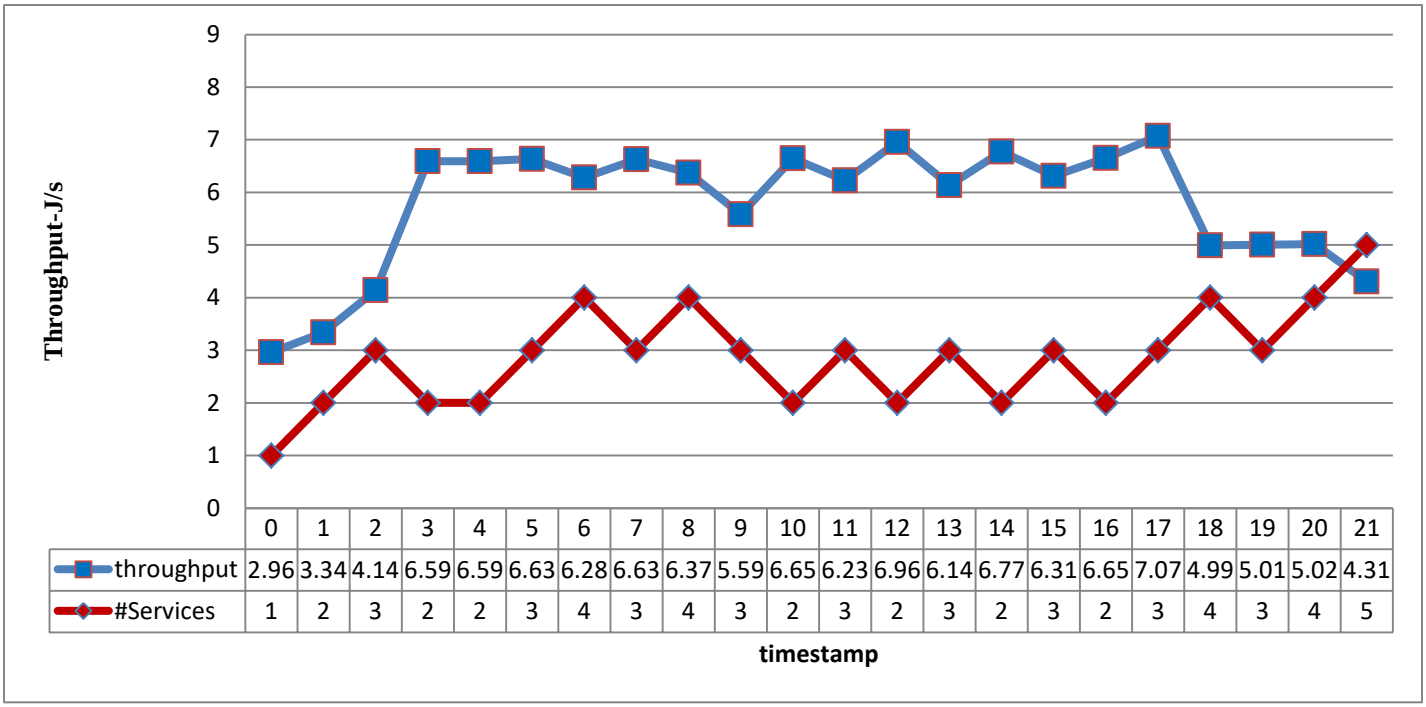
The best latency is when the number of services is three along the timestamp of the Figure. Deploying three services automatically by SC is good for a given topic type and data rate.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Latency	6.7	5.9	4.8	3.0	3.3	3.0	3.1	3.0	3.1	3.5	3.0	3.2	2.8	3.2	2.9	3.1	3.0	2.8	4.0	3.9	3.9	4.6
#Services	1	2	3	2	2	3	4	3	4	3	2	3	2	3	2	3	2	3	4	3	4	5

Figure (4.12): Reduction the Latency

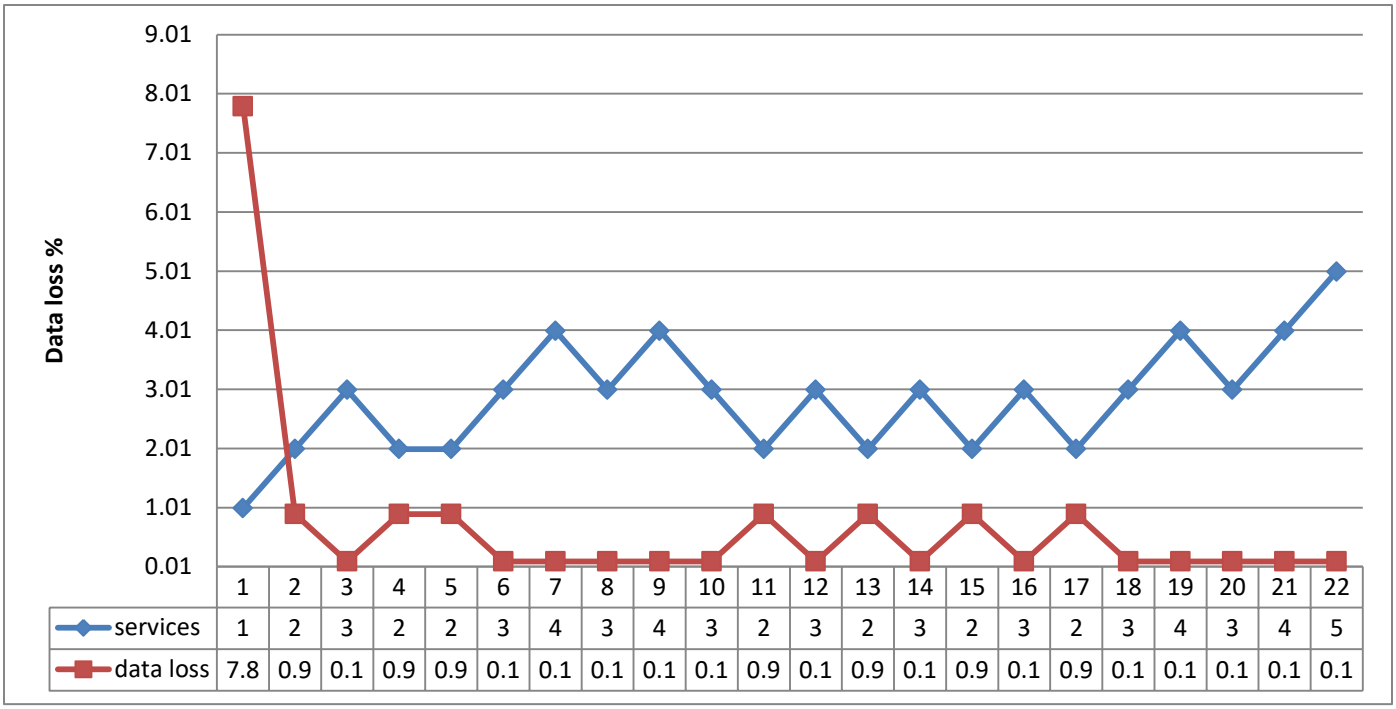
The Figure (4.13) shows the measurement of enhanced system performance by changing the number of services dynamically in terms of throughput. The x-axis represents the timestamp of an operation ΔT, and the y-axis represents the throughput at a given time for each job based on the topic name. As a result, the throughput starts at 3j/s and is stable at 6.5j/s. The best throughput is when the number of services is three along the timestamp of the Figure. SC successes on a put system on highest.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
throughput	2.96	3.34	4.14	6.59	6.59	6.63	6.28	6.63	6.37	5.59	6.65	6.23	6.96	6.14	6.77	6.31	6.65	7.07	4.99	5.01	5.02	4.31
#Services	1	2	3	2	2	3	4	3	4	3	2	3	2	3	2	3	2	3	4	3	4	5

Figure (4.13): Improvement the Throughput

The Figure (4.14) shows the measurement of enhanced system performance by changing the number of services dynamically in terms of data loss rate. The x-axis represents the timestamp of an operation ΔT, and the y-axis represents the throughput at a given time for each job based on the topic name. As a result, the data loss rate starts high at 8% and improves until it reaches 2%. The best low data loss rate is when the number of services is three along the timestamp because there are enough services processing the data stream. Notice there is a data loss only for a few services (5 services) and tends to be zero when there are a large number of services deploying. It is concluded that the data loss rate measure shows different behavior in dynamic environments.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
services	1	2	3	2	2	3	4	3	4	3	2	3	2	3	2	3	2	3	4	3	4	5
data loss	7.8	0.9	0.1	0.9	0.9	0.1	0.1	0.1	0.1	0.1	0.9	0.1	0.9	0.1	0.9	0.1	0.9	0.1	0.1	0.1	0.1	0.1

Figure (4.14): Data Loss in Dynamic Model

The SC receives a data block from the currently executing service containing details of latency, execution time, response time and data loss. These blocks are collected until they reach a value of (ΔT). The average latency value (Current-latency) is calculated according to Equation (2.2), the total time for collection block ΔT according to Equation (2.6), the throughput value, and according to Equation (2.1), the data loss according to Equation (2.8). In addition, the SC makes the decision to deploying new service or stop one by changing the state of the currently deploying services counter add or subtract by one.

Based on the aforementioned, it is noted that there is a relationship linking increasing the number of services (dynamically) with an improvement in latency, throughput, and reducing data loss by using the best model outcome from integrating PCA and KNN algorithms and best ***k***.

In practical experiments, the number of services is automatically increased without the intervention of any human element by using the SC. It makes decisions about deploying services or stopping services based on the system services state. When the overload is on the currently deploying system services, the decision is to spawn services to avoid congestion, contrary, kill services to avoid starvation.

4.5. Results of PKNSP Model

In the SC, the model is made by integration of the PCA and KNN MLs to predict the appropriate number of services and use for the dimensionality reduction for the best feature selection to diagnose and predict breast cancer. Feature selection is embedded in this model. PCA is used to select the best features from a set of data after dimensionality reduction. The number of features is seven, but after applying PCA, it is reduced to three features. As illustrated in Figure (4.15), the scale and the shades represent the correlation between the two features in the reference dataset.

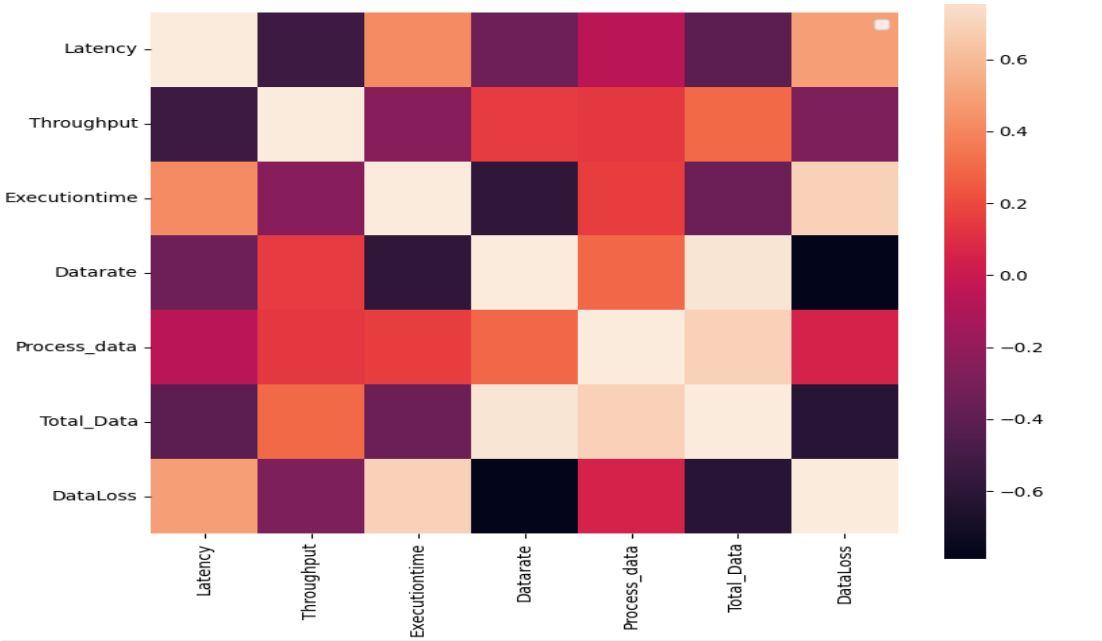


Figure (4.15). The Correlation for Reference Dataset

The middle values, which are neither having too high nor too low variance, are used in this study. That is, not those that have too high variance or too low. As illustrated in Figure (4.16), the number of features is seven, after applying PCA, the principal components are three (features) based on the threshold uses in PCA. The y-axis represents the Explained variance (EV), the x-axis represents the number of new principal components. Explained variance (EV) is the amount of covariance after the projection of the new values for principal components into the original feature space.

Based on the use of a cumulative explained variance threshold (82%), the number of principal components (KP) to be retained can be chosen, and it is found that this threshold is appropriate for the data of features in terms that are not highly correlated. Therefore, it does not have the same information and teaches the model well. In addition, this threshold explains a significant portion of the data variance.

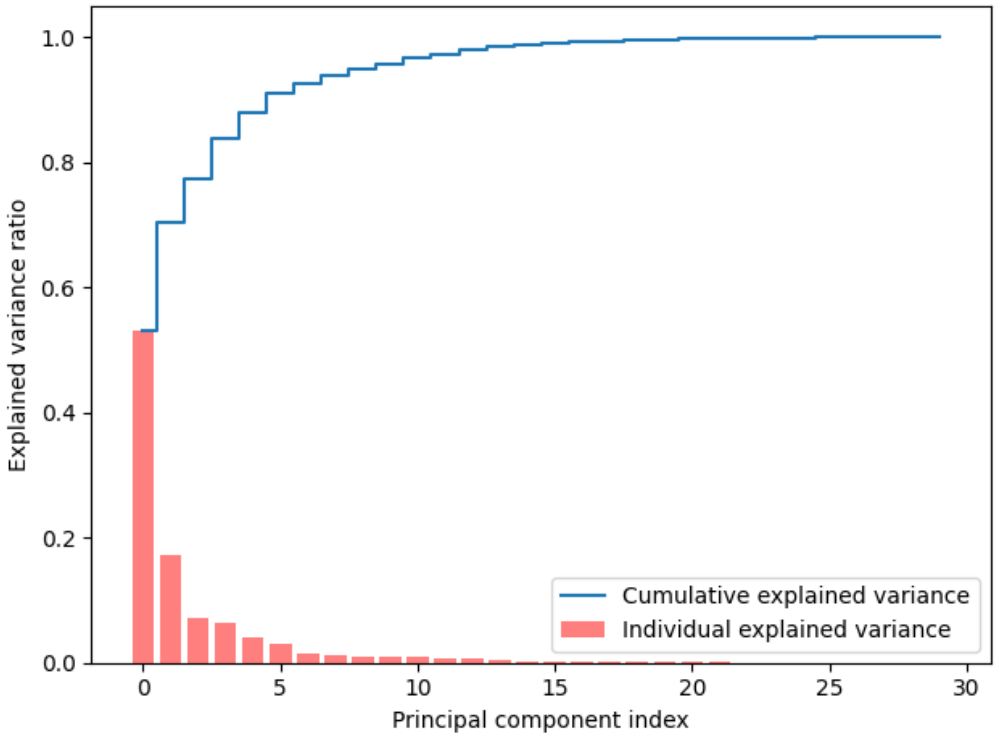


Figure (4.16): The EV with Principal Components

Evaluate the model using the MSE compared with (OCSVM, SVR, and Random Forest). The results obtained from applying MLs with minimum MSE are illustrated in Figure (4.17) , shows the best ML algorithm for the implementation in the SC for making decisions. The x-axis represents a set of MLs used to predict the best number of services, and the y-axis represents the MSE values for each algorithm used. As a result, the minimum MES using KNN is 0.00434. The integration model between PCA and KNN finds the minimum MSE and the best K because the dimensionality reduction is used before applying KNN. It is concluded that PCA and KNN are used in the SC because of a small dataset and a simple relationship, and it is also interpretable.

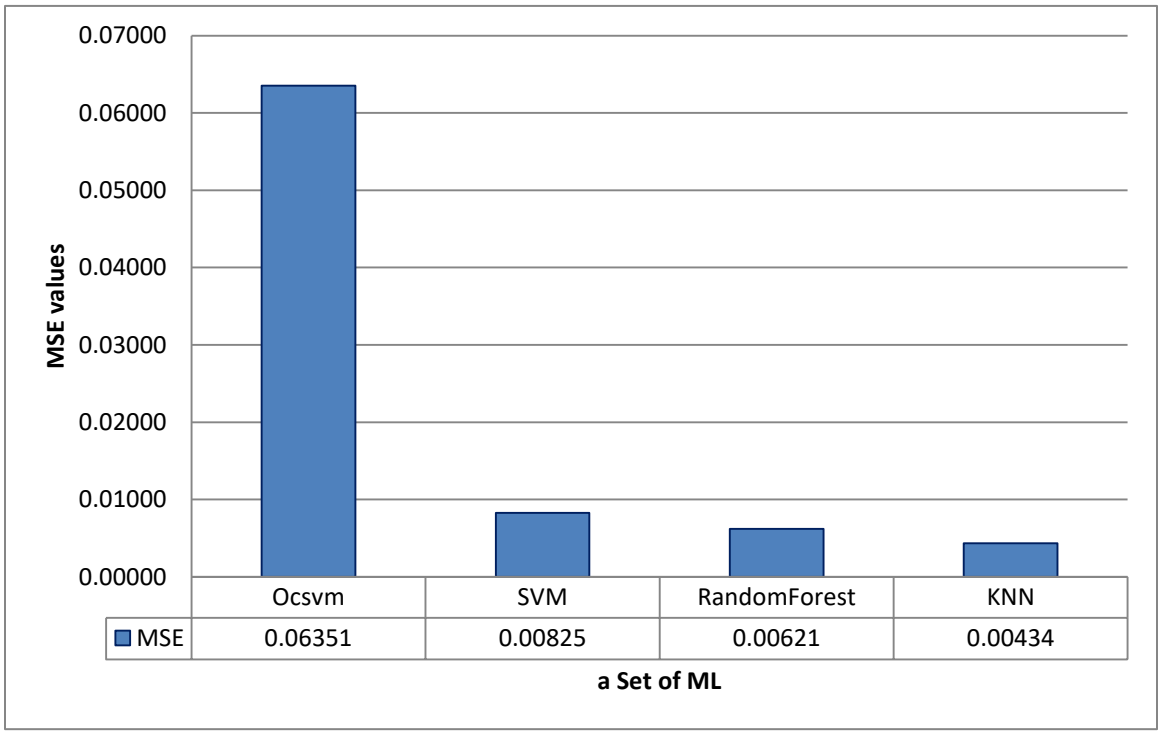


Figure (4.17): The MSE for MLs

Determine the number of neighbors for the KNN algorithm (***K***) that is used in the final model. The Figure (4.18), shows ***K*** that is used in the final model. The x-axis represents the number of K values, and the y-axis represents a set of MSE values obtained from an iteration of applying the KNN to the produced model from feature selection. As a result, k equals three, giving a low MSE. Notice the MSE degraded and increased for K greater than 5, with three components from PCA. Conclusion: No more tests are required.

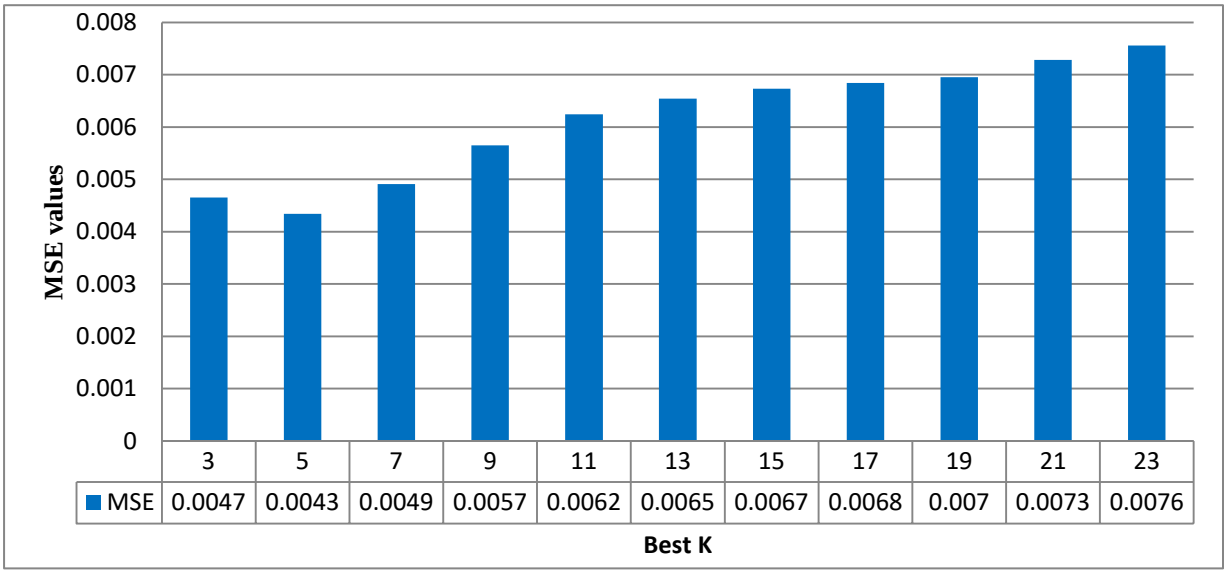


Figure (4.18): Selection #K of Neabouers

4.6. Result of BCP Model

Based on the constituents of the confusion matrix of the BCP model shown in Figure (4.19), the model achieved an accuracy, precision, and F1-Score of 99.12, 99.13, and 99.12, respectively. The reduction of the data size by applying PCA achieved a reduction of 68%. It achieved a reduction of features from 30 to 8, as shown in Figure (4.20) where (a) represents the original features and (b) represents the features of applying PCA. This is useful when sending data to the cloud.

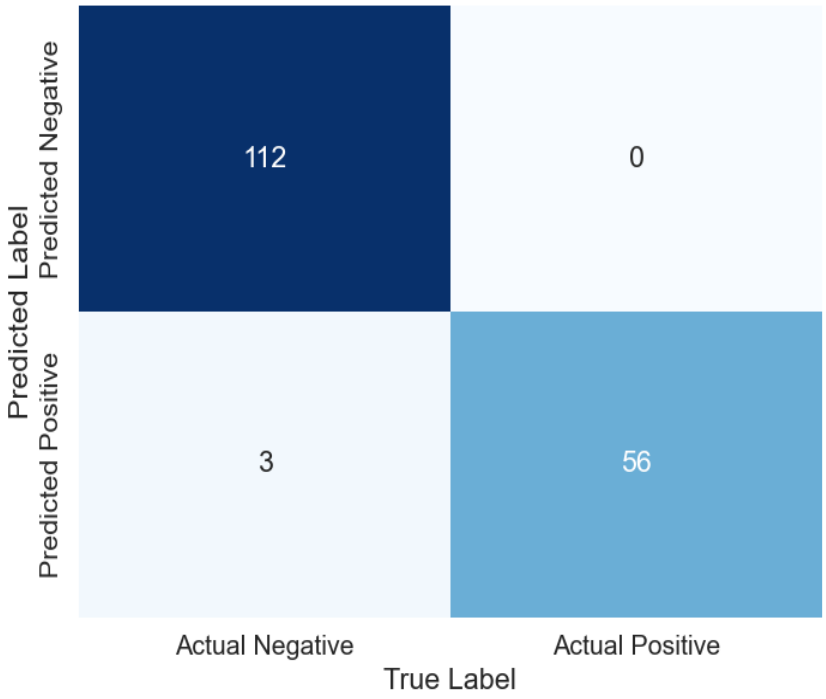


Figure (4.19): Confusion matrix of the IPK

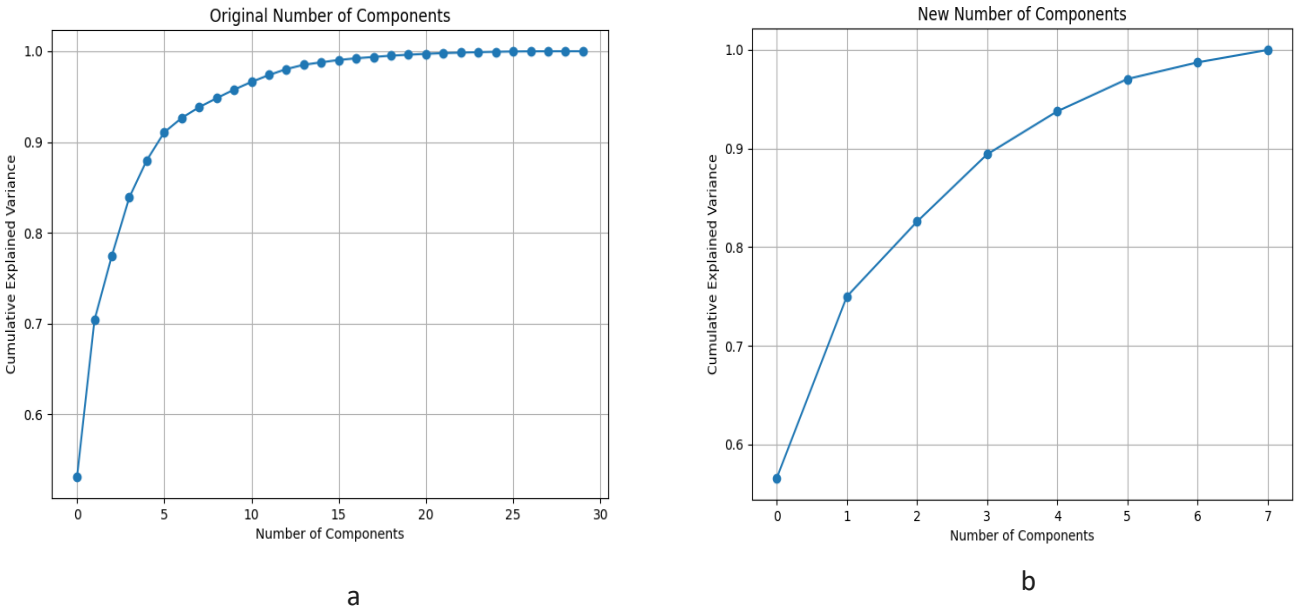


Figure (4.20): Data Size Reduction

Table (4.7) shows the results of using the PCA to produce new data components after dimensionality reduction and embedded feature selection based on the use of a cumulative explained variance threshold (95%). The new data components are used in the KNN, RF, SVM, and CNN. Compare the results in terms of accuracy between using PCA and without using PCA.

Table (4.7): Results from BCP model		
Method	**Accuracy with PCA**	**Accuracy without PCA**
RF	96.73	**94.49**
SVM	97.86	92.98
CNN	97.37	93.86
KNN	**99.12**	94.15

The high accuracy is obtained from the BCP model because the KNN performs better when given smaller amounts of data since there are fewer chances for overfitting due to reduced variance between isolated neighborhoods within smaller sample sizes than larger ones. Therefore this method becomes useful in cases where there isn't an abundance amount of available relevant information about a certain problem domain e.g. medical applications. Using PCA for dimensionality reduction and then feature selection with KNNR, you can follow these main steps:

1. Applying PCA to select the best features from the feature space.
2. Select the best principal components that capture the most significant variance (they are not very high and low variance).
3. Applying KNN using the selected principal components as features with the best number of K for KNN.

The BC datasets consist of a large number of features; therefore, PCA dimensionality reduction in integration with KNN helps improve the results. Due to its reliance on distance measures. Table (4.8) shows the results of comparing the PKNSP model with feature selection that is evaluated by measures of accuracy, precision, recall, and f1-score.

Table (4.8): Comparison for PKNSP with Feature Selection method

Method	Accuracy	precision	recall	f1-score
features selection	91.22	0.94	0.89	0.92
features selection+PCA	89.47	0.94	0.92	0.93
PKNSP	**99.12**	**0.99**	**0.98**	**0.99**

4.7. Results of Using the OCSVM

A model for anomaly detection based on using an unsupervised One-Class Support Vector Machine to detect anomalies is proposed in this study. The parameters of the kernel are RBF, the coefficient of gamma (gamma), and the minimum for train error (nu) between (0,1).

In practical experiment, an injection of 2% of anomalies data in the block. The OCSVM model is evaluated by accuracy using the TP, TN, FP, and FN, after applying the best parameter values (gamma=0.001, and nu=0.02) that are obtained from the several tests that are done on the data using a set of values.

Table (4.9) presents the values used to find high accuracy. By using the equation to find the accuracy (mentioned in Chapter Two section 2.6.1 Equation (4)), based on the anomaly values that are injected with the normal data, the accuracy for each parameter is calculated after knowing the respective values for (TP, TN, FP, and FN).

Table (4.9): The Best Parameters with Accuracy							
(gamma, nu)	(0.01,0.02)	(0.001,0.001)	(0.001,0.002)	(0.001,0.01)	(0.001,0.02)	(0.002,0.02)	(0.003,0.02)
Accuracy	93.57	98.83	98.64	98.83	**99.44**	99.22	99.22

Figure (4.21) illustrates the best parameters with the highest accuracy. The collector service makes processing more accurate and fast by using ML algorithms for anomaly detection. The result of this function is one of two possibilities:

i) The block data is normal.

ii) The data is abnormal (contains abnormalities), which will be sent to the correction function.

Where the low accuracy is at the value of the parameter (0.01,0.02), and the high at the value of the parameter(0.001,0.02).

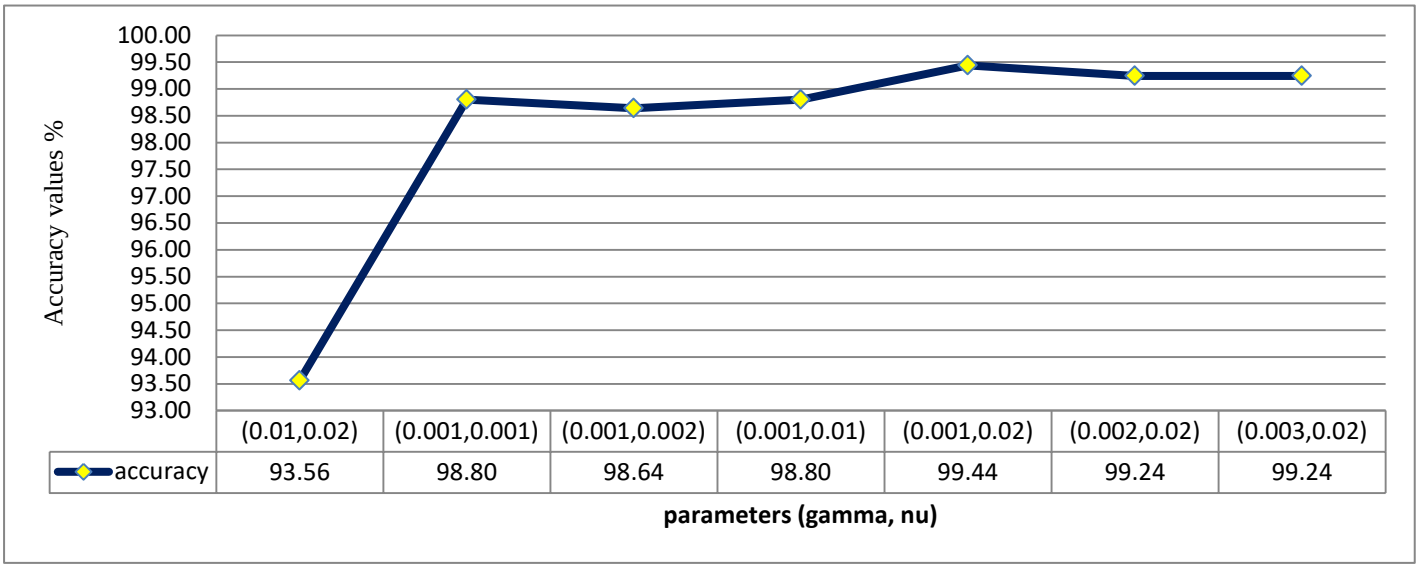


Figure (4.21): The Set of Parameters Values

To prove the effectiveness of the OCSVM model, that is compared with two previous studies that are conducted Intel Berkeley dataset for anomaly detection [104], [120]. Moreover, the proposed models' results are better than all these models.

Which its accuracy is (93% and 99.05%). whereas the accuracy in this study appears enhanced (99.44%). In addition, OCSVM is compared with two MLs (KNN, SVM, and ISF algorithms), as shown in Figure (4.22).

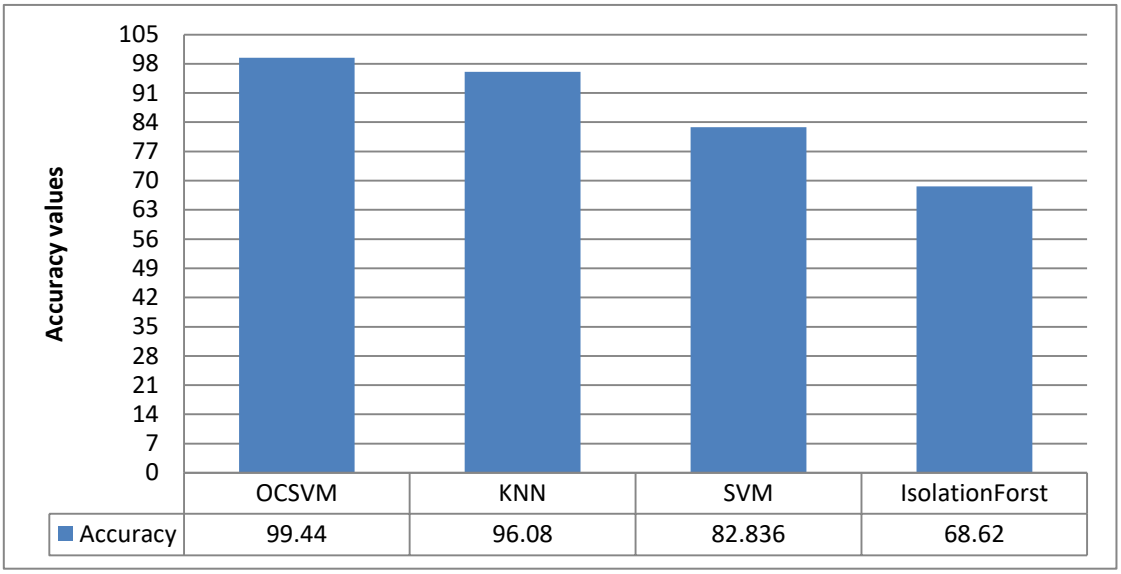


Figure (4.22): The Accuracy Using Different Techniques

4.8. Comparison with Related Work

Using a set of common methods for processing and making decisions in the fog node near the data sources as in the basic configuration in section 4.4. is focused in this study. Then, building the SC and adding to the architecture for monitoring the system measures and making decisions either (spawn or kill of services) by several ML models. The results, as illustrated in Table (4.10), are obtained from the implementation of the SC model. The results are compared with previous studies in terms of response time, latency, and accuracy for anomaly detection. Experiments demonstrate the practicality of SC in the proposed architecture and the important relationship between the number of services and the enhancement of the system performance measures.

Therefore, fast and accurate decisions are made in efficient time by using PKNSP for prediction, leading to high reliability and avoiding congestion on the services. Experiment settings:

1. Messages are published from the Intel Berkeley dataset (four IoT devices) to the SB that receives the same number of messages (2313155) in each experiment.
2. Collect the data in the block with a size of 2500 messages.
3. The SC with one service is the initial state of system operation.

Table (4.10): The Results from Implemented the SC

Number of Services	Latency	Throughput	Response time
1	6.76	2.96	6.96
2	3.46	5.78	3.71
3	**3.43**	**5.83**	**3.63**
4	3.58	5.59	3.87
5	4.64	4.31	4.89

The best result when the number of services is three for latency, throughput, and response time. The unit for results was the second of each block that contained 2,500 messages. After converting these unit results to milliseconds for latency and response time, the results became as in the Table (4.11).

For the architecture and system's performance, the findings from related works in terms of the average latency in [7] for a message is 20.45ms, Delivery latency traffic latency is 48.56ms in [21], and in [24] is 8.83ms.

The response time in [15] is 109ms, the FogFlow-Discovery for ID- and topic-based queries in [26] is 39.5ms , and the response time is 17.86ms in [25]

Notice: The unit in the results obtained from related work was millisecond for one message, whereas in the recent study, the seconds are converted to millisecond for each message in all results as shown in Table (4.11).

Using SB (without SC) in this study by using Smart-homes dataset and the average latency is 4.34 ms, and response time is 4,56 ms. However, Using SB and SC by using Intel Berkeley dataset , the average latency is 1.37 ms, and response time is 1.54 ms .

Table (4.11): Compare results from this work with related works after converting (s to ms)

Reference	Latency (ms)		Response time (ms)	Accuracy %	Improvement rate
[7]	20.45		-		6.699
[24]	8.83		-		15.515
[15]	-		109		1.413
[26]	-		39.5		5.570
[21]	48.56		-		2.821
[25]	31.33		38.90		4.54
[120]	-		-	99.05	
[104]	-		-	93	
This work (SB only)	4.34		4.56	99.42	33.772
This work (SB & SC)	**1.37**		**1.54**	**99.44**	
[1]This work (SB & SC) Overload (4-IoTs)	1.97		2.20	99.44	69.543
[2]This work (SB & SC) Overload (4-IoTs)	Light	2.09	2.31	99.44	65.550
	Voltage	2.08	2.32	99.44	65.865
	Hum	2.24	2.53	99.44	61.161
	Temp	2.89	3.17	99.44	47.405

Notice: [1]Smart home dataset with same topic name (Temperature).

[2]Intel Berkeley dataset with difference topic name (Light, Voltage, Humidity, Temperature).

The findings demonstrate the effectiveness of the model through deploying

a set of services and SC with an MQTT broker. The improvement in average latency is 91.65%, the response time is 97.34%, the reduction in data loss is 84.49%, and the throughput is 7.07 j/s in the proposed architecture. In contrast, the previous studies didn't use the SC and didn't split the data stream into several blocks in the services.

The BCP model is being compared with other ML models in terms of accuracy, recall, precision, and score on the dataset. All the models constructed before a final model with their performance indices and the comparison are shown in Table (4.12).

Table (4.12): Comparison of the BCP model with the other existing models DBC Dataset.

Reference	Year	Method	Accuracy%	Precision%	F-score%
[40]	2018	PCA+KNN	96.4		
[37]	2019	PCA+KNN	95.57		
[38]	2019	SVM	97.36		
[39]	2019	SVM	98.82	99.07	98.41
[121]	2020	LR with Area under the curve	98.06		
[34]	2021	SVM	98.51		
[36]	2021	CNN	98.37		
[122]	2022	Voting Classifier	97.61		
		Polynomial SVM	99.03		
		KNN with hyperparameter	97.35		
		PCA +Logistic Regression	94.87	94.81	92.9
[123]	2023	SVM		98	96
		KNN		94	96
		RF		96	97
[124]	2023	Voting Classifier (LR+SVM)	98.77	98.83	98.68
In this work	**2023**	**BCP**	**99.12**	**99.13**	**99.12**

4.9. Summary of the Chapter

The results emphasize that there are some measures that can play an important role in enhancing the system's performance. Such these measures can also impact on decision-making in terms of adding or removing the services. The Initial deployment of services is dynamic by checking the given topic name in emergency service with a set of active topic names.

In terms of system performance, the results assert that integrating SB with SC have a positive effect on improving throughput and latency in services. As it increases in reliability by guaranteeing continuous data flow, avoids system failure and services stopping from working.

According to the most important results, regarding the PKNSP, BCP, and OCSVM models, it is concluded that each model can be suitable for a specific purpose. For example, when determining the number of services in the system is very critical and important to avoid defects in the data stream and generate wrong values. Anomaly detection, messing value, and breast cancer diagnosis are other examples of different fields in which the proposed structure can be applied.

Besides using ML algorithms in service, integrating the ML algorithms in SC indicate to that PCA and KNN can enhance the accuracy, precision, f score, and MSE of the model.

Chapter Five

Conclusions and Future Directions

Chapter Five: Conclusions and Future Directions

5.1. Introduction

The most significant findings of the tests and the suggested studies that can be highlighted will be stated in this chapter. Hence, this chapter is divided into two primary portions. The first is devoted to highlighting the most significant findings of this study using suggested tools and algorithms. The second section suggests some potential directions for this study's future.

5.2. Conclusions

This dissertation introduces a adaptive architecture in the fog node based on integrating a broker with SC and a set of services which uses ML for data processing and makes dynamic decisions to add (spawn) or remove (kill) services automatically. Improvement of the system's performance by using the response time, latency, throughput, and data loss. Finally, the enhancement of preprocessing of data streams in efficient time .

The adaptive architecture is capable of deploying a set of services in the fog node, adapting to the expansion of IoT devices, and dealing with heterogeneous data. The automated deployment and dynamic publish/subscribe increase the speed of data preprocessing and data cleanliness from any problems that hinder the optimal performance of the system before sending data to applications that require the data. The SC can make a decision automatically in efficient time by reading information messages delivered from services. SC can successfully determine whether the system needs to add or remove services.

The decisions are made when the system lacks the congestion of data on the currently implemented service or there is a starvation case in these services.

Based on the findings of the proposed models, many conclusions can be drawn:

1) The adaptive architecture avoids the bottlenecks inside the services with the exploitation all resources in the fog node.
2) No special configuration is required on the IoT device level when using a broker in a fog node and can apply to the minimum resources.
3) Employed a broker as a flexible and distributed gateway for data streams. The broker is more secure by guaranteeing the publishers are not aware of the subscriber, and the system does not fail when one IoT device is shut down or disconnected.
4) The duplication of subscribed services is a feasible approach for handling high-rate data streams, enhancing system performance, and utilizing available system resources. The proposed method proved to be effective where the overload in the service and according to the topic of data published, the decision is to add services to the system under the same topic or remove them.
5) The proposed OCSVM hyperparameters that control the behavior of the algorithm (gamma, nu, kernel). In this dissertation, the best hyperparameters by cross-validation are selected and to achieve good performance to the OCSVM model for anomaly detection tasks. In addition, dividing the data stream into several blocks in the services to discover missing values and the applying OCSVM ML for anomaly detection, led to high accuracy.
6) SC has succeeded in controlling increased numbers of services automatically and maintaining the best possible system performance. Screen table used to collect measures from services and input to the

SC algorithm to predict the best number of services the system needs.

7) The proposed integration of the ML algorithms is proved to be effective makes an accurate decision and adds less overhead since training is offline and usage is straightforward online. dimensionality reduction using PCA and then selecting the best features and selecting the best K-neighbors (for KNN) is significant in building the PKNSP, and BCP model well.
8) SC still uses part of the system's resources.

5.3. Future Works

Adaptive architecture is expected to be a path of interest for researchers because of its flexibility and scalability, which allow adding services and connecting IoT devices or sensors easily, especially in large places such as cities or health facilities. In future works can be drawn:

1) SC can be expanded so that it has the ability to process data on different fog nods and different locations.
2) SC can deal with service scalability in several locations depending on the data of IoT, which is heterogeneous data.
3) The broker guarantees a fair distribution of data to services according to a specific topic, which can be developed into a matrix of topics in order to accommodate the various devices that generate data in huge quantities.
4) Applying the proposed architecture to the cloud, with all measures using and without the broker. Thus, comparing performance and determining the best results based on the nature of the data and the environment of the problem.

References

[1] A. Bahga and V. Madisetti, *Internet of Things: A hands-on approach*. Vpt, 2014.

[2] S. P. Janani, I. J. Jebadurai, G. J. L. Paulraj, J. Jebadurai, and S. Durga, "Preparedness for managing pandemic using distributed mobile brokers - Using COVID 19 use case," *Mater Today Proc*, vol. 51, pp. 2384–2388, 2022, doi: 10.1016/j.matpr.2021.11.586.

[3] M. Muneeb, K. M. Ko, and Y. H. Park, "A fog computing architecture with multi-layer for computing-intensive iot applications," *Appl Sci*, vol. 11, no. 24, pp. 11585–11598, 2021, doi: 10.3390/app112411585.

[4] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," *internet Soc*, vol. 80, pp. 1–50, 2015.

[5] M. Rzepka, P. Boryło, M. D. Assunção, A. Lasoń, and L. Lefèvre, "SDN-based fog and cloud interplay for stream processing," *Futur Gener Comput Syst*, vol. 131, pp. 1–17, 2022, doi: 10.1016/j.future.2022.01.006.

[6] A. A. Mutlag *et al.*, "MAFC: Multi-agent fog computing model for healthcare critical tasks management," *Sensors (Switzerland)*, vol. 20, no. 7, pp. 1853–1872, 2020, doi: 10.3390/s20071853.

[7] S. Tuli *et al.*, "HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments," *Futur Gener Comput Syst*, vol. 104, pp. 187–200, 2020, doi: 10.1016/j.future.2019.10.043.

[8] Ş. Kolozali *et al.*, "Observing the pulse of a city: A smart city framework for efficient time discovery, federation, and aggregation of data streams," *IEEE Internet Things J*, vol. 6, no. 2, pp. 2651–2668, 2018.

[9] P. Raj and A. C. Raman, *The Internet of Things: Enabling technologies, platforms, and use cases*. CRC press, 2017.

[10] A. Bahga and V. Madisetti, *Cloud computing: A hands-on approach*. CreateSpace Independent Publishing Platform, 2013.

[11] D. Q. Tu, A. S. M. Kayes, W. Rahayu, and K. Nguyen, "IoT streaming data integration from multiple sources," *Computing*, vol. 102, no. 10, pp. 2299–2329, 2020, doi: 10.1007/s00607-020-00830-9.

[12] Z. Mahmood, *Fog computing: concepts, frameworks and technologies*. Springer, 2018.

[13] R. Buyya and S. N. Srirama, *Fog and edge computing: principles and paradigms*. John Wiley & Sons, 2019.

[14] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using LSTM networks," *Comput Ind*, vol. 131, pp. 103498–103509, 2021, doi: 10.1016/j.compind.2021.103498.

[15] E. Badidi and A. Ragmani, "An Architecture for QoS-Aware Fog Service Provisioning," *Procedia Comput Sci*, vol. 170, pp. 411–418, 2020, doi: 10.1016/j.procs.2020.03.083.

[16] A. E. Bagaskara, S. Setyorini, and A. A. Wardana, “Performance Analysis of Message Broker for Communication in Fog Computing,” *ICITEE 2020 - Proc 12th Int Conf Inf Technol Electr Eng*, pp. 98–103, 2020, doi: 10.1109/ICITEE49829.2020.9271733.

[17] W. Hanon and M. A. Salman, “Review the deployment and role of broker in IoT platforms,” in *IICETA 2022 - 5th International Conference on Engineering Technology and its Applications*, 2022, pp. 308–315. doi: 10.1109/IICETA54559.2022.9888675.

[18] S. Sarkar and S. Misra, “Theoretical modelling of fog computing: A green computing paradigm to support IoT applications,” *IET Networks*, vol. 5, no. 2, pp. 23–29, 2016, doi: 10.1049/iet-net.2015.0034.

[19] R. Rezapour, P. Asghari, H. H. S. Javadi, and S. Ghanbari, “Security in fog computing: A systematic review on issues, challenges and solutions,” *Comput Sci Rev*, vol. 41, p. 100421, 2021, doi: 10.1016/j.cosrev.2021.100421.

[20] V. Cardellini, F. Lo Presti, M. Nardelli, and G. R. Russo, “Runtime Adaptation of Data Stream Processing Systems: The State of the Art,” *ACM Comput Surv*, vol. 54, no. 11, pp. 1–36, 2022, doi: 10.1145/3514496.

[21] V. N. Pham, G. W. Lee, V. Nguyen, and E. N. Huh, “Efficient solution for large-scale iot applications with proactive edge-cloud publish/subscribe brokers clustering,” *Sensors*, vol. 21, no. 24, pp. 8232–8253, 2021, doi: 10.3390/s21248232.

[22] A. Malki, E. S. Atlam, and I. Gad, “Machine learning approach of detecting

anomalies and forecasting time-series of IoT devices," *Alexandria Eng J*, vol. 61, no. 11, pp. 8973–8986, 2022, doi: 10.1016/j.aej.2022.02.038.

[23] Y. Deng, C. Han, J. Guo, and L. Sun, "Temporal and spatial nearest neighbor values based missing data imputation in wireless sensor networks," *Sensors*, vol. 21, no. 5, pp. 1–24, 2021, doi: 10.3390/s21051782.

[24] J. Ahn and B. M. Lee, "Smart Edge Broker for Location-Based Transfer between Services and Distributed Data in IoT Smart Services," *Mob Inf Syst*, vol. 2020, no. Jul, pp. 1–12, 2020, doi: 10.1155/2020/8896252.

[25] P. Cardoso, J. Moura, and R. Marinheiro, "Software-Defined Elastic Provisioning of IoT Edge Computing Virtual Resources," *arXiv Prepr arXiv200311999*, vol. 1, pp. 1–27, 2020, doi: 10.48550/arXiv.2003.11999.

[26] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities," *IEEE Internet Things J*, vol. 5, no. 2, pp. 696–707, 2018, doi: 10.1109/JIOT.2017.2747214.

[27] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, "Deep Reinforcement Learning-based Scheduling in Edge and Fog Computing Environments," *arXiv Prepr arXiv230907407*, 2023.

[28] N. Sai Lohitha and M. Pounambal, "Integrated publish/subscribe and push-pull method for cloud based IoT framework for real time data processing," *Meas Sensors*, vol. 27, p. 100699, 2023, doi: 10.1016/j.measen.2023.100699.

[29] K. Yang, S. Kpotufe, and N. Feamster, "An Efficient One-Class SVM for

Anomaly Detection in the Internet of Things," *arXiv Prepr arXiv210411146*, 2021, [Online]. Available: http://arxiv.org/abs/2104.11146

[30] A. Al Shorman, H. Faris, and I. Aljarah, "Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT botnet detection," *J Ambient Intell Humaniz Comput*, vol. 11, no. 7, pp. 2809–2825, 2020, doi: 10.1007/s12652-019-01387-y.

[31] R. M. Aziz, M. F. Baluch, S. Patel, and P. Kumar, "A Machine Learning Based Approach to Detect the Ethereum Fraud Transactions with Limited Attributes," *Karbala Int J Mod Sci*, vol. 8, no. 2, pp. 262–274, 2022, doi: 10.33640/2405-609X.3229.

[32] M. M. Rahman, Y. Ghasemi, E. Suley, Y. Zhou, S. Wang, and J. Rogers, "Machine Learning Based Computer Aided Diagnosis of Breast Cancer Utilizing Anthropometric and Clinical Features," *Irbm*, vol. 42, no. 4, pp. 215–226, 2021, doi: 10.1016/j.irbm.2020.05.005.

[33] H. Saleh, S. F. Abd-El Ghany, H. Alyami, and W. Alosaimi, "Predicting Breast Cancer Based on Optimized Deep Learning Approach," *Comput Intell Neurosci*, vol. 2022, 2022, doi: 10.1155/2022/1820777.

[34] B. Nicula, M. Dascalu, N. N. Newton, E. Orcutt, and D. S. McNamara, "Automated paraphrase quality assessment using language models and transfer learning," *Computers*, vol. 10, no. 12, p. 166, 2021, doi: 10.3390/computers10120166.

[35] D. Baby, S. J. Devaraj, J. Hemanth, and M. M. Anishin Raj, "Leukocyte classification based on feature selection using extra trees classifier: A transfer

learning approach," *Turkish J Electr Eng Comput Sci*, vol. 29, no. 8, pp. 2742–2757, 2021, doi: 10.3906/elk-2104-183.

[36] M. Desai and M. Shah, "An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN)," *Clin eHealth*, vol. 4, pp. 1–11, 2021, doi: 10.1016/j.ceh.2020.11.002.

[37] H. Rajaguru and S. R. Sannasi Chakravarthy, "Analysis of decision tree and k-nearest neighbor algorithm in the classification of breast cancer," *Asian Pacific J Cancer Prev*, vol. 20, no. 12, pp. 3777–3781, 2019, doi: 10.31557/APJCP.2019.20.12.3777.

[38] H. Saoud, A. Ghadi, M. Ghailani, and B. A. Abdelhakim, "Using Feature Selection Techniques to Improve the Accuracy of Breast Cancer Classification," in *Lecture Notes in Intelligent Transportation and Infrastructure*, 2019, vol. Part F1405, pp. 307–315. doi: 10.1007/978-3-030-11196-0_28.

[39] D. A. Omondiagbe, S. Veeramani, and A. S. Sidhu, "Machine Learning Classification Techniques for Breast Cancer Diagnosis," in *IOP Conference Series: Materials Science and Engineering*, 2019, vol. 495, no. 1, p. 12033. doi: 10.1088/1757-899X/495/1/012033.

[40] E. S. Kumar, C. S. Bindu, and S. Madhu, "Deep convolutional neural network-based analysis for breast cancer histology images," in *Machine Learning and Deep Learning in Efficient time Applications*, 2020, pp. 168–186. doi: 10.4018/978-1-7998-3095-5.ch008.

[41] M. Jayson Baucas and P. Spachos, “Fog and IoT-based Remote Patient Monitoring Architecture Using Speech Recognition,” in *Proceedings - IEEE Symposium on Computers and Communications*, 2020, vol. 2020-July, pp. 1–6. doi: 10.1109/ISCC50000.2020.9219649.

[42] A. K. Sangaiah and S. C. Mukhopadhyay, *Intelligent IoT systems in personalized health care*. Academic Press, 2020.

[43] H. G. C. Ferreira, E. D. Canedo, and R. T. De Sousa, “IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and arduino,” in *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, 2013, pp. 53–60.

[44] B. Wukkadada, K. Wankhede, R. Nambiar, and A. Nair, “Comparison with HTTP and MQTT in Internet of Things (IoT),” in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018, pp. 249–253.

[45] Q. Huangpeng and R. Othman Yahya, “Distributed IoT services placement in fog environment using optimization-based evolutionary approaches,” *Expert Syst Appl*, vol. 237, p. 121501, 2024, doi: 10.1016/j.eswa.2023.121501.

[46] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, and B. Qureshi, “An overview of iot sensor data processing, fusion, and analysis techniques,” *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–23, 2020, doi: 10.3390/s20216076.

[47] M. Humayoun *et al.*, “From Cloud Down to Things: An Overview of

Machine Learning in Internet of Things," in *2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2023, pp. 1–5.

[48] C. Martín, D. Garrido, L. Llopis, B. Rubio, and M. Díaz, "Facilitating the monitoring and management of structural health in civil infrastructures with an Edge/Fog/Cloud architecture," *Comput Stand Interfaces*, vol. 81, p. 103600, 2022, doi: 10.1016/j.csi.2021.103600.

[49] K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the internet of things," *Sensors (Switzerland)*, vol. 19, no. 8, pp. 1–27, 2019, doi: 10.3390/s19081955.

[50] Z. Benamor, Z. A. Seghir, M. Djezzar, and M. Hemam, "A comparative study of machine learning algorithms for intrusion detection in IoT networks," *Rev d'Intelligence Artif*, vol. 37, no. 3, pp. 567–576, 2023, doi: 10.18280/ria.370305.

[51] S. Nanglia, M. Ahmad, F. Ali Khan, and N. Z. Jhanjhi, "An enhanced Predictive heterogeneous ensemble model for breast cancer prediction," *Biomed Signal Process Control*, vol. 72, p. 103279, 2022, doi: 10.1016/j.bspc.2021.103279.

[52] F. Hussain *et al.*, "A framework for malicious traffic detection in iot healthcare environment," *Sensors*, vol. 21, no. 9, p. 3025, 2021, doi: 10.3390/s21093025.

[53] A. Bahga and V. Madisetti, "Internet of Things: A Hands-On Approach (Edition: 1)," *Sl Vpt, Univ Press Priv Ltd*, 2015.

[54] J. Wang, W. Yi, M. Yang, J. Ma, S. Zhang, and S. Hao, “Enhance the trust between IoT devices, mobile apps, and the cloud based on blockchain,” *J Netw Comput Appl*, vol. 218, p. 103718, 2023, doi: 10.1016/j.jnca.2023.103718.

[55] A. Heidari, N. J. Navimipour, M. A. J. Jamali, and S. Akbarpour, “A hybrid approach for latency and battery lifetime optimization in IoT devices through offloading and CNN learning,” *Sustain Comput Informatics Syst*, vol. 39, p. 100899, 2023, doi: 10.1016/j.suscom.2023.100899.

[56] S. Wu, J. Han, C. Cai, and Q. Wang, “Topology identification method of low-voltage distribution network based on measurement data of IoT devices,” *Energy Reports*, vol. 9, pp. 370–376, 2023, doi: 10.1016/j.egyr.2023.04.114.

[57] A. Arman, P. Bellini, D. Bologna, P. Nesi, G. Pantaleo, and M. Paolucci, “Automating IoT data ingestion enabling visual representation,” *Sensors*, vol. 21, no. 24, p. 8429, 2021.

[58] M. A. Kandi, D. E. Kouicem, M. Doudou, H. Lakhlef, A. Bouabdallah, and Y. Challal, “A decentralized blockchain-based key management protocol for heterogeneous and dynamic IoT devices,” *Comput Commun*, vol. 191, pp. 11–25, 2022, doi: 10.1016/j.comcom.2022.04.018.

[59] F. Samie, L. Bauer, and J. Henkel, “From cloud down to things: An overview of machine learning in internet of things,” *IEEE Internet Things J*, vol. 6, no. 3, pp. 4921–4934, 2019.

[60] G. Ortiz, M. Zouai, O. Kazar, A. Garcia-de-Prado, and J. Boubeta-Puig,

"Atmosphere: Context and situational-aware collaborative IoT architecture for edge-fog-cloud computing," *Comput Stand Interfaces*, vol. 79, 2022, doi: 10.1016/j.csi.2021.103550.

[61] H. Sabireen and V. Neelanarayanan, "A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges," *ICT Express*, vol. 7, no. 2, pp. 162–176, 2021, doi: 10.1016/j.icte.2021.05.004.

[62] A. Kaur, P. Singh, and A. Nayyar, "Fog computing: Building a road to IoT with fog analytics," *Fog Data Anal IoT Appl Next Gener Process Model with State Art Technol*, pp. 59–78, 2020.

[63] M. F. Nurnoby and T. Helmy, "A Efficient time Deep Learning-based Smart Surveillance Using Fog Computing: A Complete Architecture," *Procedia Comput Sci*, vol. 218, pp. 1102–1111, 2022, doi: 10.1016/j.procs.2023.01.089.

[64] A. Alatram, L. F. Sikos, M. Johnstone, P. Szewczyk, and J. J. Kang, "DoS/DDoS-MQTT-IoT: A dataset for evaluating intrusions in IoT networks using the MQTT protocol," *Comput Networks*, vol. 231, p. 109809, 2023, doi: 10.1016/j.comnet.2023.109809.

[65] G. Beniwal and A. Singhrova, "'A systematic literature review on IoT gateways,'" *J King Saud Univ - Comput Inf Sci*, vol. 34, no. 10, pp. 9541–9563, 2022, doi: 10.1016/j.jksuci.2021.11.007.

[66] B. M. Alencar, J. P. Canário, R. Lobão Neto, C. Prazeres, A. Bifet, and R. A. Rios, "Fog-DeepStream: A new approach combining LSTM and Concept Drift for data stream analytics on Fog computing," *Internet of Things*

(Netherlands), vol. 22, p. 100731, 2023, doi: 10.1016/j.iot.2023.100731.

[67] S. Mirampalli, R. Wankar, and S. N. Srirama, "Evaluating NiFi and MQTT based serverless data pipelines in fog computing environments," *Futur Gener Comput Syst*, 2023, doi: 10.1016/j.future.2023.09.014.

[68] G. C. Hillar, *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[69] Y. Sasaki, T. Yokotani, and H. Mukai, "MQTT over VLAN for Reduction of Overhead on Information Discovery," in *2019 International Conference on Information Networking (ICOIN)*, 2019, pp. 354–356.

[70] S. Pooja, D. V Uday, U. B. Nagesh, and S. G. Talekar, "Application of MQTT protocol for real time weather monitoring and precision farming," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2017, pp. 1–6.

[71] B. Mishra and A. Kertesz, "The use of MQTT in M2M and IoT systems: A survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[72] D. Soni and A. Makwana, "A survey on mqtt: a protocol of internet of things (iot)," in *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*, 2017, vol. 20, pp. 173–177.

[73] H. H. Alshammari, "The internet of things healthcare monitoring system based on MQTT protocol," *Alexandria Eng J*, vol. 69, pp. 275–287, 2023, doi: 10.1016/j.aej.2023.01.065.

[74] Z. Y. M. Yusoff, M. K. Ishak, and L. A. B. Rahim, “A java servlet based transaction broker for internet of things edge device communications,” *Bull Electr Eng Informatics*, vol. 11, no. 1, pp. 488–497, 2022, doi: 10.11591/eei.v11i1.3455.

[75] V. Seoane, C. Garcia-Rubio, F. Almenares, and C. Campo, “Performance evaluation of CoAP and MQTT with security support for IoT environments,” *Comput Networks*, vol. 197, p. 108338, 2021, doi: 10.1016/j.comnet.2021.108338.

[76] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, “Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset),” in *Lecture Notes in Networks and Systems*, 2021, vol. 180, pp. 73–84. doi: 10.1007/978-3-030-64758-2_6.

[77] J. Sidna, B. Amine, N. Abdallah, and H. El Alami, “Analysis and evaluation of communication protocols for IoT applications,” in *Proceedings of the 13th international conference on intelligent systems: theories and applications*, 2020, pp. 1–6.

[78] D. Silva, L. I. Carvalho, J. Soares, and R. C. Sofia, “A performance analysis of internet of things networking protocols: Evaluating MQTT, CoAP, OPC UA,” *Appl Sci*, vol. 11, no. 11, p. 4879, 2021.

[79] M. A. Tariq, M. Khan, M. T. Raza Khan, and D. Kim, “Enhancements and challenges in coap—a survey,” *Sensors*, vol. 20, no. 21, p. 6391, 2020.

[80] S. V, V. A, and S. Pattar, “MQTT based Secure Transport Layer Communication for Mutual Authentication in IoT Network,” *Glob*

Transitions Proc, vol. 3, no. 1, pp. 60–66, 2022, doi: 10.1016/j.gltp.2022.04.015.

[81] B. Mishra, "Performance evaluation of MQTT broker servers," in *International Conference on Computational Science and Its Applications*, 2018, pp. 599–609.

[82] S. Ohno, K. Terada, T. Yokotani, and K. Ishibashi, "Distributed MQTT broker architecture using ring topology and its prototype," *IEICE Commun Express*, vol. 10, no. 8, pp. 582–586, 2021, doi: 10.1587/comex.2021xbl0096.

[83] Q. Xu, Q. Zhang, B. Yu, N. Shi, C. Wang, and W. He, "Decentralized and expressive data publish-subscribe scheme in cloud based on attribute-based keyword search," *J Syst Archit*, vol. 119, p. 102274, 2021, doi: 10.1016/j.sysarc.2021.102274.

[84] B. Shakya, A. Bruce, and I. MacGill, "Survey based characterisation of energy services for improved design and operation of standalone microgrids," *Renew Sustain Energy Rev*, vol. 101, pp. 493–503, 2019, doi: 10.1016/j.rser.2018.11.016.

[85] A. Agrawal, S. Choudhary, A. Bhatia, and K. Tiwari, "Pub-SubMCS: A privacy-preserving publish–subscribe and blockchain-based mobile crowdsensing framework," *Futur Gener Comput Syst*, vol. 146, pp. 234–249, 2023, doi: 10.1016/j.future.2023.04.018.

[86] M. M. R. Perumalla, S. K. Singh, A. Khamparia, A. Goyal, and A. Mishra, "Machine Learning Frameworks and Algorithms for Fog and Edge

Computing," *Fog, Edge, Pervasive Comput Intell IoT Driven Appl*, pp. 67–84, 2020, doi: 10.1002/9781119670087.ch4.

[87] Z. Li *et al.*, "Trade-off analysis between delay and throughput of RAN slicing for smart grid," *Comput Commun*, vol. 180, pp. 21–30, 2021, doi: 10.1016/j.comcom.2021.07.004.

[88] T. Orihara and T. Tanno, "Inter-response time shaping by percentile schedule with discrete trial procedure," *Learn Motiv*, vol. 81, p. 101866, 2023, doi: 10.1016/j.lmot.2022.101866.

[89] A. Garnaev and W. Trappe, "A multiple access channel game with users implementing throughput and latency metrics," *ICT Express*, 2023, doi: 10.1016/j.icte.2023.04.004.

[90] C. Mwase, Y. Jin, T. Westerlund, H. Tenhunen, and Z. Zou, "Communication-efficient distributed AI strategies for the IoT edge," *Futur Gener Comput Syst*, vol. 131, pp. 292–308, 2022.

[91] G. Y. Gombolay *et al.*, "Review of machine learning and artificial intelligence (ML/AI) for the pediatric neurologist," *Pediatr Neurol*, 2023.

[92] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digit Commun Networks*, vol. 4, no. 3, pp. 161–175, 2018, doi: 10.1016/j.dcan.2017.10.002.

[93] S. Dalwinder, S. Birmohan, and K. Manpreet, "Simultaneous feature weighting and parameter determination of Neural Networks using Ant Lion

Optimization for the classification of breast cancer," *Biocybern Biomed Eng*, vol. 40, no. 1, pp. 337–351, 2020, doi: 10.1016/j.bbe.2019.12.004.

[94] P. Dadheech, V. Kalmani, S. R. Dogiwal, V. K. Sharma, A. Kumar, and S. K. Pandey, "Breast cancer prediction using supervised machine learning techniques," *J Inf Optim Sci*, vol. 44, no. 3, pp. 383–392, 2023, doi: 10.47974/jios-1348.

[95] D. A. Gzar, A. M. Mahmood, and M. K. Abbas, "A Comparative Study of Regression Machine Learning Algorithms: Tradeoff Between Accuracy and Computational Complexity," *Math Model Eng Probl*, vol. 9, no. 5, pp. 1217–1224, 2022, doi: 10.18280/mmep.090508.

[96] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for ai-enabled iot devices: A review," *Sensors (Switzerland)*, vol. 20, no. 9, p. 2533, 2020, doi: 10.3390/s20092533.

[97] A. A. S. Aliar, J. Yesudhasan, M. Alagarsamy, K. Anbalagan, J. Sakkarai, and K. Suriyan, "A comprehensive analysis on IoT based smart farming solutions using machine learning algorithms," *Bull Electr Eng Informatics*, vol. 11, no. 3, pp. 1550–1557, 2022, doi: 10.11591/eei.v11i3.3310.

[98] F. A. Pourhosseini, K. Ebrahimi, and M. H. Omid, "Prediction of total dissolved solids, based on optimization of new hybrid SVM models," *Eng Appl Artif Intell*, vol. 126, p. 106780, 2023, doi: 10.1016/j.engappai.2023.106780.

[99] Y. Zheng, J. Luo, J. Chen, Z. Chen, and P. Shang, "Natural gas spot price prediction research under the background of Russia-Ukraine conflict - based

on FS-GA-SVR hybrid model," *J Environ Manage*, vol. 344, p. 118446, 2023, doi: 10.1016/j.jenvman.2023.118446.

[100] V. Martinez-Viol, E. M. Urbano, K. Kampouropoulos, M. Delgado-Prieto, and L. Romeral, "Support vector machine based novelty detection and FDD framework applied to building AHU systems," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2020, vol. 2020-Septe, pp. 1749–1754. doi: 10.1109/ETFA46521.2020.9212088.

[101] X. Xia *et al.*, "GAN-based anomaly detection: A review," *Neurocomputing*, vol. 493, pp. 497–535, 2022.

[102] B. Kumar, A. Sinha, S. Chakrabarti, and O. P. Vyas, "A fast learning algorithm for One-Class Slab Support Vector Machines," *Knowledge-Based Syst*, vol. 228, p. 107267, 2021, doi: 10.1016/j.knosys.2021.107267.

[103] A. Jalalifar, H. Soliman, M. Ruschin, A. Sahgal, and A. Sadeghi-Naini, "A Brain Tumor Segmentation Framework Based on Outlier Detection Using One-Class Support Vector Machine," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2020, vol. 2020-July, pp. 1067–1070. doi: 10.1109/EMBC44109.2020.9176263.

[104] T. B. Dang, D. T. Le, T. D. Nguyen, M. Kim, and H. Choo, "Monotone Split and Conquer for Anomaly Detection in IoT Sensory Data," *IEEE Internet Things J*, vol. 8, no. 20, pp. 15468–15485, 2021, doi: 10.1109/JIOT.2021.3073705.

[105] D. Hu, C. Zhang, T. Yang, and G. Chen, “An Intelligent Anomaly Detection Method for Rotating Machinery Based on Vibration Vectors,” *IEEE Sens J*, vol. 22, no. 14, pp. 14294–14305, 2022, doi: 10.1109/JSEN.2022.3179740.

[106] G. Priyadarshi and B. K. Naik, “Desiccant coated fin tube energy exchanger design optimization implementing KNN-ML tool and adsorption/desorption kinetics analysis using finite difference based transient model,” *Int J Therm Sci*, vol. 192, p. 108422, 2023, doi: 10.1016/j.ijthermalsci.2023.108422.

[107] L. CUI *et al.*, “A method for satellite time series anomaly detection based on fast-DTW and improved-KNN,” *Chinese J Aeronaut*, vol. 36, no. 2, pp. 149–159, 2023, doi: 10.1016/j.cja.2022.05.001.

[108] L. Huang, T. Song, and T. Jiang, “Linear regression combined KNN algorithm to identify latent defects for imbalance data of ICs,” *Microelectronics J*, vol. 131, p. 105641, 2023, doi: 10.1016/j.mejo.2022.105641.

[109] A. Sumayli, “Development of advanced machine learning models for optimization of methyl ester biofuel production from papaya oil: Gaussian process regression (GPR), multilayer perceptron (MLP), and K-nearest neighbor (KNN) regression models,” *Arab J Chem*, vol. 16, no. 7, p. 104833, 2023, doi: 10.1016/j.arabjc.2023.104833.

[110] J. Behera, A. K. Pasayat, H. Behera, and P. Kumar, “Prediction based mean-value-at-risk portfolio optimization using machine learning regression algorithms for multi-national stock markets,” *Eng Appl Artif Intell*, vol. 120, p. 105843, 2023, doi: 10.1016/j.engappai.2023.105843.

[111] O. Lifandali, N. Abghour, and Z. Chiba, “Feature Selection Using a Combination of Ant Colony Optimization and Random Forest Algorithms Applied to Isolation Forest Based Intrusion Detection System,” *Procedia Comput Sci*, vol. 220, pp. 796–805, 2023, doi: 10.1016/j.procs.2023.03.106.

[112] M. A. N. D. Sewwandi, Y. Li, and J. Zhang, “Granule-specific feature selection for continuous data classification using neighborhood rough sets,” *Expert Syst Appl*, vol. 238, p. 121765, 2024, doi: 10.1016/j.eswa.2023.121765.

[113] T. K. Sahoo, A. Negi, and H. Banka, “Dimensionality reduction using PCAs in feature partitioning framework,” in *Statistical Modeling in Machine Learning: Concepts and Applications*, Elsevier, 2022, pp. 269–286. doi: 10.1016/B978-0-323-91776-6.00008-7.

[114] L. Bai, C. Song, X. Zhou, Y. Tian, and L. Wei, “Assessing project portfolio risk via an enhanced GA-BPNN combined with PCA,” *Eng Appl Artif Intell*, vol. 126, p. 106779, 2023, doi: 10.1016/j.engappai.2023.106779.

[115] M. Topolski, “The modified principal component analysis feature extraction method for the task of diagnosing chronic lymphocytic leukemia type b-cll,” *J Univers Comput Sci*, vol. 26, no. 6, pp. 734–746, 2020, doi: 10.3897/jucs.2020.039.

[116] A. Liang, Y. Hu, and G. Li, “The impact of improved PCA method based on anomaly detection on chiller sensor fault detection,” *Int J Refrig*, 2023, doi: 10.1016/j.ijrefrig.2023.09.002.

[117] M. T. H. Kaib, A. Kouadri, M. F. Harkat, A. Bensmail, and M. Mansouri,

"Improving Kernel PCA-based algorithm for fault detection in nonlinear industrial process through fractal dimension," *Process Saf Environ Prot*, 2023, doi: 10.1016/j.psep.2023.09.010.

[118] D. Zhao, Z. Shen, and S. Zhao, "Feature Selection Based on Two-stage Resampling Technique for Imbalanced Dataset," *Procedia Comput Sci*, vol. 221, pp. 316–321, 2023, doi: 10.1016/j.procs.2023.07.043.

[119] P. Supsermpol, V. N. Huynh, S. Thajchayapong, and N. Chiadamrong, "Predicting financial performance for listed companies in Thailand during the transition period: A class-based approach using logistic regression and random forest algorithm," *J Open Innov Technol Mark Complex*, vol. 9, no. 3, p. 100130, 2023, doi: 10.1016/j.joitmc.2023.100130.

[120] X. S. Xianhao Shen, C. Z. Xianhao Shen, Y. Z. Changhong Zhu, and S. N. Yihao Zang, "A Method for Detecting Abnormal Data of Network Nodes Based on Convolutional Neural Network," 電腦學刊, vol. 33, no. 3, pp. 049–058, 2022, doi: 10.53106/199115992022063303004.

[121] F. Khan *et al.*, "Cloud-Based Breast Cancer Prediction Empowered with Soft Computing Approaches," *J Healthc Eng*, vol. 2020, 2020, doi: 10.1155/2020/8017496.

[122] D. Sharma, R. Kumar, and A. Jain, "Breast cancer prediction based on neural networks and extra tree classifier using feature ensemble learning," *Meas Sensors*, vol. 24, p. 100560, 2022, doi: 10.1016/j.measen.2022.100560.

[123] V. Jaiswal, P. Saurabh, U. K. Lilhore, M. Pathak, S. Simaiya, and S. Dalal, "A breast cancer risk predication and classification model with ensemble

learning and big data fusion," *Decis Anal J*, vol. 8, p. 100298, 2023, doi: 10.1016/j.dajour.2023.100298.

[124] T. O. Omotehinwa, D. O. Oyewola, and E. G. Dada, "A Light Gradient-Boosting Machine algorithm with Tree-Structured Parzen Estimator for breast cancer diagnosis," *Healthc Anal*, vol. 4, p. 100218, 2023, doi: 10.1016/j.health.2023.100218.

المستخلص

ظهرت في الآونة الاخير ثورة في مجال استخدام اجهزة انترنت الاشياء, والتي اخذت بالتوسع لتشمل كافة مرافق الحياة والمجالات المتعلقة بحيات الانسان اليومية. كما هو الحال في تطبيقات الرعاية الصحية, التنبؤ بالطقس, والمدن الذكية , وغيرها من مجالات الحياة. يعد انترنت الاشياء من المصادر الغنية بالبيانات المتولدة من كافة اجهزة انترنت الاشياء وعلى مدار اليوم وفي الوقت الحقيقي. هذا الكم الهائل من البيانات يكون غير متجانس , وذات تدفقات مستمرة, وموزعة جغرافيا على نطاق واسع.بالتالي, هي بحاجة الى تقنيات فعالة لإدارة الموارد ,التخزين, المعالجات والتحليل. القيم المفقودة والشذوذ من العيوب التي قد يعاني منها دفق البيانات المتولد من اجهزة انترنت الاشياء والتي تؤدي الى انخفاض الدقة والموثوقية بالنظام, وبالتالي تدهور اداء النظام بصورة عامة.

معظم اجهزة انترنت الاشياء تعاني من نقص بالموارد اللازمة لمعالجة البيانات محليا قرب مصدر انتاجها. لذلك تعتبر الحوسبة السحابة والحافة من الحلول الواعدة في هذا المجال. ولكن للأسف تعاني من البعد عن مصدر انتاج البيانات , نقص في موارد اللازمة للمعالجات في الحافة , قابلية التوسع محدودة, مصادر البيانات تكون غير متجانسة, وانخفاض عرض النطاق الترددي. هذه العيوب سوف تؤدي الى مشاكل في الامان , زمن وصول عالي, استجابة بطيئة, فقدان البيانات, تحميل زائد على الشبكة.

تهدف هذه الاطروحة الى انشاء بنية تكيفية في عقد الضباب, تقوم بتجميع وحدة تحكم ذكية مع خدمات للمعالجات المسبقة ويكون لديهم اشتراك في وسيط . ومن خلال الاستفادة من مزايا الضباب في توفير الوارد وسرعة المعالجات والتخزين, يتم تحسين اداء النظام وزيادة الموثوقية بالاعتماد على مجموعة من مقاييس الاداء. في وحدة التحكم الذكية يتم استخدام مجموعة من خوارزميات التعلم الالي التي يتم دمجها لإنتاج موديلات جديد للتنبؤ بعدد الخدمات الازم نوفرها للنظام الحالي , التنبؤ بسرطان الثدي وتشخيصه (حميد او خبيث) , واخيرا يمكن للمتحكم الذكي اتخاذ القرارات في الوقت الفعلي عندما يكون هناك حمل زائد من قبل البيانات على احدي خدمات النظام بإضافة وتشغيل خدمة مساعدة للخدمة الحالية او عندما تكون عدد الخدمات اكبر من حاجة النظام , يقوم بإيقاف عمل الخدمات.

تمت التجارب العملية على ثلاث قواعد للبيانات وهي : المنزل الذكي , وبيانات مختبرات Berkley, والبيانات الطبية لتشخيص سرطان الثديBreast Cancer . تم استخدام الدقة ومتوسط مربع الخطأ لقياس دقة وجودة النماذج المقترحة. حيث كانت الدقة في اكتشاف الشذوذ ٩٩,٤٤%,و ٩٩,٤٢% . نموذج التنبؤ بعدد الخدمات الواجب تشغيلها في النظام ٠,٠٠٤٣ باستخدام متوسط مربع الخطأ. وفي تشخيص والتوقع بسرطان الثدي , كان دقة النموذج ٩٩,١٢% كما اظهر النموذج تقليل في حجم البيانات وصل الى ٦٨%. البنية التكيفية المقترحة , اظهرت تحسن في مقاييس الاداء من حيث تشغيل متحكم ذكي يراقب مجموعة من الخدمات ويقرر اذا م اكانت تعاني من ازدحام ام لا. التحسن كان في متوسط زمن الوصول ٩١,٦٥%, زمن الاستجابة ٩٧,٣٤%, فقدان البيانات انخفض بمعدل ٨٤,٤٩%, وكانت الانتاجية ٧عملية / الثانية.

جمهورية العراق
وزارة التعليم العالي والبحث العـــــلمي
جامعـــــــــــــــة بابـــــــــل
كليـــــــــة تكنولوجيا المعلومـــــات
قســـــــم البرمجيـــــــــــــــات

تحسين اداء خدمات الضباب باستخدام وسيط متحكم و ذكي

أطروحة مقدمة إلى

مجلس كلية تكنولوجيا المعلومات جامعة بابل استيفاءً جزئياً لمتطلبات درجة دكتوراه الفلسفة في تكنولوجيا المعلومات/ البرمجيات

من قبل
وائل عباس هنون علي

أشراف
استاذ مساعد دكتور مهدي عبد سلمان

١٤٤٥هـ ٢٠٢٣ م