SMART FOG – A COLLABORATIVE APPROACH TO SHARE COMPUTATIONAL POWER OF FOG DEVICES FOR FOG COMPUTING IN SMART CITY IoT NETWORK

स्मार्ट फ़ॉग - स्मार्ट सिटी आईओटी नेटवर्क में फ़ॉग कंप्यूटिंग के लिए फ़ॉग उपकरणों की कम्प्यूटेशनल शक्ति साझा करने के लिए एक सहयोगात्मक दृष्टिकोण

A Thesis

Submitted for the Award of Ph.D. degree

by

NALAWADE SURAJ RAJARAM

नलवडे सूरज राजाराम

Under the Supervision of

Dr. ASHOK KUMAR JETAWAT

Professor
Pacific Academy of Higher Education
& Research University, Udaipur

FACULTY OF COMPUTER ENGINEERING
PACIFIC ACADEMY OF HIGHER EDUCATION AND RESEARCH UNIVERSITY UDAIPUR

2024

DECLARATION

I, **NALAWADE SURAJ RAJARAM** S/o **Mr. NALAWADE RAJARAM ANNA** resident of 4624, Suryoday Niwas, Adarsh Colony, Near Modern High School, Rahimatpur Road, Koregaon. Satara Maharashtra, hereby declare that the work incorporated in the present thesis entitled **"SMART FOG – A COLLABORATIVE APPROACH TO SHARE COMPUTATIONAL POWER OF FOG DEVICES FOR FOG COMPUTING IN SMART CITY IoT NETWORK" ("स्मार्ट फ़ॉग - स्मार्ट सिटी आईओटी नेटवर्क में फ़ॉग कंप्यूटिंग के लिए फ़ॉग उपकरणों की कम्प्यूटेशनल शक्ति साझा करने के लिए एक सहयोगात्मक दृष्टिकोण")** is my own work and is original. This work (in part or in full) has not been submitted to any University for the award of a Degree or a Diploma. I have properly acknowledged the material collected from secondary sources wherever required. I solely own the responsibility for the originality of the entire content.

Date: **(NALAWADE SURAJ RAJARAM)**

Place: Udaipur

FACULTY OF ENGINEERING

PACIFIC ACADEMY OF HIGHER EDUCATION & RESEARCH UNIVERSITY, UDAIPUR

Dr. ASHOK KUMAR JETAWAT
Professor

CERTIFICATE

It gives me immense pleasure in certifying that the thesis entitled **"SMART FOG – A COLLABORATIVE APPROACH TO SHARE COMPUTATIONAL POWER OF FOG DEVICES FOR FOG COMPUTING IN SMART CITY IoT NETWORK" ("स्मार्ट फ़ॉग - स्मार्ट सिटी आईओटी नेटवर्क में फ़ॉग कंप्यूटिंग के लिए फ़ॉग उपकरणों की कम्प्यूटेशनल शक्ति साझा करने के लिए एक सहयोगात्मक दृष्टिकोण")** submitted by **NALAWADE SURAJ RAJARAM** is based on the research work carried out under my guidance. he has completed the following requirements as per Ph.D. regulations of the University.

i. Coursework as per University rules.

ii. Residential requirements of the University.

iii. Regularly presented Half Yearly Progress Report as prescribed by the University.

iv. Published/ accepted a minimum of two research papers in a refereed research journal.

I recommend the submission of the thesis as prescribed/ notified by the University.

Dr. ASHOK KUMAR JETAWAT
Professor
Pacific Academy of Higher
Education & Research University.

Date:

COPYRIGHT

I, **NALAWADE SURAJ RAJARAM**, hereby declare that the Pacific Academy of Higher Education and Research University, Udaipur, Rajasthan shall have the rights to preserve, use and disseminate this dissertation/ thesis **"SMART FOG – A COLLABORATIVE APPROACH TO SHARE COMPUTATIONAL POWER OF FOG DEVICES FOR FOG COMPUTING IN SMART CITY IoT NETWORK" ("स्मार्ट फ़ॉग - स्मार्ट सिटी आईओटी नेटवर्क में फ़ॉग कंप्यूटिंग के लिए फ़ॉग उपकरणों की कम्प्यूटेशनल शक्ति साझा करने के लिए एक सहयोगात्मक दृष्टिकोण")** in print or electronic format for academic/ research purpose.

Date: **(NALAWADE SURAJ RAJARAM)**

Place: Udaipur Signature of the Candidate

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest regards and gratitude to my eminent and esteemed supervisor and guide **Dr. Ashok Kumar Jetawat**, Professor, Department of Computer Engineering, Pacific Academy of Higher Education and Research University, Udaipur, Rajasthan, for giving me inspiration, guidance, valuable suggestions, opinions and correction for the betterment of my research work. I will always be grateful to him. He was always available for help at any point in time. His guidance helped me in the time of research work and writing of my thesis.

As I reflect on this milestone, I am reminded of the profound significance of the support network which surrounds us. To **Dr. Jayshree Jain** madam, I extend my deepest gratitude for her unwavering support and understanding throughout this journey. In this journey, I have witnessed the impact of her support, whether it was through guiding me at different stages of my research work or offering words of encouragement when the path seemed daunting. Her contribution, perhaps less visible, has been just as vital in shaping my academic growth.

I am thankful to the University authorities, especially **Dr. HEMANT KOTHARI**, Dean, Pacific Academy of Higher Education and Research University, Udaipur, Rajasthan, and **Shri Ramesh Agrawal** who guided me at different stages of research work and others in PAHER for their support and encouragement. Without their precious support, it would not be possible to conduct this research, and Ms. **Kusum Madam** and **Dr. Surya** Pacific University Udaipur for helping me in creating the Plagiarism report.

A special note of gratitude to my father, **Shri Rajaram Anna Nalawade**, My mother, **Smt. Sangita Rajaram Nalawade**, whose blessings, love, and support have always aided me in my research endeavors. My lovely wife, **Smt. Ashwini Suraj Nalawade**, my smart son, **Mr. Advait Suraj Nalawade**, and my brother, **Mr. Uday Rajaram Nalawade**, Dr. Manisha Songire (Nalawade) Mrs. Survena Yadav, Mr. Vinod Yadav,

who has always stood by me, I express my gratitude for their love, support, and encouragement.

I am thankful to my grandfather, **Shri Namdev Shivram Deokar**, whose blessings, love, and inspiration always added to my research work.

I would like to thank **Prof. Dasharath Sagare Sir** - Founder President, YSPM's Yashoda Technical Campus, Satara; **Prof. Ajinkya D. Sagare Sir** - Vice President, **Mr. Ganesh K. Survase** Sir - Registrar, **Mr. R. D. Mohite** - Associate Director, YSPM's Yashoda Technical Campus, Satara, **Dr. R. J. Dias** working at Government College of Pharmacy, Karad, Late **Dr. A. B. Mahatme**, Ex Principal, YSPM's Yashoda Technical Campus, Sataras.

I pay all my heartfelt gratitude to my friend Mr. Navnath Pandurang Jadhav working at MAHLE Holding India Pvt Ltd, Pune, Dr. Madhuri Navanath Jadhav Professor at Pune, Mr. Amol Baburao Nalawade, Mphasis Limited, Pune, Mr. Abhijeet Avinash Salunkhe, Branch Manager at Janata Sahakari Bank, Satara, Mrs. Nikita Abhijeet Salunkhe, my colleague Prof. Hakke Dasganu G. Working at Yashoda Technical Campus, Satara, and Pacific University, Udaipur for helping me directly or indirectly making this research work a success.

Thank you to everyone who has directly or indirectly helped me on this beautiful voyage.

NALAWADE SURAJ RAJARAM

PREFACE

Smart cities have emerged as a solution to enhance services and quality of life for residents and visitors. These cities have made significant progress in optimizing resource utilization, promoting environmental protection, improving infrastructure operations and maintenance, and strengthening safety and security measures. Achieving these improvements requires the implementation of new and existing technologies, as well as the application of optimization techniques. Among the technologies supporting smart city applications, the Internet of Things, FOG computing, and cloud computing play vital roles. Integrating these three technologies into a single system, known as the integrated IoT-Fog-Cloud system, offers a sophisticated platform for developing and managing various smart city applications. By leveraging the strengths of IoT gadgets, FOG nodes, and cloud services, this platform enables applications to deliver optimal functionality and performance. The integrated system opens up numerous opportunities for enhancing applications across sectors such as energy, transportation, healthcare, and more. This research work focuses on designing an improvised SMART FOG system, which the key emphasis of the study.

Outline of the Thesis:

The entire research work is divided into six chapters as discussed. The chapterization contains the overview of the proposed SMART FOG protocol-based technique, implementation challenges, task allocation, scheduling techniques, fault tolerance mechanisms, literature review of different authors, result analysis/testing, performance evaluation, and conclusion.

- **Chapter - 1 Introduction:** Serves as a foundation for the research work by highlighting the need for the study. It accomplishes this by referencing various articles and analyzing surveys to establish a solid base for the proposed research. To clarify the background concepts of fog computing, different terminologies related to fog computing are defined and explained. This ensures that readers have a clear understanding of the key terms and concepts associated with the research topic. The chapter also provides an overview of the proposed SMART FOG protocol-based technique. It explains the core

features and functionality of the technique, highlighting how it differs from existing approaches. Additionally, a comparative study is conducted to compare the proposed technique with other relevant methods in the field. This comparison helps to establish the unique benefits and advantages of the SMART FOG protocol-based technique. By encompassing these elements, the first chapter sets the stage for the research work, presenting the need for the study, providing a solid base through article references and survey analysis, clarifying fog computing concepts, and introducing the proposed SMART FOG protocol-based technique along with its comparative study.

- **Chapter -2 Literature Review:** Focuses on reviewing past studies conducted in the research area. It involves examining a broad range of previously completed research projects and providing a comprehensive background of other relevant research works. These sources of literature include journals, articles, research papers, and reputable platforms such as the OpenFog Consortium, IEEE conferences and journals, Springer publications, and online fog computing articles and resources. By conducting this review, the chapter aims to gather existing knowledge, identify gaps in the research field, and build upon the work that has already been done. It provides a critical analysis and synthesis of the literature, highlighting key findings, methodologies, and advancements in fog computing and related domains. The review of the literature serves several purposes. Firstly, it helps to establish the current state of the research area, providing a context for the proposed study. Secondly, it helps the researcher identify research gaps or areas that require further exploration. By examining the existing literature, the chapter also highlights the strengths and weaknesses of previous approaches, leading to insights and inspiration for the proposed research. The sources of literature mentioned, such as the OpenFog Consortium, IEEE, Springer, and online fog computing articles and resources, represent reputable and authoritative platforms in the field. By consulting these sources, the chapter ensures a comprehensive and reliable review of the existing literature, contributing to the overall credibility and validity of the research project.

- **Chapter -3 Research Methodology:** This is dedicated to describing the methodology used in the research project. It primarily focuses on the architecture of the proposed system, including the use of block diagrams to visualize the system's structure. The chapter provides a detailed explanation of the different layers within the architecture, highlighting their functions and interactions. In addition to the system architecture, the chapter also explores the various technologies employed in the implementation of the proposed system. It delves into the specifics of these technologies, discussing their relevance and suitability for the project. The methodology chapter also outlines the research methods employed in the study. It mentions the use of questionnaires or surveys to gather data and insights from relevant stakeholders or experts in the field. These methods help in understanding the requirements, challenges, and expectations associated with the proposed system. By gathering feedback through questionnaires, the research project can align its objectives with the needs of the intended users or beneficiaries. Furthermore, the chapter addresses any gaps or open challenges that were identified during the literature review. It highlights how these gaps are addressed or resolved through the proposed research. The focus is on designing and developing the proposed system to bridge these gaps and overcome challenges identified in previous studies.

- **Chapter - 4 SMART FOG-based Technique:** Focuses on the implementation of the proposed system. The chapter discusses the total work done in the system and outlines the next steps and milestones to be achieved. It also addresses the challenges encountered during the selection of communication protocols and security measures for each layer of communication. The sharing of computational power between IoT devices and fog devices is identified as a challenging aspect, and an improvised method is proposed to enable this sharing. The proposed SMART FOG protocol-based technique aims to execute tasks in the fog environment to avoid latency issues associated with sending requests to cloud centers.

- **Chapter – 5 Allocation and Scheduling of Computational Power:** The focus is on the allocation and scheduling of computational resources shared with IoT devices. The chapter explores different techniques of resource allocation and scheduling, identifying the most efficient ones suitable for fog computing. The current work is tested according to the proposed system, and the results are evaluated to meet the objectives of the research. The evaluation specifically assesses the impact of the proposed work on latency issues in the existing system. Testing and evaluation are crucial for validating the hypothesis, which centers around implementing the SMART FOG protocol-based technique to create a fog environment that shares computational power with IoT devices.

- **Chapter – 6 Conclusion and Future Work:** Provides a summary of the research work and its outcomes in comparison to the expected results defined during the design phase. A detailed analysis is conducted to project future possibilities and enhancements to the system resulting from the study. The chapter also highlights key challenges and issues that warrant further investigation for future development. This chapter serves as a conclusion to the research, summarizing its findings and suggesting avenues for future research and improvement.

In conclusion, based on the evaluation of various accuracy parameters, it can be inferred that the MLP classifier and Logistic Regression are the most suitable classification algorithms for resource allocation and task offloading in a SMART FOG environment. These classifiers consistently outperform the others and demonstrate their effectiveness in achieving accurate and reliable results.

INDEX

Publish Research Papers

Plagiarism Report

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBRIVATIONS

ACK	:	Acknowledgment
AMQP	:	Advanced Message Queuing Protocol
ANN	:	Artificial Neural Networks
CCTV	:	Closed-Circuit Television
CoAP	:	Constrained Application Protocol
CON	:	Confirmable Message
CPU	:	Central Processing Unit
CSP	:	Cloud Service Providers
DCPS	:	Data-Centric Publish-Subscribe
DDS	:	Data Distribution Service
DLRL	:	Data Local Reconstruction Layer
DOTS	:	Dynamic Optimization of Time Sequences
EDA	:	Estimation of Distribution Algorithm
EDF	:	Earliest Deadline First
LFC	:	Least Slack Time
FCFS	:	First-Come, First-Served
FLPSO	:	Fuzzy Logic and Particle Swarm Optimization
FPFTS	:	Fuzzy-Possibilistic Fuzzy Time Series
H2H	:	Human-to-Human
HAN	:	Home-Area Network
HH	:	Hybrid Heuristic
HTP	:	Hypertext Transfer Protocol
IACO	:	Improved Ant Colony Optimization
ICT	:	Information and Communications Technology
IEEE	:	Institute of Electrical and Electronics Engineers
ILP	:	Integer Linear Programming
IoE	:	Internet of Energy
IoMT	:	Internet of Medical Things
IoP	:	Internet of People
IoS	:	Internet of Things

IP	:	Internet Protocol
IPSO	:	Improved Particle Swarm Optimization
IT	:	Information Technology
ITS	:	Intelligent Transportation System
KNN	:	K-Nearest Neighbor
LAN	:	Local Area Network
LoRaWAN	:	WAN Long Range Wide Area Network
LR	:	Logistic Regression
LWM2M	:	Light-Weight Machine-To-Machine Communication
M2M	:	Machine-to-Machine
MAPE-K	:	Monitor, Analyze, Plan, Execute, and Knowledge
MCC	:	Matthews Correlation Coefficient
MCCV	:	Minimum Critical-Cycle Variance
MEC	:	Mobile Edge Computing
MILP	:	Mixed Integer Linear Programming
MIPs	:	Million Instructions Per Seconds
ML	:	Machine Learning
MLP	:	Multilayer Perceptron
MQTT	:	Message Queuing Telemetry Transport
MTC	:	Machine Type Communication
NCA	:	Network Computing and Applications
NFC	:	Near Field Communication
NFV	:	Network Function Virtualization
NLP	:	Natural Language Processing
NON	:	Non-confirmable
PERA	:	Packetized Ensemble Resource Allocation
PRC	:	Precision-Recall Curve
PTPN	:	Preemptive Task Priority Network
PTZ	:	Pan-Tilt-Zoom
QoS	:	Quality of Service
RFID	:	Radio Frequency Identification
RR	:	Round Robin

RST	:	Representational State Transfer
SC	:	Smart Cities
SDN	:	Software-Defined Networking
SEM	:	Structural Equation Modelling
SG	:	Smart Grid
SIoT	:	Social Internet of Things
SJF	:	Shortest Job First
SLAs	:	Service Level Agreements
SLR	:	Systematic Literature Review
SVM	:	Support Vector Machine
TCP/ IP	:	Transmission Control Protocol and Internet Protocol
TIPS	:	Time-Invariant Power Scheduling
TLS	:	Transport Layer Security
TN	:	True Negative
TP	:	True Positive
UDP	:	User Datagram Protocol
URL	:	Uniform Resource Locator
Wi-Fi	:	Wireless Fidelity
WRR	:	Weighted Round Robin
WSN	:	Wireless Sensor Networks
XMPP	:	Extensible Messaging and Presence Protocol

Chapter-1
Introduction

To improve services and quality of life for citizens and visitors, several cities have recently made progress toward becoming smart cities. These cities now have improved resource utilization, increased environmental protection, enhanced infrastructure operations and maintenance, and robust safety and security measures. To improve services and performance in their various sectors, smart cities rely on implementing new and existing technologies and various optimization techniques. The IoT[1], FOG computing and cloud computing are a few of the technologies assisting smart city applications. These three can be combined into one system, an integrated IoT-Fog-Cloud system, to create a sophisticated platform for creating and managing various kinds of smart city applications. With the help of this platform, applications will be able to deliver the best functionality and performance possible by utilizing the best features of IoT gadgets, FOG nodes, and cloud services. Numerous opportunities for improving and optimizing applications in the fields of energy, transportation, healthcare, and other industries will be presented by the use of this strong platform. The improvised SMART FOG system design would be the main focus of this research project.

1.1 Fog Computing

Fog computing is referred to as a distributed computing paradigm that essentially extends the cloud's services to the network's edge. According to Cisco, Fog computing is a continuation of the cloud computing paradigm from the network's core to its edges. It makes networking, computing, and storage between end devices and conventional cloud servers easier. Fog computing uses both the cloud and the edge devices that are situated between end devices and cloud servers to run applications rather than only using the cloud for this purpose. Edge and cloud computing are both benefits of fog computing. While making use of edge devices' proximity to the endpoints, it also uses the cloud's on-demand scalability.

By effectively exploiting the resources present at the edge nodes to do partial computing and by performing filtering operations in the nodes, it essentially lessens the strain on the cloud server. Fog computing is typically confused with two ideas in particular. Mobile Edge Computing and Mobile Cloud Computing are these ideas.

[1]Internet of Things

MCC[2] essentially contends that data processing and storage are carried out on a cloud, away from mobile devices. As a result, it transfers data and processing power from individual mobile devices to the cloud. MEC[3] is a network architecture concept that extends cloud computing capabilities to the edge of the network. It brings computation, storage, and networking resources closer to the end-user or device, reducing latency and improving overall system performance. MEC enables the execution of applications and services at the edge of the network, closer to where the data is generated and consumed. A cloudlet, on the other hand, is a concept related to edge computing and MEC. It refers to a small-scale data center or server cluster deployed at the edge of the network, typically near mobile devices or end-users. Cloudlets provide computational resources and services to nearby devices, offering low-latency access to data and applications.

In comparison, MEC is a broader term that encompasses the concept of cloudlets. MEC involves deploying computing capabilities at various points in the network, such as base stations, access points, or edge routers, whereas a cloudlet specifically refers to a small-scale server cluster. Cloudlets are one implementation of MEC, but MEC can also involve distributed edge computing without using dedicated cloudlet infrastructure. It may be viewed as a more focused version of the cloud computing concept. It resembles a cloud server that is situated at the edge of a mobile network. Fog computing combines these two ideas with some of its characteristics to increase its dependability and utility.

1.2 Fog Computing Architecture

The bandwidth, particularly on cellular networks, is a significant issue with cloud computing. As the IoT grows and more physical devices are wirelessly connected, the issue will only become worse. This issue is resolved by Fog computing, which stores data locally on computers and other gadgets known as fog nodes. Any device having computation, storage, and network connection, such as handheld devices, tablets, PCs, routers, etc., can be used as a fog node.

[2]Mobile Cloud Computing
[3]Mobile Edge Computing

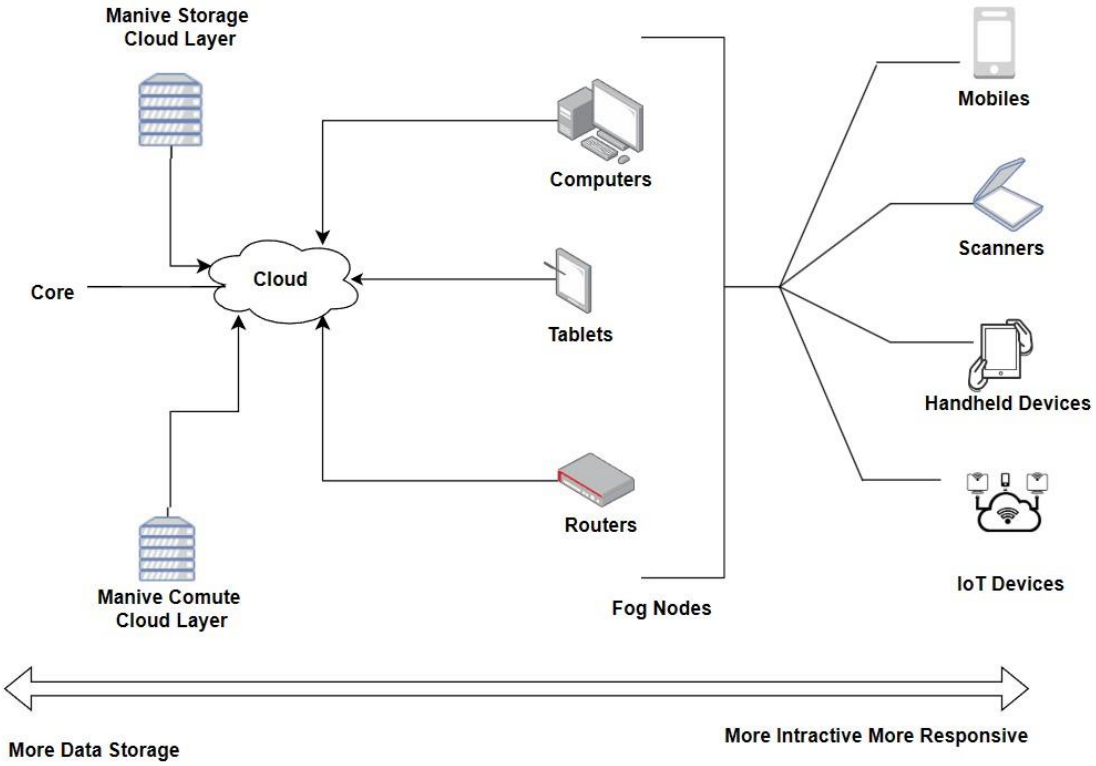

Figure 1.1: Fog Computing Architecture (Lai, 2021)

Figure 1.1 shows Fog-based architecture, fog nodes, also known as edge devices or fog devices, are distributed throughout the network, closer to the data sources and end-users. These fog nodes can be various devices such as routers, switches, access points, edge servers, IoT devices, or other computing resources. The architecture extends the capabilities of cloud computing by providing localized data processing, storage, and analytics at the edge of the network. These fog nodes are controlled by the Fog Data Service, which performs a variety of functions like data reduction, data virtualization, data control and security, and edge analytics. Additionally, data might be uploaded to the cloud for long-term analyses.

Kopras (2023) discussed that the widely adopted cloud computing paradigm is evolving with the integration of fog computing, placing computing nodes in closer proximity to end-users to meet stringent latency requirements. However, effective task offloading, considering transmission and computation energy consumption, poses challenges. Task allocation becomes intricate due to the multitude of arriving tasks with diverse computational, communication, and delay requirements, alongside a variety of computing nodes with differing capabilities. The research work introduces an optimal task allocation procedure aimed at minimizing energy consumption for

wirelessly connected users in a network comprising Fog Nodes located at Access Points and Cloud Nodes. The assignment of Access Points and computing nodes to offloaded tasks, along with Fog Node operating frequencies, is optimized using a Mixed-Integer Nonlinear Programming approach. Realistic energy consumption and delay models, along with their pertinent parameters reflecting device characteristics, are employed. Results indicate the profitability of distributing task processing among multiple Fog Nodes and the cloud, often selecting distinct nodes for transmission and computation. The proposed algorithm demonstrates superior performance, achieving the lowest energy consumption and task rejection rate compared to alternative allocation strategies. Additionally, a heuristic algorithm is presented, decoupling wireless transmission optimization from implemented computations and wired transmission, providing optimal or near-optimal solutions across various scenarios.

1.3 Issues Related to Fog Computing

Cloud computing is expanded by Fog computing, which also affects IoT. These gadgets, also known as fog nodes, can be set up anywhere there is a network connection. Fog computing provides extra storage capabilities at the periphery to handle the demands. As a result, the Fog server must modify its services, which increases administration and maintenance expenses. The operator must also deal with the following problems.

1.3.1 Privacy

Because wireless dominates fog computing, network privacy is a major challenge. The network operator manually creates settings, deploys fog nodes at the edge of the internet, and incurs significant maintenance costs. The exposure of personal information when utilizing networks is receiving more attention. The Fog nodes have easier access to the end consumers. Because of this, Fog nodes gather more sensitive data than faraway clouds. To address these problems, encryption techniques like HAN[4] might be applied.

[4]Home-Area Network

1.3.2 Network Security

Fog networks may be vulnerable to various network-level attacks, such as Denial of Service, Man-in-the-Middle, or network sniffing attacks. It is crucial to implement robust network security measures, including firewalls, intrusion detection systems, and secure communication protocols, to detect and prevent these attacks and protect the integrity and availability of the network. Fog nodes and IoT devices connected to the fog network can be targets for exploitation and compromise. Weak device security, such as default or easily guessable passwords, outdated firmware, or unresolved vulnerabilities, can lead to unauthorized access and control. Implementing secure device configurations, regular security updates, and strong security policies can mitigate these risks.

1.3.3 Network Management

Network management in Fog computing refers to the processes, tools, and strategies used to efficiently control, monitor, and maintain the network infrastructure and devices in a fog computing environment. Fog computing introduces additional complexity to network management due to the distributed nature of the architecture and the heterogeneity of devices involved. Efficient network management in fog computing is crucial to ensure the reliable and secure operation of the fog network. It involves continuous monitoring, optimization of network performance, resource allocation, configuration management, fault handling, and security measures to maintain a robust and scalable fog computing environment. Fog computing environments require continuous monitoring of network performance to ensure efficient and reliable service delivery.

Network administrators need to monitor network traffic, latency, bandwidth utilization, and other performance metrics to identify bottlenecks, congestion, or potential issues that could impact the quality of service. Real-time monitoring tools and analytics are employed to proactively manage and optimize network performance. If SDN[5] and NFV[6] approaches SD are not used, controlling the network, the fog nodes, and the connections between each node would be difficult when linked to heterogeneous devices.

[5] Software-Defined Networking
[6] Network Function Virtualization

1.3.4 Placement of Fog Servers

The placement of fog servers requires careful consideration to ensure optimal performance and cost-effectiveness for the area. One approach to reducing maintenance costs is to thoroughly assess the capabilities and workload of each server node before deployment. Before deploying fog servers, a comprehensive analysis should be conducted to understand the specific needs and requirements of the area. This analysis can involve evaluating factors such as network traffic patterns, latency requirements, data processing demands, and the distribution of edge devices. By examining the workload completed by each server node, it becomes possible to identify the areas where fog servers would be most beneficial. This assessment helps in determining the optimal placement of Fog servers, ensuring that they are strategically located to reduce latency and efficiently process data closer to the source. Additionally, considering the proximity of Fog servers to edge devices can help minimize data transmission delays and enhance real-time processing capabilities. Placing Fog servers near areas with high concentrations of edge devices can improve response times and reduce network congestion. Furthermore, it is essential to assess the scalability and flexibility of Fog server deployments. As the needs of the area evolve, Fog servers should be easily adjustable and expandable to accommodate changing demands.

Effective placement of fog servers involves analyzing the workload of server nodes, considering network traffic patterns, optimizing proximity to edge devices, and ensuring scalability. By carefully considering these factors, it is possible to deploy fog servers in a manner that meets the needs of the area while minimizing maintenance costs.

1.3.5 Delay in Computing

Delays in computing can have significant impacts on the efficiency and performance of various services and applications that rely on data processing. One of the primary reasons for delays is the aggregation of data. When data from multiple sources is collected and combined for processing, it may take time to complete the aggregation process, leading to delays in computing. Additionally, resource overuse can exacerbate the delay issue. Fog servers, which are responsible for processing data locally, may become overloaded with tasks, leading to slower processing times. This

resource constraint can hinder the effectiveness of Fog computing services, making them less responsive and efficient. To address these challenges and reduce delays in computing, it is essential to implement efficient data aggregation techniques. Data should be aggregated in a manner that minimizes processing time while ensuring the accuracy and integrity of the information. This involves optimizing algorithms and strategies for data aggregation to achieve faster processing.

Furthermore, Fog nodes, which are distributed computing resources, should be carefully managed to avoid resource overuse. Scheduling algorithms that prioritize critical tasks and consider the mobility of Fog nodes can help distribute the processing load more effectively. By using a priority and mobility paradigm in scheduling, fog nodes can be dynamically allocated based on their availability and proximity to data sources, reducing delays and improving overall performance. Moreover, optimizing the communication and networking infrastructure between fog nodes and data sources is crucial. Efficient data transmission protocols and network configurations can minimize latency and ensure timely data delivery to Fog servers for processing. Overall, addressing the delay in computing in fog environments requires a comprehensive approach that involves optimizing data aggregation, managing resources effectively, and improving communication infrastructure. By doing so, Fog computing services can offer faster and more responsive data processing, enhancing the overall user experience and system performance.

1.3.6 Energy Consumption

In Fog computing settings where multiple fog nodes are used, the distribution of computing tasks can result in increased energy consumption. To address this issue, reducing energy usage becomes crucial. This can be achieved through various strategies, such as employing energy-efficient hardware components, implementing dynamic resource allocation techniques, utilizing sleep mode and power management features, adopting energy-aware task scheduling algorithms, implementing data compression and aggregation methods, monitoring energy consumption, and exploring the integration of renewable energy sources. By implementing these measures, fog computing environments can minimize energy consumption, improve sustainability, and reduce long-term energy costs.

Fog computing, while offering numerous benefits, faces challenges in terms of network security, privacy, interoperability, resource management, and scalability. Network security and privacy concerns arise due to the distributed nature of fog computing, necessitating robust security mechanisms and encryption techniques to protect sensitive data. The heterogeneity of Fog nodes and edge devices poses interoperability challenges, requiring standardization efforts and protocols for seamless communication and device management. Resource management and load balancing become complex with increasing numbers of devices and applications, necessitating dynamic resource provisioning and monitoring. Additionally, scalability becomes crucial to handle the growing demands of Fog computing, requiring scalable architectures and mechanisms for efficient resource allocation. Addressing these issues through effective security measures, interoperability standards, resource management techniques, and scalable architectures is essential for the successful implementation and operation of Fog computing systems.

Fog computing offers substantial benefits but also faces several issues. Security is a paramount concern as distributing computing resources closer to the edge increases the attack surface. Interoperability challenges persist among diverse IoT devices and fog nodes, hindering seamless data exchange. Resource allocation and load balancing are complex due to dynamic workloads. Privacy issues arise from the vast data generated and processed at the edge. Standardization efforts, security protocols, and robust management systems are crucial to address these challenges and unlock the full potential of fog computing, ensuring it can efficiently support IoT applications while safeguarding data and systems.

1.4 IoT-Based Architectures and Protocols

IoT-based architectures and protocols are essential components that enable the seamless integration and communication of various devices and systems in the IoT ecosystem. These architectures and protocols play a crucial role in ensuring efficient data exchange, interoperability, and security in IoT applications.

1.4.1 Three and Five-Layer Architectures

The IoT is a transformative concept that envisions a network of interconnected devices, sensors, and systems communicating and exchanging data to provide innovative services and valuable insights. In the realm of IoT architecture, two

common frameworks are the Three-Layer Architecture and the Five-Layer Architecture. Accordingly, Figure 1.2 shows Three-Layer Architecture comprises the Perception Layer, where data is collected from IoT devices and sensors; the Network Layer, responsible for facilitating communication between devices and data processing systems; and the Application Layer, where data is processed and transformed into meaningful insights. On the other hand, the Five-Layer Architecture presents a more comprehensive model with the addition of the Middleware Layer, which acts as an intermediary for data normalization and transformation, and the Business Layer, where business logic and decision-making occur based on insights generated from the Application Layer. Both architectures play a crucial role in organizing the flow of data and services within the IoT ecosystem, catering to diverse use cases and providing a structured framework for the successful implementation of IoT solutions. The choice between these architectures depends on the specific requirements and complexity of the IoT application at hand. Three-layer design was first used in the early stages of this field of study. The perception, network, and application layers are its three layers.

The physical layer, which has sensors for sensing and gathering environmental data, is the perception layer. It detects certain physical parameters or locates other intelligent objects in the surrounding area. The network layer is in charge of establishing connections with other intelligent objects, network components, and servers. Additionally, it uses its characteristics to communicate and interpret sensor data.

The Application layer, delivering application-specific services to the user is the responsibility of the application layer. It describes a variety of uses for the IoT, including smart homes, smart cities, and smart health. The three-layer design encapsulates the core concept of the IoT, however research on IoT frequently focuses on its more intricate details, therefore it is insufficient. Because of this, the literature has suggested a lot more layered structures. The first is the five-layer architecture, which also has layers for processing and business. Perception, transport, processing, application, and business layers make up the five layers as shown in Figure 1.2, The perception and application layers play the same role as in a three-layer design.

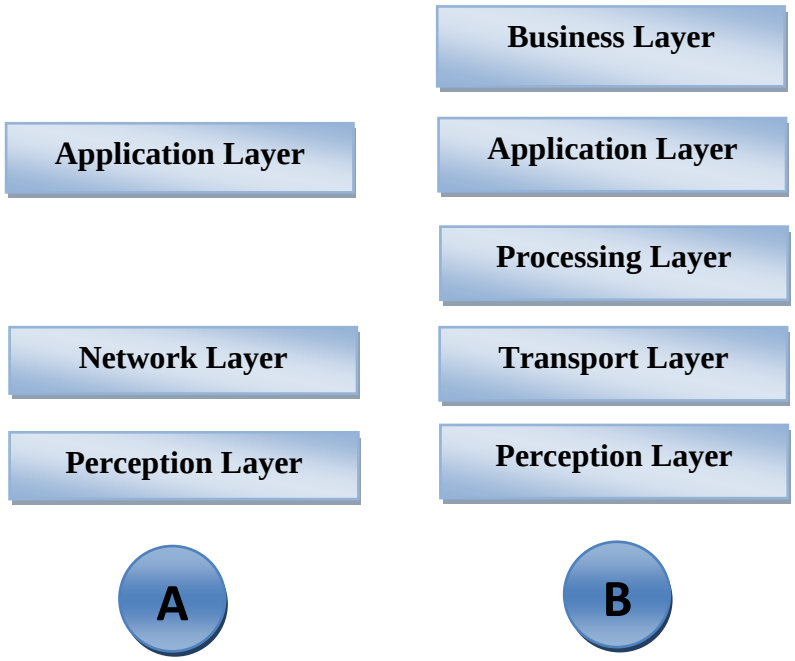

Figure 1.2: Three(A) and Five-Layer(B) Architectures (Lai, 2021)

The Transport layer, through networks including WiFi[7], 3G[8], LAN[9], Bluetooth, RFID[10], and NFC[11], the transport layer moves sensor data from the perception layer to the processing layer and back again. The processing layer, also known as the middleware layer, plays a crucial role in a fog computing architecture. This layer receives a substantial volume of data from the transport layer and performs various tasks such as processing, storing, and analyzing it. It possesses the capability to handle and provide a diverse range of services to the lower tiers. The processing layer leverages different technologies, including modules for big data processing, cloud computing infrastructure, and databases. By utilizing these technologies, the processing layer enhances the overall functionality and performance of the fog computing system. The business layer oversees the whole IoT system, including all applications, revenue streams, and user privacy.

IoT-based architectures and protocols are pivotal in enabling the seamless operation of interconnected devices and systems. These frameworks, including restful APIs,

[7]Wireless Fidelity
[8]3rd Generation
[9]Local Area Network
[10]Radio Frequency Identification
[11]Near Field Communication

MQTT, and CoAP, facilitate efficient communication and data exchange. They play a crucial role in building scalable, interoperable, and secure IoT ecosystems. Selecting the appropriate architecture and protocol depends on specific use cases, emphasizing the need for careful consideration in implementing IoT solutions that align with performance, scalability, and security requirements.

1.4.2 IoT Device Connectivity: Architectures and Protocols

The IoT is only able to function properly and transfer data when all of the connected devices are online and securely linked to a communications network. Standards and protocols for the IoT start to become relevant here. Both IP and non-IP networks can be used to link devices that are part of the IoT. IP network connections are highly complicated and need an increase in memory as well as power from the IoT devices; nevertheless, range is not an issue. On the other hand, non-IP networks have a range constraint and need a far lower amount of power and memory than IP networks do.

1.4.3 IoT Protocol Architecture

The architecture of the IoT is dependent on the functioning and execution of its components in various industries. The IoT is constructed on top of a fundamental process flow, which has two main architectures: a 3-layer architecture and a 5-layer architecture.

1.4.4 Layer IoT Architecture

The most fundamental architecture consists of three distinct layers. It is composed of three layers: the perception layer, the network layer, and the application layer respectively. The physical layer is known as the perception layer, and it is comprised of all of the intelligent sensor-based devices that collect data from their surrounding environment.

The network layer is in charge of establishing connections between the many devices and applications that make up the IoT ecosystem. It is comprised of all of the wireless and wired communication technologies that are currently available. After that, the data is sent to the application layer for processing.

It is the responsibility of the application layer to provide the user with services that are unique to the program. It describes a variety of applications that may be used to implement IoT, including smart homes, smart cities, and health care.

1.4.5 Five-Layer IoT Architecture

The three-layer design has been expanded into a five-layer architecture figure 1.3 shows this by adding two more layers, the processing layer, and the business layer respectively. In the 5-layer design, the perception and application layers function in a manner that is analogous to the 3-layer architecture. Networking technologies such as WiFi, Bluetooth, 3G, RFID, and NFC are utilized by the transport layer to convey the sensor data from the perception layer to the processing layer and vice versa. The processing layer, also known as the middleware layer, is responsible for storing, analyzing, and processing large amounts of data delivered by the transport layer. This layer uses a wide variety of technologies, including databases, cloud computing, and Big Data processing modules. The whole IoT system, including apps, companies, and the privacy of individual users, is managed by the business layer.

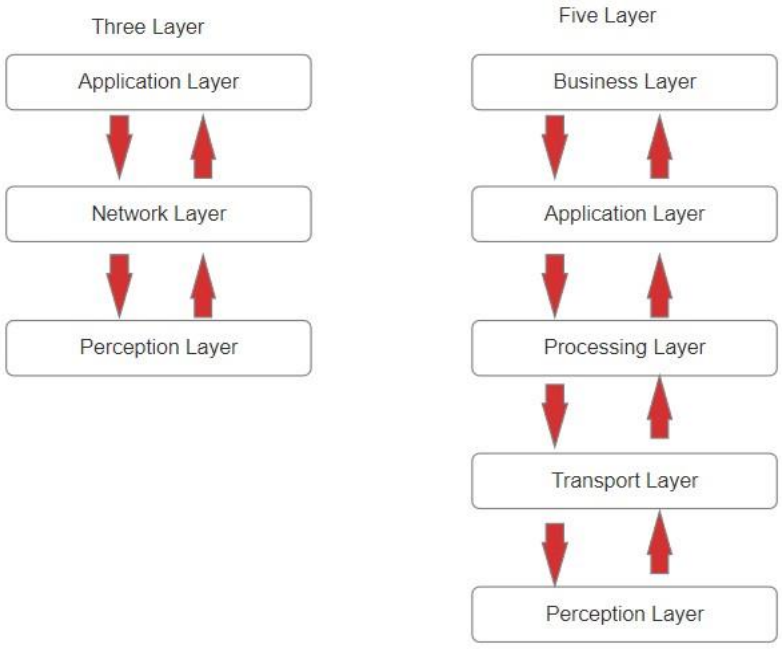

Figure 1.3: IoT Architecture (Lai, 2021)

The Transport layer, through networks including WiFi, 3G, LAN , Bluetooth, RFID, and NFC, the transport layer moves sensor data from the perception layer to the processing layer and back again. IoT architecture can be centralized or decentralized,

depending on the application requirements and scale. The design of the architecture needs to consider scalability, interoperability, data integrity, and energy efficiency to create a robust and reliable IoT ecosystem that can support a wide range of applications and services.

1.4.6 Types of IoT Connections

When it comes to data communication, an IoT system utilizes one of four distinct types of transmission channels. Device-to-device communication, often known as D2D[12] communication enables devices that are physically adjacent to one another to talk to one another via wireless protocols such as Bluetooth, ZigBee, or Z-Wave. By the use of a D2D connection, it is possible to create a link even in the absence of a network in Figure 1.4.

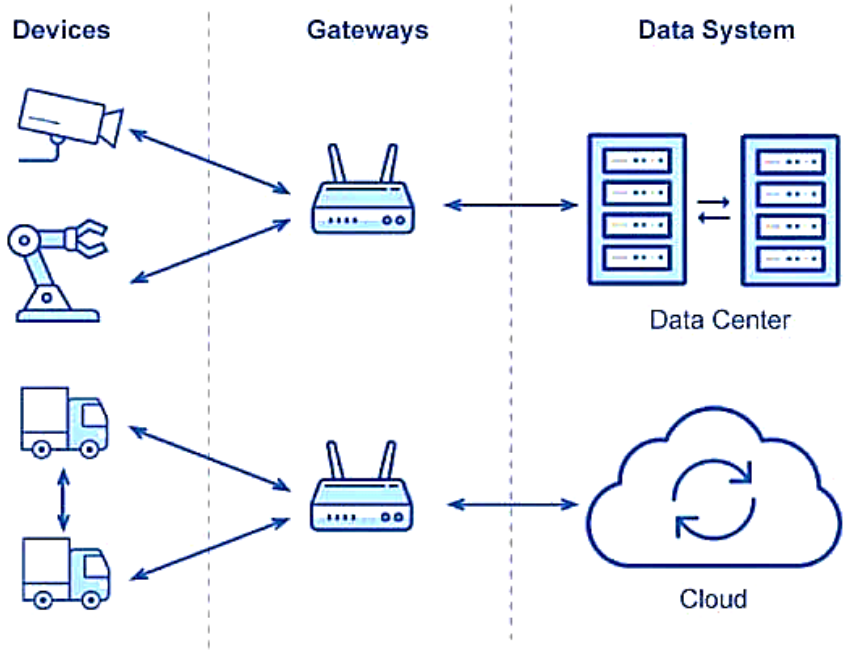

Figure 1.4: Types of IoT Connections (Adel, 2020)

The deployment of an intermediate platform enables communication to occur between devices and gateways at each stage of the network. The majority of the time, gateways are employed for two distinct functions: first, to collect data from sensors and transmit it to the appropriate data system; and second, to evaluate data and transmit it back to the device if any problems are discovered while the data is being analyzed. Both of these functions are essential to the operation of a gateway.

[12]Device-to-Device

When someone refers to "gateway-to-data systems communication," they are referring to the process in which data is sent from a gateway to the appropriate data system. Communication between the many different data systems might take place either within a data center or within the cloud itself. For this sort of connection, the protocols need to be easy to put into action and uncomplicated to include programs that are already in existence. They are required to have high availability, appropriate capacity, and trustworthy disaster recovery capabilities.

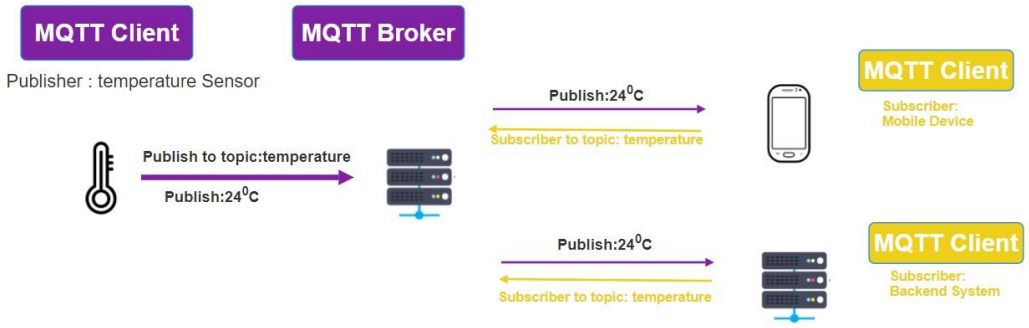

Figure 1.5: Publish / Subscribe Architecture (Ansari, 2018)

Figure 1.5 shows MQTT[13] Publish / Subscribe Architecture there are two types of IoT protocols: Protocols for the network layer: and IoT network protocols that link devices requiring medium to high amounts of electricity to the network. With this protocol, it is possible to communicate data from one end of the network to the other within the network. A few of the most common network protocols for the IoT are HTTP[14], LoRaWAN[15], Bluetooth, and Zigbee.

Data protocols for the IoT: Data protocols for the IoT link low-power IoT devices. These protocols are capable of providing end-to-end communication with the hardware even in the absence of any Internet connection. Connection in the data protocols of the IoT can be accomplished by either a wired or cellular network. MQTT, CoAP[16], AMQP[17], XMPP[18], DDS[19] are some common IoT data protocols.

[13] Message Queuing Telemetry Transport
[14] Hypertext Transfer Protocol
[15] Long Range Wide Area Network
[16] Constrained Application Protocol
[17] Advanced Message Queuing Protocol
[18] Extensible Messaging and Presence Protocol
[19] Data Distribution Service

IoT protocols and network standards: There is a wide variety of IoT protocols available to cater to a variety of applications and needs. Yet, each has its own set of benefits and drawbacks for a variety of IoT use cases. This research work will go through some of the IoT protocols that are the most popular overall.

1.5 Cloud and Fog-Based Architectures

Recently, there has been a shift toward fog computing, a system architecture in which network gateways and sensors perform some of the data processing and analytics. Cloud and fog-based architectures are fundamental paradigms in modern computing. Cloud computing involves centralized data processing in remote data centers, while fog computing disperses computing resources to the edge of the network. Both play key roles in supporting a wide range of applications, balancing data processing, and enabling scalability in an increasingly interconnected world.

Cloud and Fog computing architectures are advanced paradigms for distributed computing. Cloud computing typically involves centralized data centers for resource-intensive tasks, while Fog computing extends this concept to the edge of the network, closer to end-users and devices. Both offer unique advantages.

Cloud computing provides scalability, cost-efficiency, and vast resources for data processing and storage. Fog computing complements this by enabling low-latency, real-time processing and reducing network congestion, making it ideal for applications like IoT and autonomous vehicles. These architectures also foster data security and privacy concerns, which require careful management. Recent research delves into optimizing the integration of Cloud and Fog, ensuring seamless coordination between central and edge resources. This involves developing efficient data transfer, task offloading, and orchestration techniques. Additionally, AI and machine learning are integrated to enhance decision-making processes in these architectures, paving the way for more intelligent, context-aware applications in diverse domains like healthcare, smart cities, and Industry 4.0.

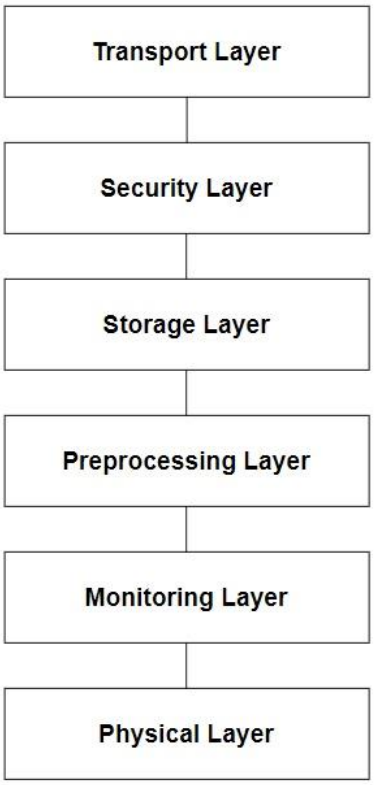

Figure 1.6: Cloud and Fog-Based Architectures (Gupta, 2016)

Figure 1.6 shows illustrate Fog architecture's tiered approach, inserting security, monitoring, and pre-processing layers between the physical and transport levels. Power, resources, and services are all tracked by the monitoring layer. Filtering, processing, and analytics of sensor data are carried out by the pre-processing layer. Data replication, dissemination, and storage are just a few of the storage capabilities offered by the temporary storage layer. The security layer also assures data integrity and privacy and performs encryption and decryption. On the network's edge, monitoring and pre-processing are carried out before data is sent to the cloud. The temporary storage layer in fog computing provides various storage capabilities, including data replication, dissemination, and storage. It serves as a crucial component for managing data within the fog environment. Additionally, the security layer plays a vital role in ensuring data integrity and privacy. It performs encryption and decryption operations to safeguard sensitive information.

Cloud and Fog-based architectures offer versatile solutions for diverse computing needs. Cloud provides centralized, scalable, and reliable data processing, while fog extends computing to the network edge, reducing latency.

1.6 Social IoT

SIoT[20] evaluate social interactions between objects in the same manner that social relationships between people are considered. SIoT represents the integration of IoT technology with social networks and human interactions. It enables smart devices to collect and share data, enhancing user experiences, enabling collaborative decision-making, and fostering a deeper connection between the physical and digital worlds through social engagement and data sharing. The three basic components of a SIoT system are as follows:

- One can navigate the SIoT. We can begin with a single device and browse all of the devices that are linked to it. New devices are simple to find, and services utilize an IoT social network like this.
- There is a requirement for reliability between gadgets
- To analyse the social networks of IoT devices, we can use models similar to those used for researching human social networks.

SIoT refers to the integration of social networking principles and techniques into the IoT paradigm. It combines the power of social interactions and networked devices to enhance communication, collaboration, and information sharing among IoT devices and users.

[20]Social Internet of Things.

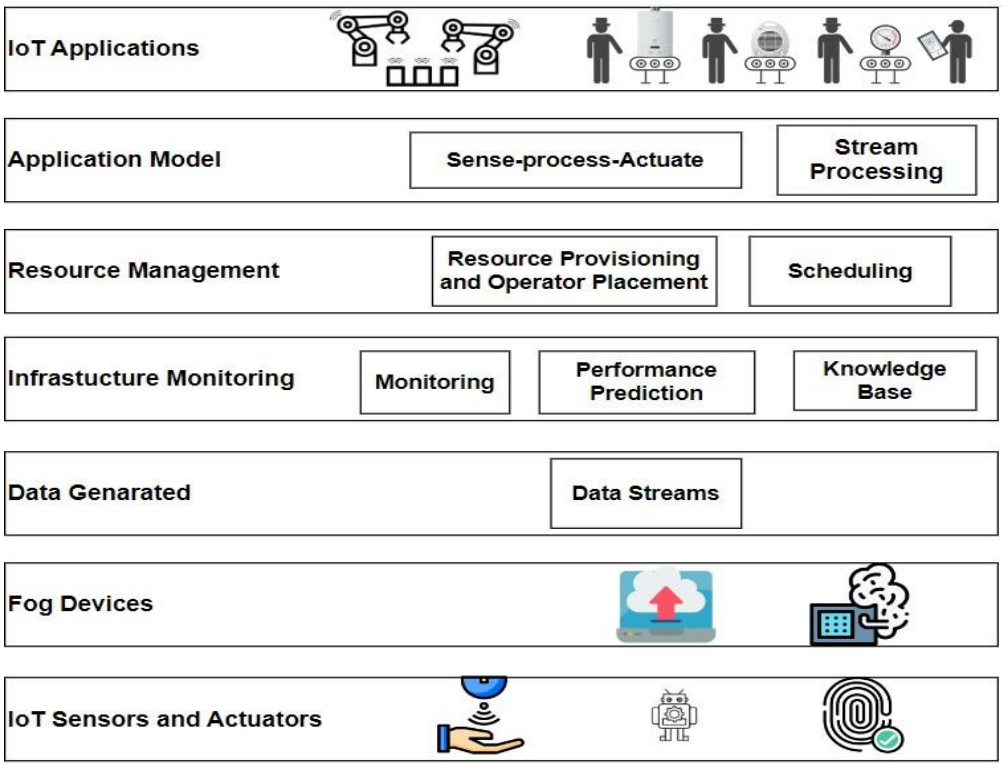

Figure 1.7: IOT & FOG Computing (Gupta, 2016)

In SIoT, devices are considered social entities, and relationships between devices are established based on trust, reputation, and user preferences. This social aspect enables devices to interact and collaborate in a more intelligent and context-aware manner. SIoT offers several benefits. It enables efficient device discovery, where devices can be easily found and connected based on their social relationships. It promotes information sharing and collaborative decision-making among devices, leading to improved efficiency and productivity. Additionally, it enhances user experience by providing personalized and socially influenced services. However, SIoT also presents challenges such as security and privacy concerns, managing complex social networks of devices, and developing appropriate social networking models for IoT environments. Overall, the integration of social aspects into the IoT ecosystem through SIoT has the potential to revolutionize the way devices interact, collaborate, and share information, paving the way for more intelligent and socially aware IoT applications in Figure 1.7.

Social IoT bridges the gap between the physical and digital realms by connecting IoT devices with social interactions. It transforms data sharing, fosters collaborative decision-making, and enriches user experiences. By integrating technology with human connections, Social IoT has the potential to drive innovation, improve communication, and create more personalized and interconnected digital ecosystems.

1.7 Implication of Fog Computing

Fog Computing is a promising paradigm that complements cloud computing by extending computing and storage capabilities to the network edge. This methodology focuses on leveraging fog computing to enhance the infrastructure of a smart city. Smart cities are urban environments that integrate information and communication technologies to optimize the efficiency of various systems, such as transportation, energy, waste management, and public services. By utilizing advanced technologies and data analytics, smart cities aim to improve the quality of life for citizens, enhance sustainability, and enable better resource management.

The smart city concept opens up a new area to explore and It also brings new challenges to implement and design it as a sustainable solution. The smart city has great potential for economic growth and lifting the quality of life in cities. As increasing numbers of citizens migrate to cities, the demand for services and resources continues to increase. The World Bank predicts that over the next two decades, India’s urban population will more than double to 33 % of the total population. The emerging IoT introduces many challenges that cannot be handled by today’s cloud computing. In this research work, we deal with the IoT Environment features like low latency, high distribution, large-scale sensor network, mobility support, and device heterogeneity. This proposed SMART FOG system allows us to create a collaborative environment for IoT networks. In the proposed system, we are going to implement a SMART FOG protocol-based technique which will allow Fog nodes to share computing and storage power to IoT devices that have low computational power within IoT network. The proposed system will be able to schedule the tasks assigned to fog node for easy processing and efficient resource management. The proposed work is focused on creating a resilient environment using SMART FOG which will create trust in fog computing. As fog computing is

in its infancy, there are still many open challenges present. The SMART FOG will create trust between fog clients and fog environment by providing fault-tolerant and secure techniques for fog computing. This research will identify some of these challenges and try to find the solution in proposed system.

Fog computing has attracted by huge number of researchers, so it is a trending topic for research. The literature study motivates research in Fog Computing by introducing a bright future and application of it. The researchers stated that Fog Computing will the how today's IoT and cloud computing are working. The researchers also stated the challenges to be faced in the implementation of Fog Computing in real-life applications. Currently, researchers are working on the implementation of fog for commercial applications. The challenge for further studies and solutions from experts is that we need to keep ourselves updated for online publications and updates from OpenFog consortium related to Fog Computing.

According to author Sheikh (2023), Fog Computing's dynamic nature demands innovative solutions for effective task scheduling. Integrating K-Means clustering with fuzzy logic, addresses Fog's resource constraints, offering adaptability in task allocation. Leveraging machine learning, our methodology optimizes execution time, response time, and network usage by intelligently assigning tasks to Fog nodes.

1.8 Fog Computing Task Scheduling

Fog computing task scheduling refers to the process of efficiently allocating computational tasks to fog nodes in a fog computing environment. It plays a crucial role in optimizing resource utilization, reducing latency, and improving overall system performance. Task scheduling in fog computing involves determining which tasks should be executed, where they should be executed, and when they should be executed. This decision-making process takes into account various factors such as the computational requirements of tasks, the availability and capabilities of fog nodes, network conditions, and user requirements. Efficient fog task scheduling involves several considerations. These include load balancing, where tasks are evenly distributed among fog nodes to avoid overloading or underutilization of resources.

Proximity-aware scheduling considers the geographic proximity of fog nodes to edge devices to minimize communication delays and improve real-time processing capabilities. Furthermore, energy-aware task scheduling focuses on optimizing energy consumption by intelligently allocating tasks to energy-efficient fog nodes or selectively activating certain nodes based on workload requirements.

Deadline-aware scheduling ensures that tasks with time constraints are scheduled promptly, meeting their deadlines. Various scheduling algorithms and techniques are employed in fog computing, such as heuristic-based algorithms, optimization algorithms, and machine learning-based approaches. These algorithms aim to balance the trade-offs between task performance, resource utilization, energy efficiency, and other system objectives.

The study by Aimal (2022) addresses the challenges posed by traditional task scheduling methods in Fog computing for latency-critical applications. By introducing the "Critical Task First Scheduler", which prioritizes tasks based on their nature, particularly focusing on critical tasks with larger MIPs[21] sizes, the proposed methodology aims to reduce latency, energy consumption, and network utilization. Implemented in a healthcare scenario using the Fog simulator, the Critical task First Scheduler, scheduler demonstrates superior performance compared to First Come First Served, Shortest Job First, and cloud-only approaches. Simulation results highlight the efficacy of the Critical task First Scheduler approach in enhancing latency, energy efficiency, and network utilization for critical tasks

[21] Million Instructions Per Seconds

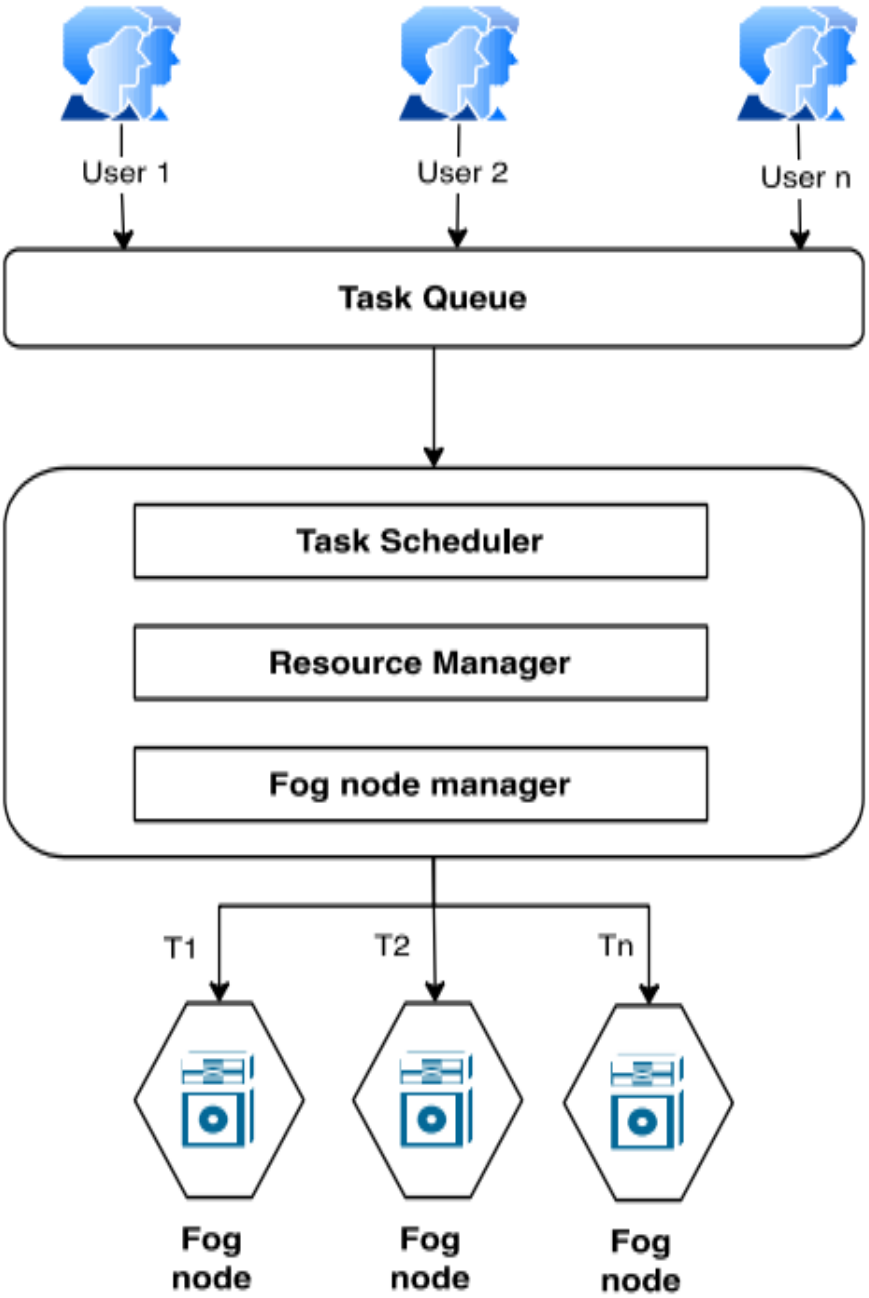

Figure 1.8: Fog Computing Task Scheduling (Alizadeh, 2020)

Figure 1.8 shows the following categories can be used to categorize task scheduling techniques in a fog computing environment are as follows:

- Static task scheduling methods
- Dynamic task scheduling methods
- Hybrid task scheduling methods

Overall, fog computing task scheduling plays a critical role in achieving efficient and effective utilization of fog resources. By carefully managing task allocation and considering various factors, fog computing systems can provide low-latency, energy-efficient, and responsive services to edge devices, enabling a wide range of applications in areas such as IoT, real-time analytics, and edge computing.

1.8.1 Static Scheduling Strategy

The task requirements must be available to the task scheduler before the initial cutting scheduling strategy for static task scheduling approaches to work. Before beginning any scheduling procedure, the task scheduler calculates the needs for each job. In this case, the tasks are sent to the system without regard to the availability of computing resources or the statuses of those resources. The First Come First Serve scheduling approach and the round-robin method are the two most popular task-scheduling techniques in this category. There is different static scheduling strategies as follows:

First Come First Served Method

The First Come First Serve CPU scheduling algorithm processes jobs in the order that they arrive in the ready queue. Newly arrived processes are added to the tail of the FIFO queue. The first process in the queue is scheduled first and removed from the queue.

Max Min Method

Performs a linear transformation on the original data. This technique gets all the scaled data in the range (0,1). The formula to achieve this is that Min-max normalization preserves the relationships among the original data values.

Minimum Completion Time Method

This algorithm locates the task with minimum execution time and allocates the task to the resource on a first come first served basis. Severe load imbalance is the major issue in this algorithm. It does not consider the resource availability and its load.

Opportunistic Load Balancing Method

Is a static load balancing algorithm. OLB keeps all nodes busy, so don't think about previous loads. However, OLB[22] does not consider the execution time of the task on this node.

Round Robin Method

reduce a multi-class problem to multiple two-class problems by learning one classifier for each pair of classes, using only training examples for these two classes, and ignoring all others

Figure 1.9 shows static task scheduling approaches, the task requirements must be known to the task scheduler before initiating any scheduling strategy. The scheduler calculates the resource needs for each job before the scheduling process begins. Consequently, tasks are sent to the system without considering the availability or status of computing resources. Within this category, several task-scheduling techniques are commonly employed.

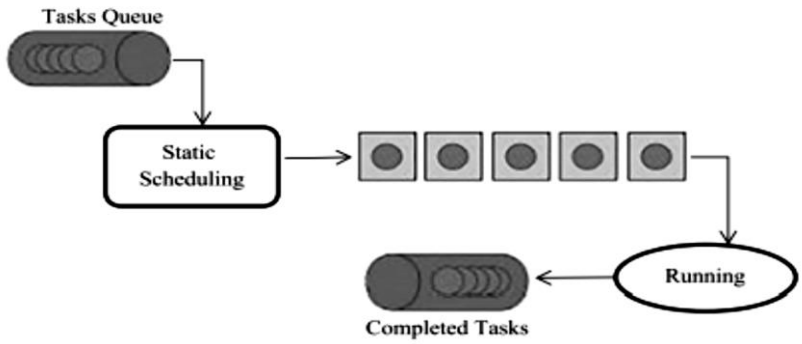

Figure 1.9: Static Scheduling Strategy (Alizadeh, 2020)

One such method is the First Come First Served approach, where tasks are executed in the order they arrive, with no consideration for their resource requirements or priorities. Another technique is the round-robin method, which assigns each task a fixed time slice for execution in a cyclic manner, regardless of their resource needs. Additionally, there are other methods like the Max-Min Method, which allocates resources to tasks based on maximum possible resource utilization, and the Min-Min Method, which focuses on minimizing the completion time of the smallest tasks.

[22] Opportunistic load balancing

Furthermore, the Minimum Completion Time Method prioritizes tasks with the shortest expected completion times, and the Opportunistic Load Balancing Method dynamically allocates resources based on real-time availability and task demands. Each method has its advantages and limitations, and the choice of a specific task scheduling technique depends on the nature of the tasks, the system's resource capabilities, and the desired performance objectives.

1.8.2 Dynamic Task Scheduling Methods

Based on the task's arrival at a specific time and the status of the system machine, dynamic scheduling methods are created. These techniques might take into account a single task at a time or several tasks at once.

Cross Entropy

Cross-entropy, also known as logarithmic loss or log loss, is a popular loss function used in machine learning to measure the performance of a classification model. Namely, it measures the difference between the discovered probability distribution of a classification model and the predicted values. As per-word cross-entropy is the average number of bits required per word, which has the advantage that you can interpret it without knowing. randomly. Perplexity is closely related to per-word cross-entropy; it just undoes the log. One advantage is that you can interpret it without knowing the base of the log.

Genetic Algorithm

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that are based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. In computer science and operations research, a genetic algorithm GA is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms EA. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover, and selection. Some examples of GA applications include optimizing decision trees for better performance, solving sudoku puzzles, hyperparameter optimization, causal inference, etc.

Immune Algorithm

The immune algorithm is a new optimization algorithm imitating the immune system to solve the multimodal function optimization problem. This paper offers a newly modified immune algorithm based on several former immune algorithms and shows its ability to solve the multimodal function optimization problem. A digital immune system is a software development practice for safeguarding applications and services from software bugs and security flaws.

Particle Swarm Optimization

An iterative optimization technique that was inspired by the behavior of social animals such as birds or fish. It involves a group of particles, or agents, that move through a search space and try to find the optimal solution to a given problem. In computational science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution concerning a given measure of quality.

Ant Colony Optimization

In computer science and operations research, the ant colony optimization algorithm is a probabilistic technique for solving computational problems that can be reduced to finding good paths through graphs. Artificial ants stand for multi-agent methods inspired by the behavior of real ants. The pheromone-based communication of biological ants is often the predominant paradigm used. Combinations of artificial ants and local search algorithms have become a method of choice for numerous optimization tasks involving some sort of graph.

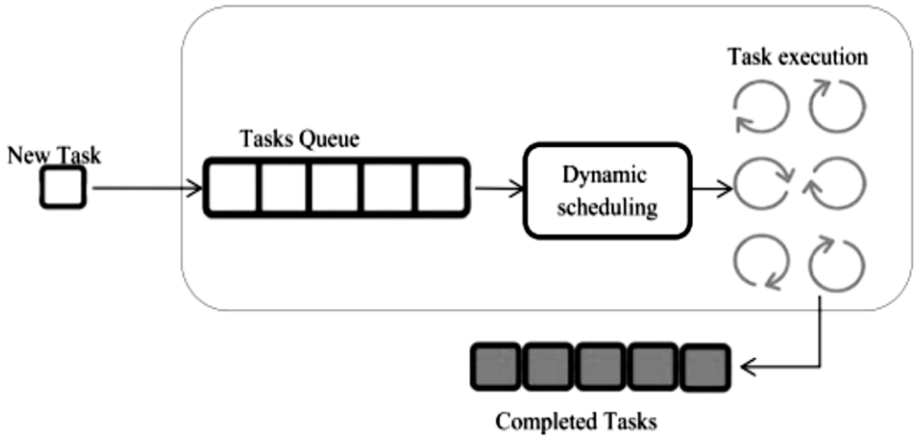

Figure 1.10: Dynamic Task Scheduling (Alizadeh, 2020)

According to Figure 1.10, the dynamic scheduling approach reduces computing costs and long-term service latency. It utilized both a double deep Q learning-based task scheduling method and the reinforcement learning technique. The allocation of user tasks to virtual machines has previously been studied through studies that took into account the propagation, waiting, transmission, and execution delays of various activities. The experimental findings supported the methodology as superior to the existing algorithms.

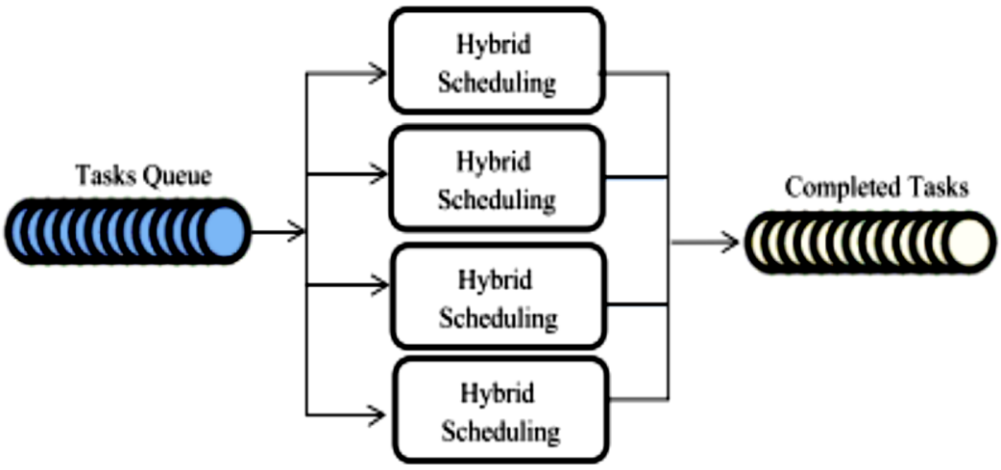

Figure 1.11: Hybrid Task Scheduling Methods (Wang, 2019)

The combination of both Static Scheduling Strategy and Dynamic task scheduling is shown in Figure 1.11. Static scheduling is a strategy where tasks are assigned to resources before program execution and remain fixed during runtime, while dynamic task scheduling involves assigning tasks based on real-time conditions and workload variations. Static scheduling strategies, such as round-robin or block scheduling, provide predictable execution patterns but may not adapt well to dynamic changes. On the other hand, dynamic task scheduling strategies, like work-stealing or task prioritization, dynamically adjust task assignments to optimize performance, considering factors like load balancing and task dependencies. Dynamic scheduling offers flexibility but introduces overhead due to runtime decisions and coordination.

1.8.3 Hybrid Task Scheduling Methods

Hybrid task scheduling methods in fog computing combine multiple approaches or techniques to optimize task allocation and resource utilization. These methods leverage the strengths of different scheduling strategies to address the unique challenges and requirements of fog computing environments.

One common approach is to combine centralized and decentralized scheduling techniques. In centralized scheduling, a central controller or orchestrator is responsible for making task allocation decisions based on a global view of the system. Decentralized scheduling, on the other hand, distributes the task allocation decision-making process among fog nodes themselves. Hybrid methods may use a combination of both approaches, with the central controller handling high-level task allocation decisions and individual fog nodes making local decisions based on their local knowledge and resources.

Another hybrid approach is to combine static and dynamic scheduling. Static scheduling involves pre-determining task assignments based on static parameters such as task characteristics and node capabilities. Dynamic scheduling, on the other hand, adjusts task assignments in real-time based on changing system conditions and workload demands. Hybrid methods can utilize static scheduling for long-term task allocation planning while incorporating dynamic scheduling to adapt to dynamic changes in the system.

Furthermore, hybrid methods may integrate heuristic algorithms with optimization techniques. Heuristic algorithms provide fast and approximate solutions by utilizing predefined rules or guidelines. Optimization techniques, such as genetic algorithms or particle swarm optimization, aim to find optimal solutions by exploring the search space. Hybrid methods leverage the speed and simplicity of heuristics for initial task allocation and use optimization techniques to refine and improve the initial solutions. Hybrid task scheduling methods in fog computing are designed to strike a balance between efficiency, scalability, adaptability, and system performance. By combining different scheduling approaches, these methods can effectively handle the complexity and variability of fog computing environments, leading to optimized task allocation, reduced latency, improved resource utilization, and enhanced overall system performance.

1.9 Fog Computing Challenges

Fog computing faces several challenges that need to be addressed for its successful implementation and operation. Overcoming the challenges requires collaborative efforts from researchers, industry stakeholders, and standardization bodies to innovate and develop solutions that maximize resource utilization, enhance security, promote interoperability, scale the system, reduce energy consumption, and optimize task allocation. By addressing these challenges, fog computing can realize its potential in enabling efficient and reliable edge computing solutions for various applications. These challenges include:

1.9.1 Drones

Drones can be used in ITS[23] applications not just as dumb sensors but also as smart fog nodes, with external devices like the Raspberry Pi, Intel Edison, and ROCK64 placed on the top of the drone to aid in traffic monitoring by seeing and locating errant cars. Similar to this, a drone can serve as a flying policeman in tele-surveillance applications, able to identify and apprehend criminals. Therefore, further research must be done to determine how drones can be used in a fog computing architecture.

1.9.2 Machine learning

Applications like ITS, healthcare, and tele-surveillance require real-time data processing and speedy replies, which might be given by implementing machine learning in fog nodes. To make judgments based on the information gathered from the sensors, the fog nodes must be intelligent enough. We are proposing a more robust approach that integrates drones with machine learning and extends it to capture the misbehaving cars and the driver's face identification. An earlier study recommended utilizing machine learning in fog nodes to anticipate busy locations.

1.9.3 Security and Privacy

The sensors' limited resources prevent a large computation cryptography approach from being used. The current priority is to secure the system and duty of the fog nodes to prevent the spread of clouds with harmful packets. Using the Diffie-Hellman problem for cryptography regarding the use of hash collision cryptography

[23]Intelligent Transportation System

for traffic security ITS applications that use a light system and fog devices signal light. Additionally, fog nodes must confirm the queries made by IoT devices when the devices themselves must confirm the security of the fog node.

1.9.4 Autonomic Fog Management and Connectivity

To meet the real-time processing needs of ITS, tele-surveillance, and healthcare applications, fog devices must be able to control themselves independently. Additionally, it poses a problem to maintain a smooth connection between all deployed devices in the fog computing architecture because of the expectation that they would be diverse.

Fog computing faces several challenges that require collaborative efforts to address and overcome. These challenges include the integration of drones as smart fog nodes for applications such as traffic monitoring and tele-surveillance, the implementation of machine learning in fog nodes for real-time data processing, ensuring security and privacy despite limited sensor resources, and achieving autonomic fog management and connectivity for efficient and smooth operations. Overcoming these challenges is crucial for realizing the potential of fog computing in enabling efficient edge computing solutions for various domains. Further research, innovation, and standardization efforts are necessary to tackle these challenges and unlock the full capabilities of fog computing.

1.10 Machine Learning Algorithms

To avoid imprecise or erroneous predictions, the data collected / generated must go through pre-processing, merging, modifying, and learning. the computational intensity and speed of a specific technique are two significant characteristics to consider while employing ML. techniques. The best algorithm is chosen based on the user application and should be fast enough to track changes in the input data and provide the desired output in a reasonable amount of time. ML algorithms create a mathematical model using sample data, known as "training data," on which to make predictions or choices. The training phase of supervised ML classifier development involves training a specific classifier from a set of labeled data. As the size of the training data increases, so do the classifiers. Some of the most popular ML algorithms are detailed further below.

1.10.1 Naive Bayes

Based on Bayes' theorem, a naive Bayes classifier is a probabilistic classifier that works by assuming that no pair of features are dependent. Naive Bayes is a simple but powerful machine learning algorithm based on Bayes' theorem and the assumption of independence between features. Despite its simplicity, Naive Bayes is often effective and computationally efficient, so it is often used in a variety of classification tasks. It is particularly suitable for text classification and spam filtering.

1.10.2 Logistic Regression

Logistic regression is a machine learning algorithm commonly used for binary classification tasks, where the goal is to predict whether an instance belongs to one of two classes. Despite its name, logistic regression is more of a classification algorithm than a regression algorithm. Logistic regression is a fundamental machine learning algorithm that is widely used in various applications such as medical diagnostics, spam detection, and credit scoring due to its simplicity, interpretability, and effectiveness. Although it is designed for binary classification, it can be extended to handle multiple classes through techniques such as one-vs-rest regression and softmax regression.

1.10.3 Sequential Minimal Optimization

SMO is a machine learning algorithm designed to train SVMs in supervised learning. SVM is used for classification and regression tasks, and SMO is a specific algorithm used to efficiently solve the optimization problems associated with training these models. Although SMO is an important algorithm for SVM training, there are alternative approaches and optimizations to solve SVM problems, such as the widely used libsvm library that implements more general optimization techniques. Still, understanding SMO provides insight into the support vector machine training process.

1.10.4 Instance-Based Learner

IBk is a machine learning algorithm used for classification and regression tasks. It is part of the family of k-NN[1] algorithms, where the prediction of a new instance is based on the majority class for classification or mean for regression of the k-nearest neighbors in a function space. The main feature of the IBk algorithm is Instance-

based learning. This means that no explicit model is created during training. Instead, save the training instance and use it to make predictions for new instances. In K-NN Predictions for new instances are determined by examining the class labels for classification or values for regression of the k-nearest neighbors in the training data set. Small values of k give the model that is more flexible and sensitive to noise, and large values of k gives the model that is smoother and less sensitive. Regression uses the average of the k nearest neighbor target values as the prediction. IBk can be computationally expensive, especially for large datasets, as it must calculate the distance for each prediction. It is often more efficient when the dataset is small. IBk performance can be sensitive to feature scaling. Therefore, it is often recommended to normalize or standardize features to obtain a similar scale. IBk is a simple but effective algorithm, especially in situations where the decision boundary is complex and not easily captured by parametric models. It is widely used in various fields such as pattern recognition, classification, and regression.

1.10.5 K-Star

K Star was developed in 2009. K Star was originally implemented as part of DiPro toolset for generating counterexamples in probabilistic model checking. K. Star A directed search algorithm also called K. It Finds the k shortest paths between the given pair of vertices in the given directed weighted graph. K Star works on the fly. This means that the graph does not have to be made explicitly available and stored in main memory. K Star can be also be controlled using a heuristic function.

1.10.6 Multi Class Classifier

A multiclass classifier is a type of machine-learning algorithm that can assign instances to one of three or more classes. Unlike binary classifiers, which distinguish between two classes such as positive or negative, multiclass classifiers handle scenarios where there are multiple possible classes. Some of the common Multi Class algorithms are Support vector machine, Random Forest, K Nearest Neighbours, Neural Networks and Decision Trees. The choice of algorithm often depends on factors such as the size and type of the dataset, computational efficiency, and the desired interpretability of the model.

1.10.7 Random Forest

A decision tree-based supervised machine learning approach called RF depends on values from a random vector that is sampled separately and with the same distribution across all of the trees in a forest. By averaging the results, this ensemble method lowers over-fitting and bias-related error, leading to superior outcomes. Random Forest is a powerful and versatile machine learning algorithm that belongs to the ensemble learning category.

Ensemble learning combines the predictions of multiple models to create a more robust and accurate model. Random forests are particularly effective for both classification and regression tasks. The main features and characteristics of the Random Forest algorithm are: Ensemble of Decision Trees: Random Forest is an ensemble of Decision Trees. A decision tree is a discrete model that makes predictions based on a series of hierarchical decisions. Random forests create multiple decision trees and combine their predictions during the training phase. During the training process. Random Forest randomly selects a subset of the training data (with permutations) to train each decision tree. This process is called bootstrapping. Additionally, at each decision point in the tree, a random subset of features is also considered. Random Forest uses a technique called bagging, where each decision tree is trained independently on a different subset of the data. The final prediction is determined by aggregating the predictions of all trees. By training multiple decision trees on different subsets of data and features, random forests become more robust and less prone to overfitting compared to a single decision tree. Overfitting occurs when a model learns the training data well enough but is unable to generalize to new, unseen data. Random Forest provides a measure of feature importance.

Analysing the contribution of each feature across multiple trees can help determine which features have the greatest impact on predictions. The training of individual decision trees in a random forest can be performed in parallel, resulting in a scalable algorithm that can efficiently process large amounts of data. Random forests tend to be less sensitive to outliers in a dataset. Because each tree is trained on a subset of the data, the impact of outliers is reduced. Random Forest has been implemented in various machine learning libraries such as Scikit-Learn in Python, making it highly accessible and widely used.

1.10.8 Random Tree

Random Tree is a term often associated with two different machine learning algorithms, Random Forest and Highly Randomized Trees Extra Trees. Both algorithms fall into the category of ensemble learning and are used for classification and regression tasks. Both Random Forest and Extra Trees are powerful algorithms that leverage the concept of ensemble learning to improve predictive performance. They are widely used in various applications such as classification, regression, and feature importance analysis. The choice between random forests and extra trees may depend on the specific properties of your data and the desired trade-offs between computational efficiency and model accuracy.

1.10.9 Multi-Layer Perceptron

It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network, we combine neurons so that the outputs of some neurons are inputs of other neurons. A multi-layer perceptron has one input layer and for each input, there is one neuron (or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes.

1.10.10 k-Nearest Neighbors

The k-nearest neighbor algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today. While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

1.10.11 Supervised

Giving training data that has previously been "known" or "labeled" with the proper response and consists of N input-output pairs (X,Y) is how supervised learning functions. The ANN then generates an output 2 for each unknown X, which is then compared against Y using an error cost or distance function. Finally, an iterative process is used to minimize this mistake. Image Classification: Training with

image/label datasets are examples of supervised learning methods. A new image is then presented later with the hope that the computer will pick up on the new object. Regression: Giving the system marked historical data so it can forecast the future result of an identical circumstance.

1.10.12 Unsupervised

Using unsupervised learning methods, it self-organizes and finds hidden patterns in unlabeled input data to create neural networks. It can analyse data without sending an error signal so that the potential fix can be assessed. Unsupervised learning can occasionally be useful since it allows the algorithm to search the past for patterns that weren't previously taken into account. Unsupervised learning is necessary because manually inspecting huge datasets like those for speech recognition is highly expensive. Clustering is a very basic but well-known example of unsupervised learning.

1.10.13 Semi-Supervised

This category is a hybrid of the previous two. The algorithm is trained on a dataset that contains both labeled and unlabeled data. It works by taking enormous amounts of input data and labeling only a subset of it as training data. Reinforcement learning, a related strategy, provides feedback to guide the computer program in interacting with a dynamic environment. In this approach, a model is deployed using a small set of labeled samples and a larger set of unlabeled samples. The goal is to use labeled data to make predictions about unlabeled data and use the additional information to improve model performance.

1.11 Fog Computing Real-Time Applications

Fog computing offers significant advantages in real-time applications. It is often utilized in IoT applications that need real-time data. It functions as a more advanced kind of cloud computing. It serves as a conduit between end users and the cloud. It may be utilized in both scenarios—between humans and machines or between machines and machines.

1.11.1 Mobile Big Data Analytics

Data acquired by IoT devices is gathered in large quantities, making cloud storage ineffective. Fog computing, which uses nodes that are considerably closer to end

systems than cloud computing, is advantageous in such circumstances. It also gets rid of additional issues like delays, traffic, processing speed, delivery time, response time, data processing, data storage, and data transportation. IoT applications of the future may use fog computing.

1.11.2 Dams Safety

Dam sensors transmit data to the cloud, where it is examined and if there are any anomalies then officials are notified the issue here is the potentially deadly information delay. Fog is utilized to address this, and because it is located close to the end systems, it is simpler to send data, evaluate it, and provide immediate response. In dam monitoring scenarios, sensors play a vital role in collecting data related to dam conditions, such as water level, pressure, temperature, and structural integrity. Traditionally, this data was transmitted directly to the cloud for analysis and decision-making. To address this challenge, fog-based architecture, also known as edge computing, is employed. Fog nodes, placed near the dam sensors, act as local processing hubs. These fog nodes receive the data from the sensors and perform real-time analysis and anomaly detection locally. By doing so, they significantly reduce the data transmission time to the cloud and enable swift evaluation of dam conditions.

1.11.3 Smart Utility Service

Here, saving time, money, and energy is the major goal. Data analysis must be conducted every minute on current data. Since end users are primarily involved, cloud computing may not be useful. These programmers daily notify users of which appliances utilize the least amount of energy. Fog is an excellent option since IoT generates a lot of network traffic that makes it difficult to transfer other data.

1.11.4 Health Data

When information needs to be shared between hospitals, strict security, and data integrity are requirements. Fog may be used to achieve this because the data is conveyed locally. The laboratories may utilize these fog nodes to update the patient's lab information, which the adjacent hospitals can simply access. Since any clinician may access this unified information, patients do not need to carry hard copies of their medical histories or health concerns.

1.11.5 Smart Cities

The idea of a "smart city" has generated a great deal of attention in recent years because it promises to improve the quality of life. An urban setting known as a "Smart City" is one in which several sectors work together to produce sustainable outcomes by analyzing real-time data. Building smart cities presents the problem of assuring accuracy and speed in reaction times when assessing the condition of infrastructure components like gas and oil pipelines, subways, and roadways. Additionally, the enormous amount of data the sensors produce creates problems with big data processing.

1.11.6 Tele-Surveillance

The concept of placing fog nodes next to CCTV[24] cameras at shopping malls and railway stations to get data from them to identify hazards like trespassing in security zones and gunshots. A video content management system is employed in the fog nodes to process and store the footage for the threat detection process.

Fog computing offers numerous benefits for real-time applications, particularly in the context of the IoT. It serves as an advanced form of cloud computing, acting as a bridge between end users and the cloud. Fog computing finds relevance in various scenarios, including human-machine and machine-machine interactions. Some notable real-time applications of fog computing include mobile big data analytics, ensuring dam safety through immediate data analysis and response, smart utility services for efficient energy consumption, secure health data exchange between hospitals, the development of smart cities for sustainable outcomes, and tele-surveillance systems for threat detection. Fog computing provides advantages such as reduced delays, improved processing speed, enhanced data storage and transportation, and localized data communication, making it a valuable solution in these real-time scenarios.

[24]Closed-Circuit Television

Chapter-2
Literature Review

2.1 IoT Overview

2.2 Fog and Edge Computing

2.3 IoT and Fog Computing Applications

2.4 Fog Computing and Smart Cities

2.5 Resource Allocation and Task Scheduling Technique

A literature review is a critical and comprehensive summary and analysis of existing scholarly research and publications relevant to a specific topic or research question. Its primary purpose is to provide an overview of the existing knowledge, identify research gaps, and establish the context and significance of the new study within the academic discourse. By identifying relevant studies, analyzing key findings, and synthesizing information from various sources, researchers can develop a well-supported theoretical framework and justify their research objectives. A well-conducted literature review showcases the researcher's ability to critically evaluate and integrate existing knowledge, laying the foundation for a new study and contributing to the advancement of the field. The chapter includes the previous studies related to Fog, IoT[25], Cloud computing, and machine learning algorithms for developing task offloading and resource allocation models.

2.1 IoT Overview

According to IEEE[26] Communication Magazine, the IoT is a framework that gives every object a digital representation and online presence. More precisely, the IoT intends to provide brand-new services and applications that connect the real and virtual worlds. M2M[27] communications serve as the foundational communication for interactions between Things and cloud-based applications. Oxford Dictionaries provides a summary definition that calls the Internet an element of "IoT the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data".

Aalsadie (2022) discussed that billions of physical objects have been connected thanks to the development of IoT cloud computing to share and gather data for various uses. Despite significant developments, some latency-specific applications are still impractical in the real world because of the limitations of current IoT devices and the distance between the cloud and IoT devices. Fog computing, which makes use of the availability of computing and storage resources at the edge of the network close to the IoT devices, has been created to address the difficulties of

[25] Internet of Things
[26]Institute of Electrical and Electronics Engineers
[27]Machine-to-Machine

latency-sensitive applications. Fog computing does, however, have several drawbacks, including heterogeneity, storage, processing, and memory constraints. As a result, it necessitates a suitable job scheduling technique for making the most use of computing resources at the Fog layer. This article offers a thorough analysis of several job scheduling techniques used in fog computing. It examines and contrasts several task scheduling techniques created for a Fog computing environment to highlight their benefits and drawbacks.

Abohamama (2022) says that applications for the IoT are now essential for raising living standards. However, the resources of conventional cloud data centers are under strain due to the growing volume of data produced by IoT devices. Because of this, cloud data centers are unable to meet the needs of IoT applications, especially those that demand fast response times. A more contemporary computing model called Fog computing brings cloud resources out to the network's edge. Task scheduling is still difficult under this computer paradigm, though. For bag-of-tasks applications in the cloud-fog environment, a semi-dynamic real-time task scheduling technique is presented in this paper. Task scheduling is formulated as an optimization issue using permutations in the suggested scheduling technique. For each scheduling cycle, a modified version of the genetic algorithm is employed to give several permutations for jobs that have arrived. The jobs are then assigned to a virtual machine with enough resources to meet the lowest expected execution time, in the order determined by the best permutation. According to an optimality analysis that was done, the suggested algorithm performs comparably to the best option. In terms of make pan, total execution time, failure rate, average delay time, and elapsed run time, the suggested method is also contrasted with the first fit, best fit, the genetic algorithm, and the bee's life algorithm.

Atzori, Iera, and Morabito's (2010) comprehensively delve into the intricate landscape of the IoT. The authors meticulously dissect the key components of IoT, spanning architectures, communication protocols, and diverse application domains. Their thorough analysis not only identifies challenges but also sheds light on the myriad opportunities within the IoT realm. This work stands as an invaluable resource, catering to researchers, practitioners, and policymakers alike, providing a holistic understanding of the evolution and impact of IoT.

Bandyopadhyay and Sen (2011) focus on the applications of IoT, accentuating technological challenges and standardization issues. This work is pivotal in bridging the theoretical concepts of IoT with pragmatic considerations, serving as a practical guide for industry professionals navigating the intricate landscape of IoT implementation and standardization.

Berman, Cabrera, Jebari, and Marrakchi's (2022) contribution to Patterns introduces the innovative "impact universe" framework. This framework serves as a compass for assessing and prioritizing public interests in IoT deployments, addressing ethical considerations and societal impacts. The work significantly contributes to the ongoing discourse on responsible IoT development, offering crucial insights for policymakers and industry leaders striving to align IoT innovations with societal needs.

Borodin's (2014) article in Educational Resources and Technologies anticipates the transformative potential of IoT in education. By envisioning the integration of IoT into educational settings, the author foresees a paradigm shift in how information is accessed and disseminated, laying the groundwork for discussions on leveraging IoT for educational advancements.

Bubnova and Kryukova's (2014) work in Economics and Society explores the intersection of IoT and customer-centric strategies in modern business practices. Focusing on social client-oriented technologies, the authors illuminate the evolving dynamics of customer engagement. This article serves as a valuable resource for businesses seeking to leverage IoT to enhance customer experiences and adapt to the dynamic landscape of market trends.

The China Internet Watch Team's report (2023) forecasts China's substantial investment in the IoT, projecting spending to reach an impressive US$298 billion by 2026. The comprehensive analysis highlights key IoT market trends in China, offering valuable insights into the nation's technological landscape and its strategic positioning in the global IoT arena. The report serves as a vital resource for industry stakeholders, businesses, and researchers seeking to understand and navigate China's dynamic IoT market.

Chiang (2016) surveyed various articles on fog and IoT. The authors overviewed the research opportunities of Fog Computing. They provide summarized information about the opportunities and challenges of Fog, over the networking context of IoT. The fundamental challenges discussed are like stringent latency requirements, network bandwidth constraints, resource-constrained devices, cyber-physical systems, uninterrupted services to the cloud, and security threats etc.

DeMedeiros, Hendawi, and Alvarez's (2023) survey, published in Sensors, focuses on AI-based anomaly detection in IoT and sensor networks. This research critically examines the integration of artificial intelligence into anomaly detection systems, providing a thorough understanding of current trends and advancements. The survey is a valuable reference for researchers and practitioners involved in IoT security, offering insights into the evolving landscape of anomaly detection.

Duarte's (2023) exploration of the number of IoT devices becomes particularly relevant in understanding the proliferation of connected devices. As the IoT continues to expand, Duarte's work, available on Exploding Topics, offers valuable data and insights into the sheer scale and growth of IoT devices. This information is crucial for stakeholders, businesses, and policymakers shaping the future trajectory of IoT.

Dubravac and Ratti's (2015) form part of the IoT report series by American International Group. The document critically analyzes the trajectory of IoT, exploring whether its development represents an evolutionary process or a revolutionary shift. This whitepaper is a foundational resource for professionals, policymakers, and academics seeking a nuanced understanding of IoT's historical development and future implications.

Guinard, Trifa, Karnouskos, Spiessand Savio's (2010) delve into interacting with the Service-Oriented Architecture-based IoT. The study emphasizes aspects like discovery, query, selection, and on-demand provisioning of web services in the context of IoT. This work is essential for researchers and practitioners involved in IoT architecture and service provisioning, providing insights into the fundamental principles shaping IoT interactions.

Guzuyeva's (2018) contribution in the proceedings of the IV International Correspondence Scientific and Practical Conference focuses on the application of information technology in large and small businesses. The research explores the role of IT in different business scales, offering practical insights into the varied applications of technology in contemporary business environments. This work serves as a valuable reference for academics, business professionals, and policymakers interested in the intersection of information technology and business operations.

Hakan's (2023) focus on bibliometric analysis and scientific mapping of IoT, featured in the Journal of Computer Information Systems, contributes to the scholarly understanding of IoT research trends. The work employs bibliometric techniques to map the landscape of IoT research, offering valuable insights into the growth, trends, and focal points of the field. This research is particularly relevant for academics, researchers, and institutions seeking to comprehend the current state and future directions of IoT research through a quantitative lens.

Katal (2022) discussed the development of new methods to make cloud operations more accessible has increased with the proliferation of IoT devices. The IoT has grown over the past ten years as a result of ongoing developments in hardware, software, and communication technologies, with a daily increase in the number of linked items. The creation of an adequate system architecture capable of processing and storing all of the data is required due to the enormous volume of data generated by these devices. The separate idea of "Fog computing" and the integrated fog-to-cloud computing paradigm is particularly important in this regard for decentralizing the cloud and bringing services closer to the finished system. Fog computing's main objective is to improve common IoT situations by minimizing delays and saving traffic by bringing awareness to the entrance point. Fog computing applications include real-time requirements, wireless networks, and low-power devices. An appropriate Fog computing protocol should be portable, adaptable, and lightweight in light of these factors.

Tzavaras, Mainas, and Petrakis (2023) presented an innovative OpenAPI framework designed for the Web of Things. Published in the IoT journal, the paper introduces a framework that leverages OpenAPI specifications to enhance the interoperability of

Web of Things applications. The authors provide insights into the development and implementation of this framework, contributing to the ongoing discourse on standardization and interoperability in the IoT domain.

Uckelmann, Harrison, and Michahelles (2011) propose an architectural approach for the future IoT. This seminal work, presented in the book "Architecting the IoT," lays out a comprehensive framework for understanding the architecture of IoT systems. The authors explore crucial aspects such as communication protocols, data models, and system components, providing a foundational resource for researchers, practitioners, and educators involved in IoT architecture.

Van Kranenburg (2014) delivers an open lecture on the IoT, contributing valuable insights to the discourse on IoT concepts and applications. This lecture, conducted at MEPhI, covers fundamental aspects of IoT, fostering a deeper understanding of the technology's implications in various domains.

Wagan (2022) conducted a comprehensive review focusing on the Internet of IoMT[28] and converging technologies with real-time applications. Published in the Journal of King Saud University – Computer and Information Sciences, the paper provides an in-depth exploration of IoMT, offering valuable insights into its applications and the integration of trending technologies in the healthcare domain.

Wang (2013) presented a system framework for security management in enterprise systems. Published in Systems and Behavioral Research Science, the research work addresses critical aspects of security management, offering a systematic approach to enhance the security posture of enterprise systems. The authors contribute valuable knowledge to the field of enterprise security.

Wattics (2011) provides insights into smart metering, emphasizing its role in energy management. As a key player in the energy management sector, Wattis offers information on smart metering solutions, contributing to the broader understanding of technologies aimed at optimizing energy consumption.

[28]Internet of Medical Things

Withers (2023) explores how organizations in the AP[29] region can effectively harness the power of the IoT. The article, featured in Computer Weekly, provides strategic insights and recommendations for APAC businesses aiming to leverage IoT technologies for enhanced operations and growth.

Wu, Sheng, and Zeadally (2013) delve into RFID, examining its opportunities and challenges. Published in the book "Next-generation Wireless Technologies," the chapter provides a comprehensive overview of RFID technology, shedding light on its applications, benefits, and challenges that need to be addressed for widespread adoption.

Xing, Li, Wilamowska-Korsak, and Zhang (2013) presented a comprehensive review of Operations Research in service industries. Published in Systems and Behavioral Research Science, the paper explores the applications of operation research methodologies in optimizing service processes across various industries. The authors contribute valuable insights into the role of operation research in enhancing operational efficiency in service-oriented sectors.

Xu (2011) provides a comprehensive exploration of enterprise systems, examining their state-of-the-art features and future trends. Published in the IEEE Transactions on Industrial Informatics, the paper contributes to the understanding of evolving enterprise technologies, making it a valuable resource for researchers and professionals in the field.

Ystgaard (2023) conducted a comprehensive review focused on the theory, principles, and design requirements of the human-centric IoT. Published in the Journal of Ambient Intelligence and Humanized Computing, the paper emphasizes the importance of user-centric design in IoT systems, contributing to the development of more inclusive and user-friendly IoT technologies.

Yudidharma (2023) conducted a systematic literature review, focusing on messaging protocols and electronic platforms used in the IoT for building smart homes. Published in Procedia Computer Science, the review provides a comprehensive overview of the state-of-the-art IoT protocols and platforms for smart home

[29]Asia-Pacific

applications. The authors contribute valuable insights into the technological landscape shaping smart home development.

This section describes in detail the different communication protocols used in fog computing and makes comparisons between them based on important criteria. The explanation of the research difficulties for the communication protocols of fog computing serves as its conclusion.

2.2 Fog and Edge Computing

Fog and edge computing, helps devices to get faster results by processing the data simultaneously received from the devices. Fog computing helps in filtering important information from the massive amount of data collected from the device and saves it in the cloud by sending the filtered data.

Ahmad (2021) discussed that the number of people utilizing the Internet has increased with the spread of smart gadgets worldwide. The primary goal of the fog computing paradigm is to connect a vast array of intelligent objects billions of objects to create a bright future for smart cities. Due to the widespread use of smart gadgets, it is anticipated that these devices will produce enormous volumes of data and transmit that data through the Internet. Fog Computing also refers to an edge-computing architecture that lessens the problem by implementing knowledge discovery at the edges utilizing a data analysis technique. To create a sustainable infrastructure for smart cities, the IoT and Fog Computing techniques can cooperate. The weighted round-robin scheduling technique is the one the author suggests be used to execute the job from one fog node to another fog node and finally to the cloud. First, IoT infrastructure for smart cities is designed using a Fog simulator and the emergent Fog Computing idea. Then, data gathering and routing are done using the spanning-tree routing protocol. The establishment of quick transmission and user communication via 5^{th} Generation networks is also envisaged. The effectiveness of our suggested method is then assessed in terms of reaction time, latency, and data use.

Bittencourt, Lopes, Petri, and Rana (2015) presented research on P2P, Parallel, Grid, Cloud, and Internet Computing, this work is likely to explore virtual machine migration in Fog computing. The authors may discuss the challenges and potential

benefits associated with migrating virtual machines in Fog computing environments, contributing to the understanding of resource management in distributed systems.

Bose, Aujla, Singh, Kumar, and Cao (2019) investigated the application of blockchain as a service for software-defined networks, focusing on potential denial-of-service attacks. The authors may discuss how blockchain technology can be leveraged to enhance the security of software-defined networks and address challenges related to denial-of-service attacks.

Dsouza, Ahn, and Taguinod (2014) introduce a policy-driven security management framework for fog computing in their 2014 paper presented at the IEEE International Conference on Information Reuse and Integration. The work may propose a preliminary framework and provide a case study, contributing valuable insights to the development of security management strategies in fog computing environments.

Huang, Yang, and Wang (2017) work is likely centered around secure data access control in fog computing for the IoT. The authors may discuss innovative techniques such as ciphertext update and computation outsourcing to ensure secure data access, particularly crucial in the context of IoT where data privacy is paramount.

Kumar (2022) says that network's edge, Fog computing provides an integrated key to enable communications, data collection, device management, services capabilities, storage, and analysis. This makes it possible to install infrastructure that is centrally controlled in a highly dispersed setting. The most important uses of Fog computing for smart city infrastructure are covered in the current study. The most crucial issue arises when operating a large number of IoT-based services in a smart city context. To deliver novel services, thousands of smart things, cars, phones, and people connect; in this situation, the fog computing infrastructure may be very helpful from a data and communication standpoint. Three primary topics are the subject of this section

a) The deployment of data and applications in fog nodes
b) 5^{th} Generation connectivity leveraging the fog infrastructure
c) Fog-based data management and analytics

These fog computing applications are illustrated with working models from all angles. The effective integration of smart city infrastructure has new use cases. An increasing interest in smart cities has also been taken into account while presenting the difficulties and prospects.

Martin (2017) discussed the OpenFog security requirements and its approaches. This paper provided a security overview of OpenFog architecture and also provided a survey on the functional requirements and the technical approaches. This paper aims to simulate further dialogue on security in OpenFog and fostering future development of novel technologies and practices.

Mebrek (2017) introduces a solution for the increasing demand for IoT devices through the study of the Fog Computing suitability assessment. The authors focus on energy consumption and the Quality of Service as two important aspects of the performance of Fog computing. Therefore, they present the modeling of these two metrics in the fog. The authors expressed the problem as constrained optimization and solved it efficiently using Evolutionary Algorithms. The authors stated that their approach stands out as an energy-efficient solution.

Mohamed (2017) says that Unmanned Aerial vehicle-based Fog Computing IoT aims to utilize the advantages and features of both technologies to effectively support IoT applications. This proposed Unmanned Aerial Vehicle Fog provided fast deployment of Fog capabilities at remote or challenging locations to effectively support dynamic IoT applications. The authors proposed that Unmanned Aerial vehicles equipped with Fog computing capabilities can be used to travel to a specific location when needed and remain in that location to support their IoT applications. The authors discussed some scenarios for such deployments, their advantages, and the issues involved when using the proposed model.

Naha (2018) surveyed trends, architecture, requirements, and research directions. This survey paper is useful to industries and research communities to synthesize and identify the requirements for Fog Computing. The authors overviewed Fog definition, research trends, and the technical difference between Fog and cloud. The authors investigated numerous proposed Fog architectures and discussed the components of these architectures. The authors discussed the taxonomy of Fog

computing by considering the requirements, as well as authors discussed the research gap in resource allocation and scheduling, fault tolerance, simulation tools, and Fog-based microservices. Also addressed the limitations of current research work, and presented open issues and future direction for Fog computing.

Pek, Buttyán, and Bencsáth (2013) survey likely provides an extensive overview of security issues in hardware virtualization. The authors may cover various security challenges associated with hardware virtualization, offering valuable insights into securing virtualized environments.

Pooranian (2017) proposed a distributed Fog-based networked architecture that preserves energy in Fog data centers. The authors present a new Internet of Everything architecture for Fog centers to implement the resulting Fog of Everything technology platform. The authors also present the energy-aware algorithm adopt Fog data centers and obtain numerical performance, for a real-world case study that shows that their approach saved energy consumption in Fog data centers.

Ren, Zhu, Qi, Wang, and Sangaiah, A.K. (2019) explore identity management and access control based on blockchain in edge computing for the industrial IoT. The work may discuss how blockchain technology enhances security measures, ensuring robust identity management and access control in industrial IoT environments.

Tao et al. (2019) published a survey of virtual machine management in edge computing. The authors may provide a comprehensive overview of challenges, techniques, and future directions in the management of virtual machines within the context of edge computing.

Sabireen (2021) discussed that performance, security, latency, and network failure are just a few of the problems that integrated cloud computing must contend with as IoT applications continue to grow. These problems are solved by bringing cloud computing closer to the IoT thanks to the development of fog computing IoT. The main purpose of the fog is to deliver the data produced by the edge IoT devices. Instead of sending the data to a cloud server, local processing, and data storage are carried out at the fog node. When compared to the cloud, fog computing offers high-quality, quick-response services. Fog computing may thus be the best choice for

enabling the IoT to provide a reliable and highly secure service to a large number of IoT customers. It enables management of services and resource provisioning outside of central control, closer to devices, at the network edge, or eventually at locations designated by SLAs[30]. Fog computing is a prevalent element, not a substitute for cloud computing While providing the option to connect with the cloud's data center, it permits information processing at the edge. The author presents different computing paradigms, fog computing features, an extensive reference architecture of fog with its various levels, a thorough analysis of fog with IoT, different fog system algorithms, and also systematically examines the challenges in fog computing, which serves as a middle layer between IoT sensors or devices and cloud data centers.

Satyakam (2021) says that Fog Computing is sometimes referred to as edge computing, which broadens the cloud computing concept while increasing productivity and reducing latency. With the help of cloud computing, Fog computing is employed to keep up with the steadily expanding demand for IT[31]-related services. With the rapid advancement of IT, fog computing is emerging as a desirable method for retrieving and transforming data connected to IoT applications. This article examines the idea, the architecture, and the use of Fog computing in both current and future applications. Fog technology is quite sophisticated and diverse. One of the main difficulties in running the research work objective is to review recent research on resource distribution in the fog region. The purpose of this survey is to understand the use of Fog computing and make some changes to current technology.

Savya (2021) says that for smart manufacturing, Fog computing offers processing, storage, and network services. The task requests, terminal equipment, and fog nodes, on the other hand, are very heterogeneous in a smart factory. For example, different task characteristics of terminal equipment include high real-time demands for fault detection tasks, high calculation requirements for production scheduling tasks, high storage requirements for inventory management tasks, and so on. The Fog nodes also include a variety of processing capabilities, making powerful fog nodes with

[30]Service Level Agreements
[31]Information Technology

large computational resources competent to assist terminal equipment in processing difficult tasks like factory inspection, defect detection, status analysis of devices, and so forth. With the dispersed architecture that Fog computing offers at the network's edges, access is low-latency, and application requests are handled more quickly. With this increased computational power, new techniques for managing and allocating resources may be created to benefit the Fog infrastructure.

Sood and Mahajan (2017) presented a Wearable IoT sensor-based healthcare system designed for identifying and controlling the Chikungunya virus. The work likely explores the integration of wearable IoT sensors into healthcare, aiming to enhance real-time monitoring and management of viral outbreaks. The authors may discuss the technical aspects of the system, potential applications, and the impact on public health.

Syed (2016) discussed a fog computing pattern, contributing to the field of pattern languages in programs. Patterns often encapsulate proven solutions to recurring problems, and this work may offer insights into common challenges and effective solutions within the context of fog computing.

Wadhwa and Aron (2018) presented at IEEE International Conference on Parallel & Distributed Processing, likely offer a comprehensive exploration of Fog computing integrated with the IoT. The paper may delve into the architecture, applications, and future directions of this integration, shedding light on the evolving landscape of Fog computing in conjunction with IoT.

Yuan and Li (2018) introduced a reliable and lightweight trust computing mechanism for IoT edge devices. The authors may discuss how multi-source feedback information fusion contributes to enhancing trust in IoT edge devices, addressing reliability and security concerns associated with these devices. The research is likely to be beneficial for understanding and implementing trust mechanisms in IoT edge computing environments.

Both fog and edge computing aim to process data more efficiently, enabling faster response times and reducing the need for long-distance data transmission and storage. These technologies will continue to play a crucial role as we move towards more interconnected and data-intensive systems.

2.3 IoT and Fog Computing Applications

Popular Fog computing applications include smart grids, smart cities, smart buildings, vehicle networks, and software-defined networks. Despite the broad utilization of cloud computing, some applications and services still cannot benefit from this popular computing paradigm due to inherent problems of cloud computing such as unacceptable latency, lack of mobility support, and location awareness.

Aljumah (2018) discussed Fog Computing related security issues. They stated that Fog Computing poses a threat to privacy and security of the data and services. The existing security and privacy mechanisms of cloud computing cannot be applied to Fog computing due to the distributed structure, mobility, and heterogeneous nature. This paper provides an overview of present issues in Fog computing.

Aljumah and Ahanger (2018) conducted a comprehensive examination of Fog computing and its associated security issues, presenting their work at an international conference. Reviews, such as theirs, are essential for synthesizing existing knowledge and providing valuable insights into the security challenges associated with Fog computing. This work is particularly valuable for those seeking a consolidated understanding of these issues.

Alonso (2017) proposed Fog computing using public resource computing and storage. Authors introduced the idea to use public-resource computing and storage techniques to shift the workload of the cloud. In this idea, the devices work as participants, who form a data center between cloud and end devices. The participant can be any type of device, from a traditional personal computer to smartphones or tablets, etc. The authors simulate the use of participating nodes in video transfer applications and their results show that the proposed system can be used to solve the bandwidth and computation issue that affects the cloud storage system the author concluded that their system is a feasible solution for applications that process or store public data.

Alrawais, Alhothaily, Hu, and Cheng (2017) delve into the intersection of Fog computing and the IoT, emphasizing security and privacy issues. Given the proliferation of IoT devices and the crucial role of Fog computing in their ecosystem, this research likely scrutinizes challenges related to data security and

user privacy. The findings are likely to be pertinent for professionals engaged in designing and securing IoT systems.

Alwarafy (2020) presents a comprehensive survey addressing security and privacy issues in edge-computing-assisted IoT. Published in the IEEE IoT Journal, their work offers a systematic examination of security and privacy concerns in the context of edge computing and IoT. The authors provide insights into the current state of research and outline potential directions for addressing security challenges in this evolving landscape.

Badidi, and Ragmani (2020) propose an architecture for QoS[32]fog service provisioning, contributing to the optimization of Fog computing resources. Published in Procedia Computer Science, their work focuses on enhancing the provisioning of services in fog environments while considering QoS requirements. The authors offer a detailed architecture and discuss its implications, providing valuable insights for researchers and practitioners aiming to improve service quality in fog computing.

Chiang and Zhang (2016) provide an overview of research opportunities at the intersection of fog computing and the IoT. Published in the IEEE IoT Journal, their work addresses the evolving landscape of research possibilities within the context of fog computing and IoT. The authors discuss key challenges, potential applications, and future directions, offering valuable insights for researchers and practitioners exploring this dynamic field.

Delfin (2019) explores the emergence of Fog computing as a new era in Cloud computing. Presented at the International Conference on Computer, Mathematics, and Control, their work contributes to defining the landscape of Fog computing as a distinct paradigm. The authors discuss the characteristics and potential applications of Fog computing, providing valuable insights for researchers, practitioners, and enthusiasts aiming to understand the evolving dynamics of cloud computing.

Din (2018) Published in IEEE Access in 2018work focuses on trust management techniques for the IoT. The study likely offers a survey of existing trust mechanisms,

[32]Quality of Service

addressing the unique challenges posed by the IoT environment. Trust is crucial in IoT systems, and this research is likely to provide valuable insights for ensuring secure and reliable IoT deployments.

Ema (2019) assesses the suitability of integrating Fog computing alongside Cloud computing. Contributes to the ongoing discourse on the complementary roles of Fog and Cloud computing. The authors discuss the advantages and challenges of using Fog computing alongside cloud resources, offering valuable insights for researchers and practitioners navigating the integration of these computing paradigms.

Gandotra and Lall (2020) focus on the evolution of air pollution monitoring systems, particularly for green 5^{th}Generation, transitioning from cloud to edge computing their work contributes to the development of environmentally conscious solutions for air quality monitoring. The authors discuss the shift from traditional cloud-based systems to edge computing, providing valuable insights for researchers and practitioners involved in environmental monitoring and the deployment of green technologies.

González-Martínez (2015) presented a comprehensive survey on the intersection of cloud computing and education, offering a state-of-the-art analysis. Published in Computers & Education, their work is instrumental in understanding the transformative impact of cloud technologies on educational practices. The authors delve into various facets, from infrastructure to pedagogical approaches, contributing a valuable resource for educators, policymakers, and researchers in the field.

Heck (2018) investigates the current status and future trajectories of IoT applications within the realms of fog and edge computing. The research work provides a roadmap for understanding the evolving landscape of IoT deployment. The authors discuss potential applications and challenges, contributing valuable insights for researchers, developers, and industry professionals involved in the convergence of IoT, fog, and edge computing.

Henze (2020) addresses the critical issue of data handling requirements in cloud storage systems, providing insights into compliance measures. Published in the IEEE Transactions on Cloud Computing, their work contributes to the evolving

landscape of cloud security. By emphasizing the importance of adhering to data handling standards, the authors offer valuable guidance to practitioners and researchers navigating the complexities of secure cloud storage.

Huttunen (2019) explores the synergy of big data, cloud computing, and data science in the domains of finance and accounting their work sheds light on the transformative impact of cutting-edge technologies in financial practices. The authors provide valuable insights into the applications and implications of big data and cloud computing in the financial sector.

Kaur (2020) conducted a systematic literature review addressing security issues within the Fog computing environment. Published in the International Journal of Wireless Information Networks, their work critically examines the current state of security concerns in Fog computing. The authors provide a comprehensive overview, identifying challenges and proposing potential solutions, making it a valuable resource for researchers and practitioners navigating the security landscape of Fog computing.

Khan (2017) presented a comprehensive review of Fog computing security, focusing on current applications and security solutions. Published in the Journal of Cloud Computing, their work provides a detailed analysis of security challenges in Fog computing environments. The authors explore existing applications and propose security solutions, contributing valuable insights for researchers, practitioners, and policymakers concerned with the secure implementation of Fog computing technologies.

From the view of Lai (2021) discussed that the application of the IoT and Fog computing are essential to the development of smart cities because they allow for the administration and interchange of massive amounts of data. The expansion of transportation, tourism, industries, as well as business, has been made possible in recent years by the two major sectors of Fog computing and the IoT. Therefore, the establishment of a smart city will require careful research as well as approaches to use technology innovation to increase the city's strengths. Increase the city's power on numerous fronts. To address the difficulties of network scalability and large data processing, we have explored the advantages of Fog computing in this study using an IoT architecture that is integrated with Fog computing. As a result, the IoT

system's design is created in a way that allows the smart city to operate more effectively through network transmission, information processing, and intelligent perceptions.

Li (2016) provides an overview of the progress and challenges in mobile edge computing. Presented at the IEEE Mobile Cloud, their work addresses the dynamic landscape of mobile edge computing, discussing advancements and potential hurdles. The authors contribute valuable insights into the current state of research and development in mobile edge computing, making it a useful resource for researchers and practitioners exploring the integration of computing resources at the network edge.

Mao (2017) conducted a survey focusing on mobile edge computing from the communication perspective. Published in the IEEE Communications Surveys & Tutorials, their work offers a comprehensive analysis of mobile edge computing, emphasizing communication aspects. The authors delve into key challenges, solutions, and research directions, providing valuable insights for researchers, practitioners, and industry professionals involved in mobile edge computing.

Marbukh (2019) FoNUM[33] for the effective management of Fog computing resources, the research work contributes to the optimization of resource utilization in fog computing environments. The author introduces FoNUM as a framework, discussing its implications and potential applications. This work is valuable for researchers and practitioners seeking to enhance the efficiency and utility of Fog computing resources.

Matrouk (2021) says that applications for the IoT have emerged as the most significant methods in the world for facilitating interactions between people and objects to improve quality of life in recent years. Consequently, as more devices are employed in these applications, massive volumes of data will be generated. In 2012, Cisco put out the concept of Fog computing, which sits in between end users 'IoT devices and cloud computing. While Fog computing does not completely replace Cloud computing, it does lessen its downsides, increase its efficiency, and offer storage and computing capabilities at the edge of the internet. The job is a tiny

[33] Focuses on Fog Network Utility Maximization

portion of work that needs to be finished in a certain amount of time. Task scheduling becomes difficult in Fog computing because of the varied and scattered resources it incorporates. To find the best solution for the NP-hard issue of task scheduling, efficient task scheduling techniques must be used. In the preceding years, several scheduling algorithms were suggested; most of them were used in cloud computing, while a smaller number were used in fog computing. The primary current scheduling methods in fog computing are reviewed and analyzed in-depth in this study.

Mebrek (2017) proposes an efficient green solution addressing energy consumption and delay considerations in the context of IoT-fog-cloud computing. Presented at the IEEE NCA[34], their work contributes to the development of environmentally sustainable solutions for the integration of IoT, fog, and cloud computing. The authors discuss the challenges and benefits of their proposed approach, offering insights for researchers and practitioners aiming to enhance the efficiency of energy usage in such computing environments.

Naha (2018) conducted a comprehensive survey examining trends, architectures, requirements, and research directions in Fog computing. Published in IEEE Access, their work provides an extensive overview of the evolving landscape of Fog computing. The authors delve into key aspects, offering insights into trends, architectural considerations, and future research directions. This survey is a valuable resource for researchers, practitioners, and policymakers interested in the state of Fog computing.

Parikh, Dave, Patel, and Doshi (2019) explore security and privacy issues in cloud, fog, and edge computing. This research likely offers a comparative analysis of security challenges across these computing paradigms, providing insights into the unique considerations for each. As computing architectures continue to evolve, understanding the nuanced security and privacy issues is crucial, making this work relevant for researchers and practitioners alike.

[34] Network Computing and Applications

According to Priyadarshinee (2021) by identifying the crucial success criteria in the Indian context, the paper offers a two-stage SEM-ANN[35] model for the establishment of Smart cities using Fog Computing and the IoT. The study offers a brand-new element, called fog computing. The IoT is further broken down into three independent variables: IoP[36], IoS[37], and IoE[38]. A study of 13 smart cities and 379 respondents was used. ANN[39] and SEM[40], which quantify both linear and non-linear interactions respectively, are used to analyze the data. The results of SEM show that Fog computing has significantly benefited from the IoT, IoP, and IoS. The sole exception in the study for the direction of future research in this field is that IoE hurts Fog computing. IoT was shown to have a significant impact on Fog computing in the subsequent layer of ANN analysis using the Structural Equation Modeling accepted variables as input Fog computing. IoT was shown to have a significant impact on Fog computing in the subsequent layer of ANN analysis using the SEM-accepted variables as input Fog computing. Results from the Structural Equation Modeling and neural network are also compared.

Sha (2020) conducted a survey examining edge computing-based designs specifically addressing security concerns in the IoT. Published in Digital Communication and Networks, their work explores the intersection of edge computing and IoT security. The authors provide an overview of existing designs, shedding light on the current landscape of security solutions for IoT within the context of edge computing.

Stojmenovic and Wen (2014) presented at the Federated Conference on Computer Science and Information Systems, introduced the Fog computing paradigm. This foundational work likely outlines various scenarios where Fog computing can be applied and explores associated security issues. As pioneers in the field, the authors contribute key insights into the potential applications and security considerations of Fog computing.

[35]Structural Equation Modeling-Artificial Neural Network
[36]Internet of People
[37]Internet of Services
[38]Internet of Energy
[39]Artificial Neural Networks
[40] Structural Equation Modelling

Yi (2015) conducted a survey addressing security and privacy issues in Fog computing. Presented at the International Conference on Wireless Algorithms, Systems, and Applications, their work contributes to understanding the multifaceted challenges surrounding security and privacy in Fog computing environments. The authors explore existing issues and propose potential solutions, offering valuable insights for researchers, practitioners, and policymakers concerned with the secure implementation of Fog computing technologies.

Zhang (2018) provide a comprehensive examination of security and trust issues within Fog computing. As Fog computing becomes increasingly integral to network architectures, understanding the associated challenges becomes paramount. The authors likely explore various dimensions of security concerns, ranging from data integrity to trust establishment. The survey format indicates a broad analysis of existing literature, offering valuable insights for researchers, practitioners, and decision-makers navigating the evolving landscape of Fog computing security.

The study's findings support the building of additional Smart Cities and assist India's government reach its goal of creating 100 SC[41] as it moves toward sustainable development IoT and Fog computing are revolutionizing industries by connecting everyday objects to the internet and processing data closer to the source. They find applications in smart cities, healthcare, manufacturing, and agriculture. In smart cities, they monitor traffic and enhance public safety. Healthcare benefits from remote patient monitoring, while manufacturing gains real-time quality control. Agriculture uses IoT for precision farming. These technologies enable real-time data analysis and control, transforming the way we live and work by increasing efficiency and reducing latency in various sectors.

2.4 Fog Computing and Smart Cities

Fog computing is a technology that is rapidly influencing emerging digital technologies and applications. There are many challenges in maintaining the data through Fog computing technologies, these challenges are mentioned along with their security concerns that help for the transition from cloud to Fog

[41]Smart Cities

Abbas, Shaheen, Elhoseny, Singh, and Majid (2018), introduce a novel approach to developing sustainable smart cities using self-regulated agent systems and Fog computing. It employs systems thinking to address the complexities of urban environments, offering valuable insights into creating intelligent and eco-friendly cities. The integration of self-regulated agent systems and Fog computing showcases a forward-looking perspective on urban development.

Ahmed (2019) delivers a thorough examination by presenting a comprehensive taxonomy and a set of requirements tailored for Fog computing applications. It systematically categorizes various applications and delineates crucial criteria that are integral to their successful implementation within Fog computing environments. The work serves as a valuable resource for both researchers and practitioners, offering a structured guide for navigating the intricate landscape of Fog computing applications. With its detailed classification and outlined criteria, the study contributes significantly to enhancing the understanding and practical deployment of Fog computing technologies.

Adel (2020) says that the architecture dispersed throughout a region is referred to as Fog computing architecture. This architectural arrangement primarily focuses on software for the aim of constructing a good network, as well as physical and logical network components. Users may communicate flexibly thanks to Fog computing architecture, which also makes sure that storage services are kept up to speed for handling data. However, it has been noted that the real-time application aspect of Fog computing architecture has greatly increased its relevance in the field of education. The survey's primary goal is to provide a comprehensive literature evaluation of the technology of Fog computing in the IoT system for education. The survey also concentrates on assessing the crucial elements that play a significant role in the fields of education as well as looking into the limitations and results related to the use of Fog computing technologies in educational systems from the perspectives of privacy, security, and agility.

Al-khafajiy, Baker, Asim, Guo, Ranjan, Longo, Puthal, and Taylor (2020) this research introduces COMMITMENT, a fog computing trust management approach, addressing the critical aspect of trust in distributed environments. The proposed

model contributes to enhancing trustworthiness in fog computing systems, making it a significant addition to the field of trust management in decentralized computing.

Alavi, Jiao, Buttlar, and Lajnef (2018) focused on the integration of the IoT in smart cities, this paper presents a state-of-the-art review and outlines future trends. It provides an extensive overview of IoT-enabled smart cities, offering valuable insights into current advancements and potential directions for future development.

Arcaini, Riccobene, and Scandurra (2015) research work delves into the modeling and analysis of MAPE-K[42] feedback loops for self-adaptation in software systems. By focusing on the MAPE-K feedback loop, the authors provide insights into the mechanisms of self-adaptation, contributing to the broader field of autonomic computing.

Atlam, Walters, and Wills (2018) according paper comprehensively explores fog computing in conjunction with the IoT. It offers a critical analysis of the role of Fog computing in IoT scenarios, highlighting its potential benefits and challenges. The work serves as a valuable resource for researchers, practitioners, and policymakers interested in the intersection of fog computing and IoT.

Badraddin (2019) explore the effectiveness of service decomposition in Fog computing architecture for the IoT. It investigates how breaking down services in Fog computing environments can enhance the efficiency of IoT applications. The findings contribute to optimizing the design and deployment of services in Fog computing for improved IoT functionality.

Baouya, Chehida, Bensalem, and Bozga (2020) presented at the Mediterranean Conference on Embedded Computing, this paper explores the joint use of Fog computing and blockchain for massive IoT deployment. It investigates the synergies between Fog computing and blockchain technologies, providing insights into potential applications and benefits in large-scale IoT scenarios.

Bellavista, Berrocal, Corradi, Das, Foschini, and Zanni (2019) they surveyed offers an in-depth examination of Fog computing in the context of the IoT. It covers a wide

[42] Monitor, Analyze, Plan, Execute, and Knowledge

range of topics, including architectures, communication protocols, and application domains specific to Fog computing for IoT. The survey is a valuable resource for researchers and practitioners seeking a comprehensive understanding of the intersection of Fog computing and IoT.

According to Bonomi (2012), they surveyed that the characteristics of Fog Computing make it an appropriate platform for critical IoT services and applications like connected cars, smart grids, smart cities, and wireless sensor and actuator networks. The author mentioned three main applications that are suitable for the Fog computing architecture. First, connected vehicles, where cars are connected to other cars and the surrounding environment. Second, smart grid, is an intelligent power supply system for widely distributed suppliers and consumers. Third, wireless sensor networks, are widely geographically distributed sensors for environment monitoring systems.

Bosman, Lukkien, and Verhoeven (2011) work focuses on gateway architectures for service-oriented application-level gateways. It addresses the challenges and considerations in designing gateways that facilitate communication between service-oriented applications. The research is particularly relevant for those involved in the development and optimization of gateway systems.

Bourque and Fairley's (2014) research work outline the knowledge areas essential for software engineering practitioners. As a widely recognized resource, Software Engineering Body of Knowledge serves as a guide for educators, professionals, and organizations in understanding the key concepts and practices in software engineering.

Brogi, Forti, and Ibrahim (2018) researcher investigates the cost aspects associated with deploying fog applications. By addressing the economic considerations of deploying applications in Fog computing environments, the authors contribute to the understanding of the financial implications and challenges in Fog application deployment.

Cheng (2018) proposed easy programming of IoT services, authors stated the issues in current programming models. As per author the most of the existing Fog Computing frameworks either lack services programming models or defined

programming models based on their own private data model and interfaces. So, the openness and interoperability of smart cities are quite limited. To tackle this problem authors proposed an approach to design and implement a new Fog Computing framework, named Fog Flow, for IoT smart city platforms. The proposed framework may allow developers to program elastic IoT services easily over the cloud and edges. In this paper to showcase how smart city use cases can be used with Fog Flow authors describe different use cases and implement the application for anomaly detection of energy consumption in smart cities.

Dang (2017) addressed the data security and performance issues. For this, the authors proposed a Region-based Trust-Aware technique for trust-based computation allocation to fog nodes of the region. The authors also proposed privacy-aware role-based access control for fog nodes and also developed a mobility management service that handles the changes in users and fog device's locations.

Ghobaei-Arani (2020) discusses the fundamental difficulty in fog computing is scheduling. Tasks that need computational intensity and tasks that require data intensity are separated into two categories in a Fog environment. The task execution time is shortened because the scheduler migrates the data to the high-productivity resource when scheduling jobs that need intensive computation. On the other hand, it is tried to minimize the amount of data transfer when scheduling the tasks needing data intensity. As a result, data transfer takes less time.

Hamza and Attila (2020) explore approaches to integrating blockchain with Fog computing, addressing the intersection of these two emerging technologies. The work provides insights into the challenges and opportunities associated with combining blockchain and Fog computing, contributing to the ongoing discourse on secure and decentralized systems.

Hutun, Sariand Austerberg (2019) investigated the security implications of Fog computing in the IoT, this paper provides insights into potential security challenges and solutions. It contributes to the growing body of knowledge on securing IoT ecosystems, particularly in the context of Fog computing.

Khakimov (2018) discussed the study of Fog Computing Structure. As the authors discussed, the growth of mobile traffic, the support of mobility, and geometric

distribution are no less important. So, the emergence of Cloud computing for centralized storage, retrieval, and management of information, and integration to different mobile applications is an important task, for Fog computing is introduced by Cisco, which is designed for local processing of tasks on Fog devices. In this paper, authors emulated the operations of network nodes under Fog computing conditions.

Kumari (2019) says that current power grid system has to be upgraded since there is a rising daily demand for electricity. The contemporary, ICT-based smart grid has already taken the place of the conventional electricity system. As the volume, velocity, and diversity of the data generated by Singapore's smart meters fluctuate, it is difficult to store, handle, and evaluate. Cloud computing, which offers real-time response for several applications, is used to store and analyze the data. Fog computing is a novel technique that places most of the computer resources near to the end users, and has evolved to address the latency issue during data processing. To bridge the gap between the processing power of remote data centers and SDs[43] in SG[44] systems, it functions as a bridge between SG and Cloud computing. In the forthcoming fifth generation, it is necessary to set up an advanced sensors and measurement system with a communication network backbone to address the difficulties 5th Generation. To determine the amount of energy needed by smart devices at the fog layer, we have addressed the architecture of SG about Fog computing in this work. Additionally, the setting of 5thGeneration network infrastructure is examined about the communication and computing components. The authors investigate the impact of Fog computing on reaction time, transmission delay, and energy management expenses for applications with a high sensitivity to delays.

Luigi (2010) considered a seminal work, this survey by Atzori, Iera, and Morabito provides a comprehensive overview of the IoT. It covers key aspects such as architectures, communication protocols, and application domains, offering a foundational understanding of IoT concepts. The survey has been widely cited and remains a fundamental reference in the IoT literature.

[43]Smart Devices
[44]Smart Grid

According to Qasem (2020), Fog computing is a novel network architecture and computing paradigm that performs some processing tasks using user or near-user devices (network edge). As a result, it gives cloud computing greater flexibility than that offered by ubiquity networks. In this research, a flexible hierarchical Fog computing-based smart city is suggested. By utilizing several network designs, including Cloud computing, autonomous network architecture, and ubiquitous network architecture, the suggested design seeks to solve the drawbacks of the earlier methods. In light of this, the suggested method reduces the latency of data processing and transmission with allowed real-time applications, distributes the processing jobs among edge devices to lower the cost of data processing, and enables cooperative data interchange among the applications of the smart city.

The architecture consists of five primary layers that may be raised or combined depending on how much data is processed and sent in a given application. Connection layer, real-time processing layer, neighborhood connecting layer, primary processing layer, and data server layer are the related layers. Utilizing simulating fog computing scenarios, a case study of a unique smart public parking, travel, and direction adviser was built, and the findings demonstrated a considerable reduction in real-time application latency, as well as cost and network use as compared to the Cloud computing paradigm. Additionally, compared to a stationary Fog-computing design, the suggested technique does not significantly compromise time, cost, or network consumption while increasing the scalability and dependability of the users' access.

According to Songhorabadi (2020), the development of smart cities today, particularly in location-aware, latency-sensitive, and security-critical applications (like emergency fire events, patient health monitoring, or real-time manufacturing), heavily depends on more sophisticated computing paradigms that can meet these requirements. Because it is situated closer to the end devices, Fog computing, a strong Cloud computing supplement, plays a significant role in this respect. However, the methods used in smart cities are typically cloud-based, which limits the flexibility and dependability as well as the security and time-sensitive services.

This research work suggests a systematic literature review (SLR[45]) for the cutting-edge Fog-based technologies in smart cities to circumvent the limits of the cloud and other associated computing paradigms, such as edge computing. In addition, a proposed taxonomy is divided into three types based on the content of the examined studies: service-based, resource-based, and application-based. Additionally, each class's evaluation criteria, tools, techniques, merits, and demerits are examined in this SLR. Additionally, each class's proposed algorithm types are specified. By categorizing future trends and difficulties into useful sub-classes, it is possible to give complete and distinct open topics and challenges while also taking into consideration different viewpoints.

Velasco (2018) reviewed distributed and ultra-dense Fog Computing infrastructure, which can be allocated at the extreme edge of wired and wireless networks for telecom operators to provide multiple unified, cost-effective, and new 5thGeneration services, such as Network Function Virtualization, Mobile Edge Computing, and services for third parties e.g., smart cities, vertical industries or IoT. The proposed architecture consists of three main building blocks: a scalable node that is seamlessly integrated into the Telecom infrastructure; a controller, focused on service assurance that is integrated into the management and orchestration architecture of the Telecom operator, and services running on top of the Telecom infrastructure.

Zhang (2020) discussed building "smart cities" makes extensive use of Fog computing and IoT technologies, which has the potential to significantly improve the administration and interchange of urban information. Fog computing and the IoT, two emerging network technologies, may be utilized to make it simpler to create smart cities, which are beneficial for the growth of urban business, industry, and other industries as well as tourist and transit management. As a result, the creation of a smart city will significantly strengthen the city's capacity for overall growth.

[45]Systematic Literature Review

We examine the benefits of Fog computing and suggest a Fog computing-based IoT architecture that successfully addresses the issues of huge data processing and network scalability. Based on this, a layered fog computing network architecture is suggested to improve the city's functioning through different intelligent perceptions, information processing, and network transmission techniques.

2.5 Resource Allocation and Task Scheduling Technique

Task scheduling and resource allocation are mandatory parts of cloud computing research. The efficiency of resource uses depends on the scheduling and load-balancing methodologies, rather than the random allocation of resources. Cloud computing is widely used for solving complex tasks

Aazam (2015) discussed a system for supplying resources in a Micro data center using fog. This document unequivocally indicates that resources are calculated and managed based on variable customer service client relinquish likelihood. Based on previously predicted resources from the service provider, this model offers the best service possible to its clients. This mechanism is evaluated using the Cloud Sim toolkit, and a probabilistic model is utilized to estimate the availability of resources. This technique aids in determining the precise number of resources that the client will require. Additionally, it decreases resource waste, boosts revenue, and may be used in a wide range of cloud service provider scenarios.

Aazam (2015) provided a method to estimate resources for a Fog-based mini data center and created a pricing structure for an IoT. Issues including resource estimation, reservations, and pricing strategy for current and potential clients depending on their attributes were all tackled in this research work. Using data from previous resource usage by cloud service users, resources are allotted to existing clients. As a result, resource prediction and pre-allocation are also based on user behavior and the likelihood that resources will be used in the future. This method makes it simpler for all types of cloud service providers to anticipate how their customers will use their resources.

Abdul-Qawy (2015) introduces the power of information and communications technology ICT[46] as it becomes a vital component of our infrastructure that is essential to our way of life by enabling the connecting of heterogeneous devices in various ways. To mention few, examples include personal computing, sensing, surveillance, smart homes, entertainment, transportation, and video streaming. The Internet is a vital living system that is continually changing and growing, giving rise to new technologies, applications, protocols, and algorithms. As wireless communication trends pick up speed, mobile broadband, and Internet access are becoming increasingly innovative. Communication devices that don't require infrastructure are becoming more common, intelligent, powerful, connectable, compact, affordable, and simple to install. This introduces a new direction for ICT civilization in the future the IoT. The IoT, formerly known as Machine-to-Machine communications, has recently gained prominence in the ICT industry and research communities. We present an overview of the IoT paradigm, its concepts, principles, and prospective advantages in this article. We concentrate on the key IoT technologies, developing protocols, and well-known applications. This introduction might be useful for anyone who wants to learn more about the IoT and get involved in its development.

Abomhara (2014) discussed connecting common things, connectivity fosters the growth of the IoT. Because even little interactions between these things might contribute to collective intelligence in an IoT network, the connectivity of these objects is crucial. It makes things compatible with and accessible via networks. By connecting smart items and apps, this connection can open up new commercial prospects for the IoT. The IoT requires connectivity that goes beyond just attaching a Wi-Fi module and calling it a day. Network compatibility and accessibility are made possible via connectivity. Accessibility involves joining a network, whereas compatibility gives everyone the same capacity to use and create data. If this seems familiar, it's probably because Metcalfe's Law applies to the IoT

[46]Information and Communications Technology

Attar and Sutagundar (2018) research work provides a survey on resource management for fog-enhanced services and applications. It comprehensively reviews existing approaches and strategies for managing resources in Fog computing environments. The survey serves as a valuable resource for researchers and practitioners seeking an overview of the state-of-the-art in fog computing resource management.

Bitam (2018) discussed the better distribution of a set of jobs among all the Fog computing nodes and suggested a BLA[47]. When compared to the genetic algorithm and particle swarm optimization algorithms, the suggested approach performs better in terms of execution time and memory allocation. However, the suggested technique has a low degree of scalability. Additionally, it ignores the dynamic work scheduling. Furthermore, the approach has only been evaluated by the BLA, authors on tiny datasets, and task execution reaction time is lengthy.

Chen (2014) discussed the main function of the IoT is data collection from its surroundings, which is made possible by the dynamic changes that occur around the devices. These devices' states fluctuate dynamically, for instance, whether they are sleeping or waking up, connected or not, and in various contexts that include temperature, location, and speed. The quantity of devices fluctuates dynamically with a person, place, and time in addition to the status of each gadget. Device context, such as location and speed, as well as the states of devices, such as sleeping and waking up, connected and/or disconnected, also alter dynamically. Additionally, the quantity of devices may fluctuate.

Confais (2017) proposes a first-class object store service for fog facilities. The proposed system is built with Scale-out Network Attached Storage 2 system and Inter Planetary File System 3, a Bit Torrent-based object store spread throughout the fog infrastructure. Authors used Scale-out Network Attached Storage on each site to reduce inter-site exchanges that are mandatory for metadata management in Inter Planetary File System implementation. This experiment gives direction to improve the performance and fault tolerance of Fog.

[47]Binary Lion Algorithm

Edemacu and Bulega (2014) explore the resource sharing between M2M and H2H[48] traffic under a time-controlled scheduling scheme in Long-Term Evolution networks. The research addresses the challenges of efficiently allocating resources to accommodate both M2M and H2H communications, contributing to enhanced network performance.

As per the research, Giordano (2016) Global Smart City projects are made possible by new IoT applications that make use of ubiquity in connection, big data, and analytics. With the help of these brand-new apps, users will be able to monitor, manage, and control devices remotely as well as extract valuable insights and information from huge streams of real-time data. The adoption of new paradigms is necessary to support this novel approach. To develop control systems based on the decentralization of control functions across distributed autonomous and cooperative entities that are operating at the edge of the network, agent technology is integrated with the emerging notion of Fog computing in this study. The Rainbow platform aims to minimize the computation's distance from the physical component. Using adaptive and decentralized algorithms that take advantage of the concepts of collective intelligence, multi-agent systems running on top of Rainbow develop smart services.

As per Gu (2015) a methodology for heuristic resource management in fog was put forth by the author. For the formulation of this problem, mixed linear programming and mixed nonlinear programming serve as the fundamentals. To cut costs in Medical Cyber-Physical Systems, they proposed a two-step nonlinear heuristic approach. They demonstrated that this method performs better than previous algorithms by comparing it to the existing greedy technique. Due to the high computing complexity of the mixed integer linear programming model, they used a two-phase linear heuristic approach to lower the cost of medical cyber-physical systems.

Hamdoun, Rachedi, and Ghamri-Doudane (2015) the paper introduces an interference-aware bipartite graph approach for radio resource sharing in MTC[49]

[48]Human-to-Human
[49]Machine Type Communication

within Long-Term Evolution Advanced Networks. The work addresses the challenges of efficient radio resource allocation for MTC applications, contributing to improved communication reliability and performance.

Hassan (2015s) authors focus on the most current research directions in this area to highlight the IoT idea in general and discuss the major issues of the IoT ecosystem. IoT a new technology that expresses a contemporary wireless communications network, may be characterized as an intelligent and interoperable node connected to a dynamic global infrastructure network. It also aims to execute the connection notion of everything from anywhere at any time. The IoT environment has a wide range of challenges that have an impact on their performance. These challenges can be categorized into two groups: i) General challenges, like communication, heterogeneity, virtualization, and security; and ii) Unique challenges, like WSN[50], and QoS, which is thought of as a factor that unites both general and special challenges.

As per Heer (2011) is focused on the security problem in IP-based IoT systems. The author claimed that the internet serves as the foundation for all device connectivity in an IoT system. Security concerns in IP[51]-based IoT systems are therefore a major problem. Additionally, the capabilities and life cycle of every IoT system object should be taken into account while designing the security architecture. It also incorporates the use of security standards and a trusted third party. It is also desired to have a security architecture that can grow to accommodate both small- and large-scale IoT objects. The study made the point that because the IoT has spawned a new kind of cross-network communication between various objects, standard end-to-end internet protocols are unable to accommodate this communication. Therefore, to assure end-to-end security, new protocols need to be established taking the translations at the gateways into account. All communication-related levels have their security concerns and needs. As a result, if just one layer's criteria are met, the system will be susceptible, hence security needs to be provided for all layers.

[50]Wireless Sensor Networks
[51]Internet Protocol

Hoang and Dang (2017) present paper an optimization approach for task scheduling in fog-based regions and the cloud. The work addresses the coordination and scheduling challenges in distributed fog and cloud environments. Optimizing task scheduling enhances resource utilization and performance in fog computing architectures.

Jia, Hu, Zeng, Xu, and Yang (2018) authors present a double-matching resource allocation strategy designed for Fog computing networks with a focus on cost efficiency. This strategy aims to optimize the allocation of resources to tasks in a way that enhances cost-effectiveness. The work contributes to the economic viability of fog computing networks.

Kimovski (2018) introduces an adaptive nature-inspired fog architecture, presenting a novel approach to designing Fog computing systems. The architecture draws inspiration from natural systems to enhance adaptability and efficiency in fog environments. By presenting this innovative perspective, the authors contribute to the development of more resilient and self-adapting Fog computing infrastructures.

Liu (2018) proposed object tracking using fog-based intelligent surveillance in public spaces. In this system, Fog computing platform was deployed to accelerate the proposed tracking approach. The tracker was constructed to take multiple positions' detections. The detection position was then adjusted as per the optical flow of the object and the alternate template was stored with the template update mechanism, and all were computed at the fog layer.

Li, Zhao, Gong, and Zhang (2019) researcher address energy-efficient computation offloading and resource allocation in fog computing environments for the Internet of Everything. The authors propose strategies to optimize the allocation of resources and the offloading of computations, contributing to the energy efficiency of fog-based Internet of Everything systems.

Liu, Qi, Zhou, and Wu (2018) the authors propose a task-scheduling algorithm for fog computing environments based on classification mining. This algorithm leverages data classification techniques to optimize task scheduling in fog computing systems. The work addresses the challenge of efficient resource

utilization in dynamic fog environments, contributing to the improvement of task execution in such settings.

Liu, Yang, Wang, and Mao (2018) propose a dispersive stable task scheduling algorithm designed for heterogeneous fog networks. This algorithm aims to optimize task scheduling by considering the diverse characteristics of nodes in heterogeneous fog environments. The work contributes to the efficient utilization of resources in fog networks with varying capabilities.

Mohan and Thangavel (2013) focus on resource selection in grid computing environments, incorporating trust evaluation using feedback and performance metrics. The work addresses the challenges of selecting trustworthy and efficient resources in grid environments, contributing to the overall reliability of grid-based computing systems.

Ni, Zhang, Jiang, Yan, and Yu (2017) research work introduces a resource allocation strategy for Fog computing based on priced timed Petri nets. This modeling approach enables the efficient allocation of resources in fog environments, considering both time and cost factors. The strategy contributes to the effective utilization of resources in Fog computing architectures.

Paharia (2018) says that Fog Computing is a protective mechanism against Distributed Denial of Service attacks. The authors proposed an architecture to block the malicious traffic generated by the Distributed Denial of Service attack from user to cloud. Fog functions as a filtering layer for the traffic generated. This study primarily works to improve the overall performance of the network and enhances the reduction in traffic forwarded to the cloud.

Pham et al. (2017) research work introduces an innovative and cost-effective approach for task scheduling, focusing on collaborative efforts between cloud and fog computing. The primary goal is to dynamically allocate tasks in a manner that optimizes both cost and performance considerations. By adopting a collaborative strategy, the proposed approach enhances resource utilization efficiency and improves the overall execution of tasks. This novel method not only addresses the challenges of task allocation in a dynamic environment but also contributes to achieving a balance between cost-effectiveness and high performance. The

integration of cloud and fog computing in task scheduling demonstrates a forward-thinking solution that can potentially bring about significant advancements in optimizing computational resources and task execution efficiency.

Prakash and Ravichandran (2012) research authors propose an efficient resource selection and binding model for job scheduling in grid computing environments. This model aims to optimize the allocation of resources for executing jobs, contributing to improved efficiency and performance in grid computing systems.

Rahmani (2015) says that IoT communication is dominated by wireless nodes. Many wireless protocols are tuned to utilize less power for functioning, limited communication, or increased coverage range due to resource limitations at the perception layer. Currently, the sector offers a wide range of various approaches. To speed communication with the cloud layer and combine these many wireless protocols, the fog layer is in a perfect position. As a result, system reliability is increased, security is provided, communications between devices are routed, and the administration of sensor and actuator subnetworks is aided. Furthermore, this layer enables the compatibility of diverse protocols by detecting and comprehending the representation format. Non-IP-based devices may now be seen and reached over the Internet thanks to the Fog layer.

According to Ravi (2016), Fog's architectural design makes it simple to operate and communicate with the devices on the platform. The physical layer is the foundational component of a Fog computing system. This layer is in charge of connecting many tools or devices to a common platform and facilitating information flow. There are devices, terminals, sensors, and virtual sensors in the physical layer. The nodes of this layer are managed according to their purposes, and sensors in this layer are decentralized to detect data from surrounding places faster and communicate it to a higher layer of the architecture. The monitor layer, which is the second layer in the Fog computing architecture, keeps an eye on resource utilization as well as the performance of nodes and sensors. The preprocessing layer, which is the next layer in the Fog computing architecture, is in charge of maintaining the record data and carrying out information analysis.

Saini (2019) says that a worldwide network made up of people, intelligent things, smart gadgets, information, and data transformed thanks to the IoT. It goes without saying that as more gadgets connect to the internet, the difficulties in protecting the information they broadcast and the communications they start grow. IoT device usage has increased significantly over the years, mostly in the home and in manufacturing. With the former, a whole ecosystem centered on Amazon's Echo devices that make use of the Alexa Voice Service has been created. Apple, Google, and Microsoft have all done the same. The onus of protecting the devices falls to the platform providers because they are separate, closed platforms. We focus on cyber security in manufacturing and associated industries in this study. As more and more equipment and devices are brought online, sectors including manufacturing, oil and gas, refining, pharmaceuticals, food and beverage, water treatment, and many more are continually trying to add the necessary levels of security. Manufacturers of devices and plant operations managers are under continual pressure to safeguard their physical assets from cyber threats. Additionally, there are significant differences between each of these businesses' data types, IoT device topologies, threat management challenges, and compliance requirements.

In the research study done by Shalini (2019), paradigm of Cloud computing has been advanced or extended by Fog computing. It is a sophisticated distributed system that works over the whole network; it keeps data near to the user and speeds up information delivery. In this study, the architecture of Fog computing is discussed. The various levels are used to describe how Fog computing functions. The downsides of Cloud computing and how Fog computing addresses them were explored, along with certain obstacles, unresolved problems, and Fog computing applications in many sectors.

Skarlat (2016) provided a method to estimate resources for a Fog-based mini data center and created a pricing structure for an IoT. Issues including resource estimation, reservations, and pricing strategy for current and potential clients depending on their attributes were all tackled in this research work. Using data from previous resource usage by cloud service users, resources are allotted to existing clients. As a result, resource prediction and pre-allocation are also based on user behavior and the likelihood that resources will be used in the future. This method

makes it simpler for all types of cloud service providers to anticipate how their customers will use their resources CSP[52]. Fog colonies are small data centers made up of a lot of Fog cells. The network may switch from centralized to decentralized processing with the help of the Fog colonies, which also facilitate requests from the fog cells for resource provisioning tasks. By managing fog cells, Fog orchestration manages the Fog colony and offers resource provisioning services via the Fog cloud interface.

Sun and Zhang (2017) present a resource-sharing model based on a repeated game in Fog computing environments. The model explores the dynamics of resource sharing over time, considering the repeated interactions between nodes. The proposed approach contributes to a deeper understanding of resource-sharing dynamics in Fog computing systems.

Wang (2019) proposes to address the issue of terminal devices with constrained computational capabilities, excessive energy consumption, and offer a job scheduling HH[53] algorithm for various fog nodes. The IPSO[54] method and the IACO[55] algorithm is combined in the HH algorithm. MATLAB is used by the writers to evaluate their work. Results of the experiment demonstrate that the algorithm outperforms IPSO, IACO, and RR[56]on three performance criteria Make-span, energy consumption, and reliability. After all, the clustering of tasks and fog nodes is not covered by this approach.

The research study was done by Yin (2018), As an extension of cloud computing, Fog computing has been proposed to offer processing, storage, and network service at the network edge. If the intermediary layer between the industrial cloud and terminal device is taken into account, Fog computing can offer a plethora of computational and storage capabilities, such as defect detection and status analysis of devices in assembly lines. However, the deployment of novel virtualization technologies in the job scheduling and resource management of Fog computing is hampered by limited resources and low-latency services. As a result, we create a

[52]Cloud Service Providers
[53] Hybrid Heuristic
[54]Improved Particle Swarm Optimization
[55]Improved Ant Colony Optimization
[56]Round Robin

new work scheduling model by taking containers into account. Then, to guarantee task completion on time and maximize the number of concurrent jobs for the Fog node, a task scheduling algorithm is built. Finally, by the properties of the containers, we suggest a reallocation strategy to decrease task delays. The outcomes show that our suggested task scheduling technique and reallocation strategy may successfully decrease task delays and increase the number of processes running concurrently in Fog nodes. Considers the function of containers in a task scheduling paradigm. To increase the number of concurrent jobs for the Fog node and ensure that tasks are completed on time, they also created a task scheduling algorithm. Additionally, they suggested a reallocation strategy to decrease task latency based on the features of the containers. When a job's request is approved, the task scheduler distributes it. The work is delivered directly if it can only be finished in the cloud or the fog node. The task scheduler will need to choose where to put it as long as the cloud and the fog node both successfully execute the task. Low-computing tasks are carried out at fog nodes, whilst high-computing tasks are sent to the cloud. The suggested algorithm and reallocation method shorten task delays and increase the efficiency with which Fog nodes use their resources. However, the authors overlook the computing time on the cloud, which in a practical scenario should be considered. To shorten task execution time, the picture positioning of containers is also an important issue that has to be resolved.

Yin, Luo, and Luo (2018) address task scheduling and resource allocation in Fog computing with a focus on containers for smart manufacturing. This work explores the use of containerization technology to enhance the efficiency of task scheduling and resource allocation in fog environments, specifically in the context of smart manufacturing.

Yang, Wang, Zhang, Chen, Luo, and Zhou (2018) introduced a maximal energy-efficient task scheduling algorithm tailored for homogeneous fog networks. The focus is on optimizing energy consumption in Fog computing environments with uniform node characteristics. The proposed algorithm aims to achieve efficient task scheduling while minimizing energy usage, contributing to sustainability in Fog computing.

Yang, Zhao, Zhang, Chen, Luo, and Wang (2018) introduced a delay energy-balanced task scheduling algorithm designed for homogeneous fog networks. The algorithm aims to balance both delay and energy considerations in the scheduling of tasks, contributing to improved overall system performance. DEBTS addresses the trade-off between latency and energy consumption in fog computing environments.

Yuan (2017) proposed a fast search and find density peaks-based fog node location technique. The authors used a density peaks-based fog node location strategy to locate the fog nodes and determine their resources. To locate fog node authors treated this problem as a clustering problem with different attributes. To perform this, they proposed an improved Fast Search and Find of Density Peaks-based fog node location algorithm, which uses time-sensitive features of IoT applications and improves the fast search and finds density peaks clustering algorithm to make this clustering algorithm more robust and adaptable.

Zhenqi, Haifeng, Xuefen, and Hongxia (2013) research focus on the uplink scheduling algorithm for massive M2M and H2H services in Long-Term Evolution networks. The study addresses the specific challenges associated with coordinating uplink communications for diverse services, contributing to the optimization of network resources.

The literature review is very promising for future research in Fog Computing and its different applications in Smart Cities. In above literature reviewed different proposed Fog Computing architectures. The literature also identified challenges like security and privacy issues, scheduling, and allocation of resources, etc. These challenges can be achieved using research in fog computing, IoT, and traffic congestion through ML[57] revealing the significant impact of ML techniques in addressing traffic-related challenges in IoT-based fog computing environments. Fog computing is a decentralized architecture that extends Cloud computing capabilities to the edge of the network, enabling real-time data processing and reducing latency for IoT devices. ML algorithms are leveraged to analyze the massive data generated by IoT devices, predict traffic patterns, and optimize traffic flow. The review highlights the use of ML-based traffic prediction models, such as time-series forecasting, neural

[57] Machine Learning

networks, and support vector machines, to accurately predict traffic congestion and provide timely information to drivers and transportation authorities. Moreover, the integration of fog computing and ML facilitates real-time data analytics and decision-making, enabling adaptive traffic management and intelligent resource allocation to alleviate congestion and improve overall traffic efficiency. The literature demonstrates the potential of Fog computing and ML in transforming transportation systems, reducing traffic congestion, and enhancing the overall transportation experience in smart cities and urban environments.

Chapter - 3
Research Methodology

Research methodology is the systematic and scientific approach used to conduct research, investigate problems, and gather data for a specific purpose. It involves techniques and procedures to identify, collect, analyze, and interpret data, addressing research questions or solving research problems

3.1 Significance of Research

Research methodology is the systematic approach to solving research problems, it involves selecting appropriate methods for data collection, analysis, and interpretation to ensure the validity and reliability of results. Key components include defining research questions, conducting literature reviews, choosing qualitative, quantitative, or mixed methods, and employing tools like surveys, experiments, or case studies. The proper methodology enables rigorous and reproducible findings, essential for advancing knowledge in their field. Understanding and applying the right methodology is crucial for producing high-quality, impactful research that withstands academic scrutiny.

The methodology ensures the research's validity, reliability, and reproducibility. Key aspects include selecting tools like surveys, experiments, or case studies, and applying statistical or thematic analysis techniques. A well-defined methodology is crucial for producing credible, high-quality research that contributes meaningfully to the academic field.

3.2 Research Gaps

Fog computing has attracted a great number of researchers so, it is a trending topic for research. The literature study motivates research in Fog Computing by introducing a bright future and its application of it. The researchers stated that Fog Computing will show how today's IoT and cloud computing work. The researchers also stated the challenges to be faced in the implementation of Fog Computing in real-life applications. Currently, researchers are working on the implementation of fog for commercial applications. The challenge for further studies and solutions from experts is that we need to keep ourselves updated for online publications and updates from the Open Fog consortium related to Fog Computing.

Even though fog computing has emerged as a potential standard paradigm that offers services to different IoT and mobile devices at the network edge, there are still many

research issues that need to be resolved. Reaching the desired performance level, computing resource provisioning in terms of task offloading, and achieving the best response time with reduced latency are some examples of research challenges due to the heterogeneous nature of fog in terms of node capabilities while residing within the IoT domain.

Fog-Cloud Collaboration is a computing model that uses both fog and cloud computing. Fog computing processes data close to the source, reducing latency, while cloud computing handles large-scale data storage and processing. Together, they provide efficient, flexible data management, enhancing IoT performance, improving security, and supporting real-time applications. This collaboration optimizes resource usage, offering a scalable, sustainable solution for complex computing needs.

Figure 3.1: Data Processing Challenges at Cloud Data Center (Deafallah, 2022)

Figure 3.1 shows cloud data centers encounter several significant challenges in data processing due to the vast scale and complexity of their operations. Managing and processing big data from various sources requires robust distributed storage systems and parallel processing capabilities. Data security and privacy are crucial concerns, necessitating stringent access controls, encryption, and compliance with privacy regulations. Latency and network congestion can impact data processing

performance, motivating the use of content delivery networks and edge computing strategies. Scalability, resource allocation, energy efficiency, data backup, and disaster recovery planning are essential for maintaining optimal performance. Moreover, addressing data processing bottlenecks, handling heterogeneous data formats, complying with data privacy regulations from multiple jurisdictions, and enabling real-time data processing pose additional challenges. Cloud data centers continuously innovate and leverage advanced technologies to overcome these challenges, ensuring efficient and reliable data processing services for their users.

3.3 Problem Statement

Our research aims to understand the importance of cloud computing and fog computing. Fog computing solves problems like delay in response, insufficient bandwidth, no immediate response, security, and reduces the latency issue of cloud computing. The central problem focuses on:

"Smart Fog is a Collaborative Approach to Share Computational Power of Fog Devices for Fog Computing in Smart City IoT Network"

The research work studies the level of computational work, latency issue, and the efficiency of fog computational devices over various parameters like processing speed, scheduling, and task allocation in the fog layer, using fog computing and Machine Learning algorithms to reduce the problem and find trends, issue, challenges, suggestion, and future potential of computing problem in Fog Computing environment will share computational power to IoT devices with low computational power. Overall, the research work finds the use of fog computing networks to solve the future journey.

3.4 Objectives

The objectives of the research are clearly defined goals that guide the study, focusing on specific outcomes to be achieved. They include exploring new areas, describing phenomena, explaining relationships, predicting future events, and applying findings to solve real-world problems. Research objectives are specific goals that guide the focus and outcomes of a study. These objectives can vary widely but generally include exploration, description, explanation, prediction, application, evaluation, theory development, action, documentation, and innovation. These

objectives ensure that the research remains focused, relevant, and systematic. They also help in evaluating the success and impact of the study. The purpose of this research work is to propose a "SMART FOG" protocol based on a technique to connect FOG computational devices which enables devices to share their resources within the Fog network and reduces the latency issue of cloud computing.

1. **To study IoT-based architectures and protocols for understanding the connectivity between IoT devices.**

2. **Analyse the current IoT infrastructure and evaluate various layers of communication protocols to design the SMART FOG Protocol-based technique.**

3. **Exploring the challenges to be faced in implementing the SMART FOG protocol-based technique on computation-enabled devices.**

4. **To explore the task scheduling and allocation techniques for Fog Computing nodes in SMART FOG.**

5. **Determine the fault tolerance mechanism in SMART FOG protocol-based technique by allocating tasks to multiple recipients.**

6. **Discover the efficiency of fog computational devices over various parameters like processing speed, scheduling, and task allocation in fog layer.**

Our research work focuses on the above-stated objective which aims to use the computational power of computation-enabled devices to collaboratively perform tasks and speed up the processing.

3.5 Hypothesis

The hypothesis is nothing but a tentative statement to predict the expected outcomes of a study. Defining hypothesis helps in designing new experiments and observations. The following hypotheses were tested for the system in the proposed research work.

This hypothesis is subdivided into $H_0 1$ to $H_0 6$ to explore the efficiency and various measures of the Smart Fog protocol-based system compared to cloud-based systems, aiming to comprehensively evaluate their impact and effectiveness. The sub hypothesis from $H_a 1$ to $H_a 6$ are as follows:

1. **$H_0$1:** There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure execution time.

2 **$H_0$2:** There is no significant difference between SMART FOG protocol-based System and cloud-based system based on the performance measure latency.

3 **$H_0$3:** There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure energy consumed.

4 **$H_0$4:** There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure cost of execution.

5 **$H_0$5:** There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure of total network usage.

6 **$H_0$6:** There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure computational power consumed.

Dividing the main hypothesis into these sub-hypotheses enables researchers to methodically investigate different facets of the Smart Fog protocol-based system and compare its efficiency and performance against a cloud-based counterpart. This methodical approach facilitates a thorough assessment of Fog Computing technology and its potential benefits in comparison to traditional cloud solutions within IoT environments.

3.6 Scope of Study

As studying about advances in new computational paradigm and the use of cloud computing and IoT have great futuristic applications. This emerging IoT introduces many challenges which cannot be handled by today's cloud computing. In this research work we deal with the IoT Environment features like low latency, high distribution, large-scale sensor network, and mobility support and device heterogeneity. This proposed SMART FOG system allows us to create a collaborative environment for IoT network. In the proposed system, we are going to implement SMART FOG protocol-based technique which will allow Fog nodes to share computing and storage power to IoT devices that have low computational power within IoT network. The proposed system will be able to schedule the tasks assigned to fog node for easy processing and efficient resource management. The proposed work is focused on creating a resilient environment using SMART FOG which will create trust in fog computing. As fog computing is in its infancy, there are still many open challenges are present. The SMART FOG will create trust between fog clients and fog environment by providing fault-tolerant and secure technique for fog computing. This research will identify some of these challenges and try to find a solution in the proposed system.

3.7 Research Methodology

The research methodology deals with the hypothesis which is the outcome of the objectives with the results. The proposed study attempts to implement SMART FOG, a collaborative approach using Fog Computing. The research involves quantitative and qualitative approaches. The SMART FOG collaboratively used computational power and storage of devices connected within Fog layer. The fog layer acts as an intermediate between IoT networks and Cloud centers. In this research, we have identified different open challenges in fog computing and tried to resolve some of them using SMART FOG. The hypothesis is nothing but a tentative statement to predict the expected outcomes of a study. Defining hypotheses helps in designing new experiments and observations. The following hypothesis is being tested for the system in the proposed research work. The hypothesis has arrived at the expected outcome of the system. "SMART FOG protocol-based technique to create Fog Computing environment will share computational power to IoT devices with low computational power".

3.7.1 Sources of Information

Information is gathered from journals, articles, and publications about fog computing are the main sources of information. The OpenFog consortium provided white papers which are very useful as the main source of information. The OpenFog community provided papers on the taxonomy of Fog Computing, basic architectural design, and structure of fog computing and their multiple layers of implementation. For further information, various articles from industrial experts who are connected to Fog Computing will act as a lighthouse in the dark.

3.7.2 Data Collection

The universe for the present study is comprised of smart city applications, all peoples within a smart city, and cloud centers. The universe also includes the devices within the IoT network, fog devices, etc. The smart city sample application is being randomly selected to implement SMART FOG and the overall performance and efficiency are being evaluated.

As per the SMART FOG applications, the IoT network generates some amount of data using sensors that are being used in computing and to test the performance of SMART FOG. The data generated in this research work is application-oriented. If we consider security surveillance applications for smart cities then the data is collected of images of events, video streams collected from cameras, and being used for further applications. The main input to the proposed system is requests from IoT devices for shared computational power.

Cloud -Fog Simulators for Data Collection

The primary goal of the research work is to examine and find the technologies associated with the SMART FOG project. To find new emerging technologies that can impact the cloud system in SMART FOG computing and also improve the reliability of Predictive models based on Artificial Intelligence and Machine Learning Algorithms is being developed to resolve computing problems.

iFogSim Simulators

Many available simulators can simulate the scenarios of cloud-fog computing environments such as EdgeCloudSim, MobIoTSim, SimpleIoTSimulator, IBM BlueMix, Google IoT Sim, and EmuFog. Most of the available simulators are

similar in their functionalities, programming language, or architecture. Therefore, we limited our study to only eight main simulators. The simulators are analyzed both from theoretical and practical perspectives. In theoretical comparisons, all eight simulators (iFogSim, iFogSim2, FogNetSim++, EdgeCloudSim, FogComputingSim, PureEdgeSim, YAFS, and LEAF) are compared based on their technical and non-technical characteristics, whereas for practical comparison use iFogSim, are in terms of their execution time, memory usage, and CPU consumption for simulating different applications under varying complexities.

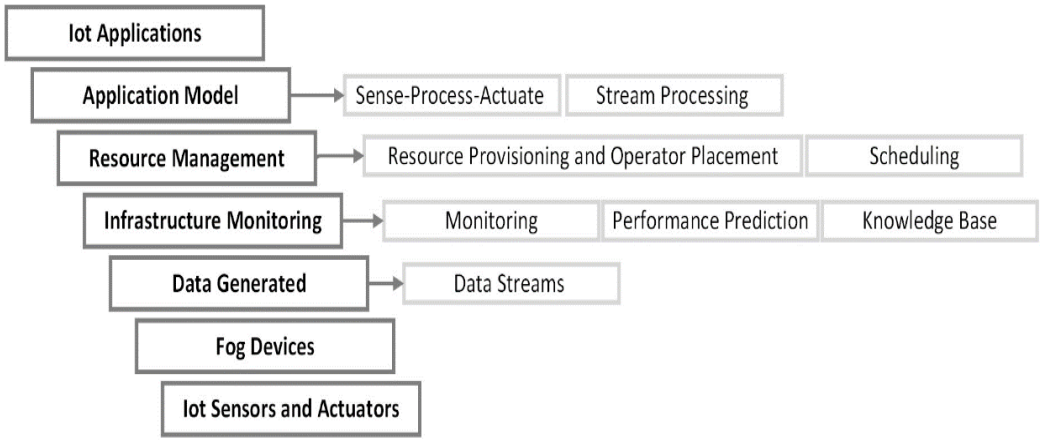

Figure 3.2: iFogSim Architecture (Muhammad, 2023)

The iFogSimToolkit provides a platform for modeling and simulation of resource management techniques in edge, Fog computing, and cloud environments. A newer version of iFogSim, adds distributed clustering, mobility, and microservices management as new features. Furthermore, it includes new example scenarios to validate and demonstrate their extension for the iFogSim. the architecture used by iFogSim is shown in Figure 3.2.

Due to the IoT revolution, almost everything is becoming a source for data generation. As a result, a tremendous amount of data is generated every second. Huge amount of data processed on workstations. typical data sources include mobiles, various types of sensors and actuators including thermostats, engines of airplanes, factories, mobiles, computers, automobiles such as driverless cars, metros, human health data, smart devices such as Google Home, Alexa Echo Dots, smart homes, smart shoes, watches, and, in general, all wearables, etc., and the number of items on the list increases all of the time. These data need to be pre-processed before

something useful can be derived from them because only some of the generated data are relevant or useful. This section will look at the various sources from which data is generated.

3.7.3 Through Participation in Conference and Paper Published

To collect insight into the subject and dive deeper into the details of fog computing, Cloud Computing, Artificial Intelligence, and Machine Learning algorithms following conferences were attended.

a) International Virtual Conference "Emerging Era of Applications of Computer: The Survey on Fog Computing and its Applications" on 15th -16th of January 2022 Organized by Pacific University Udaipur.

b) "Use of Clustering Machine Learning Algorithms in Fog Computing for Task Scheduling and Resource Allocation" has been published in European Chemical Bulletin (ISSN: 2063-5346), Volume 11, Issue 8, 2022 Date of Publication: - August 2022.

c) National Seminar on "Implementation of Academic Bank of Credit (ABC) in Higher Education Institutes" on 21st March 2023 Organized by Avinashilingam Institute for Home Science and Higher Education for Women University Udaipur.

d) IP Awareness Training Program under "National Intellectual Property Awareness Mission" Organized by Intellectual Property Office, India on 18, January 2023.

e) "A Comparative Study of Various Classification Machine Learning Algorithms in Fog Computing: Task Scheduling" has been published in Industrial Engineering Journal (ISSN 0970-2555), Volume: 52, Issue 5, No. UGC Care Approved, Group I, Peer Reviewed Journal 4, May: 2023.

I am grateful to all Conference Organizers and my fellow presenters and researchers who not only provided me with the platform to showcase my talent but also helped me with rich technical experience by actively participating in a conference to collect data. These gatherings have provided me the stage for scholarly exchange which helped me a lot in coming out with Machine learning solutions for traffic congestion problems.

3.7.4 Performance Evaluation

The performance of the developed machine learning predictive model is analyzed using various performance measures such as prediction accuracy, incorrectly classified instances, kappa score, and various confusion matrix parameters such as true positive rate, false positive rate, precision, recall, and F1-score. Compare the performance of the model with existing traffic prediction models and assess its effectiveness in predicting traffic congestion and optimizing transportation systems.

3.7.5 Machine Learning Predictive Model Development

Designed and developed a machine learning predictive model for smart fog systems using the gathered data and insights from the literature review. Utilize appropriate machine learning algorithms such as regression, Random Forest, Random Tree, Bayes net, naïve Bays, SMO, IBK, Logistic Regression, K-Star, and Multiclass classifier in addressing cloud issues.

3.8 Tools and Technique

A method known as SMART FOG uses nearby fog nodes to complete tasks to utilize cloud centers less frequently and with less delay. Therefore, to put this system into place, firstly build an IoT network and cloud application that can handle requests from the IoT network and store the data on servers. After that, an interface protocol is created to essentially connect the cloud and Internet of Things network and process requests that he can process rather than sending them to the cloud. Finally, the IoT network receives the results.

In the proposed system, the requesting IoT device can use the publish method to submit a request to the closest fog devices, and the nearest fog device that is available will accept the request and subscribe to share computing power. A few further security precautions are required and are being implemented in SMART FOG to safeguard the connection to prevent attacks like Man-In-the-Middle or identity theft.

Fog computing requires various tools, techniques, and algorithms to optimize data processing and management close to the network edge. Key components include Network Management Software-defined networking tools that optimize traffic routing and resource allocation. Virtualization technology tools like Docker deploy

applications in isolated environments efficiently Data Analytics Real-time analytics tools, such as Apache Kafka and Apache Storm, process data streams at the fog layer, Weka 3.8.6. Machine Learning Algorithms like decision trees, K-Nearest Neighbor, Logistic regression, K-Star, IBK, J48, Bagging, MLP clustering, and neural networks analyze and predict data trends locally, and Resource Management Algorithms include load balancing and task scheduling algorithms to optimize resource utilization and performance. These tools and techniques collectively enhance the efficiency, scalability, and security of fog computing systems.

To evaluate the performance and effectiveness of the smart fog systems, various metrics and statistical methods were employed. Percentage analysis, measures of central tendency, measures of dispersion, cumulative frequency, correlation coefficient, and regression analysis were used to analyze the collected mining data. Hypothesis testing was performed using the Chi-Square test. The study involved the simulation of data and model building, utilizing multiple regression analysis. A conceptual model based on regression was developed to examine the significance of different technologies. Additionally, the study aimed to assess the usefulness of loT, Artificial Intelligence, and Machine Learning-based models in addressing commuting problems. The Weka tool and Python were used for simulation and predictive analysis. Overall, the study employed a research design that combined qualitative and quantitative research approaches. The qualitative nature of the study facilitated the exploration of various concepts and ideas, leading to findings and recommendations for improving fog computation.

3.8.1 Weka Tools Techniques

The Weka Experimenter is a tool within the Weka software package that allows users to design, run, and analyze machine learning experiments systematically. It is particularly useful for comparing multiple machine learning algorithms and configurations on various datasets, helping researchers and practitioners make informed decisions about which algorithms work best for their specific tasks. Here’s a more detailed explanation of the Weka Experimenter’s key features and functionalities.

Experiment Design: The Experimenter allows users to design experiments by specifying different machine learning algorithms, datasets, and evaluation metrics. Users can choose from a wide range of classification, regression, and clustering. Algorithms available in Weka. They can also select multiple datasets to test the algorithms' performance across different data domains.

Parameter Sweeping: Users can explore the effect of different parameter settings on the performance of machine learning algorithms. The Experimenter enables parameter sweeping, where users can specify a range of values for certain parameters of the algorithms. The Experimenter then systematically runs experiments with different parameter combinations to find the optimal settings.

Cross-Validation and Evaluation Metrics: The Experimenter supports various techniques for evaluating machine learning models, including cross-validation (k-fold cross-validation, leave-one-out cross-validation, etc.). Users can select different evaluation metrics such as accuracy, precision, recall, F1-score, and others to assess the performance of the algorithms.

Batch Execution: The Experimenter can run experiments in batch mode, allowing users to schedule multiple experiments to run sequentially or concurrently. This feature is particularly useful for running large-scale experiments overnight or on computing clusters.

Result Analysis and Comparison: After the experiments are completed, the Experimenter provides detailed summary reports and visualizations of the results. Users can compare the performance of different algorithms on various datasets using statistical tests and visualizations like charts and graphs. This comparative analysis helps users identify the best-performing algorithms and configurations for their specific problem domains.

Reproducibility: The Experimenter ensures the reproducibility of experiments by allowing users to save the experiment configurations and results. Researchers can share these configurations and results with others, making it easier to validate and replicate experiments.

Integration with Other Weka Tools: The Experimenter seamlessly integrates with other Weka tools and interfaces, allowing users to utilize preprocessing techniques, attribute selection methods, and various machine learning algorithms available in Weka.

The Weka Experimenter provides a user-friendly environment for designing, and running. And analyzing machine learning experiments. Its capabilities make it a valuable tool for researchers and practitioners who want to systematically evaluate and compare different machine-learning algorithms and configurations on multiple datasets. Some key tools and techniques available in Weka Explorer include:

- Preprocessing Tools
- Classification Algorithms
- Clustering Algorithms
- Attribute Selection
- Evaluation Techniques
- Visualization Tools

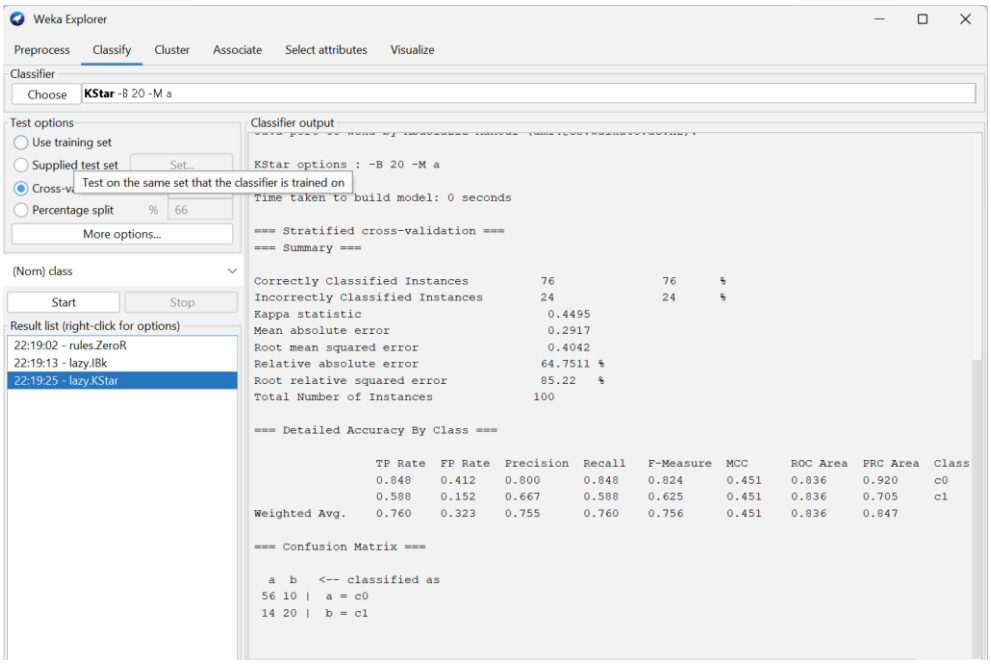

Figure 3.3: Weka Tool K-Star

Figure 3.3 shows the preprocessing tool in Weka applied to This interface includes details about the number of instances, number of attributes, relation, selected attribute tab, etc. In the present scenario, the details of the Time attribute are shown in the selected attributes tab.

3.8.2 Experimental Setup

CCTV applications process data recorded by cameras deployed at STL[58]. The source task, located at the STL, sends 10 Mbps of video data to a processing task that requires 30,000 MIPS and is responsible for traffic monitoring, enforcing traffic laws, and automatic incident detection. The 200 kbit/s of resulting data are sent to the sink task located in the cloud for further analysis and storage. 16 of these applications are running in our scenario, one for each STL.

Although the computational and network load required by the CCTV applications is constant during the entire simulation, the reported power consumption varies over time. This is because we allocate the static power consumption of fog nodes proportionally to applications running on them and fog nodes are utilized inefficiently in this experiment. Especially at night when only a few taxis are on the road, the relative power demanded by the CCTV applications rises.

3.8.3 Hypothesis Testing Tool

To test the famed null hypothesis, three types of statistical methods were used. The applied tests were the Pearson Chi-Square test, ANNOVA Test, and T-Test.

Chi-Square Test:

The Chi-Square test is a statistical method used to determine if there is a significant association between categorical variables. It compares observed frequencies with expected frequencies, assessing whether any differences are statistically significant.

The Formula for Chi Square Is

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

$c =$ degrees of freedom

$O =$ observed value(s)

$E =$ expected value(s)

[58] Seasonal and Trend decomposition using Loess

3.9 Applied Methodology

In this section, the proposed model, and the interaction among its components with the essential interfacing requirements are demonstrated. The proposed model consists of three layers concerning intelligent task offloading in fog cloud systems. It is composed of both fog and cloud servers. The underlying fog-cloud environment is comprised of distributed resources that are heterogeneous in terms of network hierarchy starting from the very basic physical layer of a network to the centralized cloud environment. Heterogeneous means these devices are dispersed at different geolocations and not stationary. The host servers, which perform as computing resources, intended for providing services to various application tasks, are enriched with a diverse set of resources. It is based on two types of applications, i.e., delay-sensitive applications and computation-intensive applications.

3.9.1 Fog-Cloud Smart Task Offloading Model

Mainly the architecture includes three layers with a smart task offloading management system which includes predictive and prescriptive constructs as shown in the figure below. The three layers included are the IoT or physical layer, Fog node layer, and the Cloud layer.

Offloading Management System is an intelligent framework designed to optimize task offloading decisions in distributed computing environments, particularly in fog computing and edge computing systems. The starting point in a task offloading procedure possess five main features:

1) Policy Repository regarding Offloading criteria
2) Recent status of fog snapshot
3) Receive, analyse, and offload the tasks
4) Prediction construct
5) Prescriptive construct

The entire process is activated by the Smart OMS. Its formation consists of a Policy Repository regarding Offloading criteria, monitoring & organizing offloading procedures, a recent snapshot of fog competence & readiness, a Prediction construct, and a Prescriptive construct.

3.9.2 Task Offloading

The volatile demand from IoT and mobile devices, which may not be predicted or anticipated immediately due to the unpredictability of the fog resources, the issue has to be handled. The main goal of this study is to provide a predictive and prescriptive approach for cost-effective task offloading and resource scheduling that will maximize the cost of these devices executing their applications. As a result, this research work cover addresses the mentioned issues and offers suggestions for how to fix them. Furthermore, some tasks share the same fog resources; as a result, there may be resource conflicts in some situations that could result in deadlock, some tasks experience delayed responses, and it's possible that new tasks won't be able to acquire resources at all, which is where latency comes into play. To ensure the equitable use of the underlying resources, it must be decided to improve fog performance by offloading some activities to adjacent nodes.

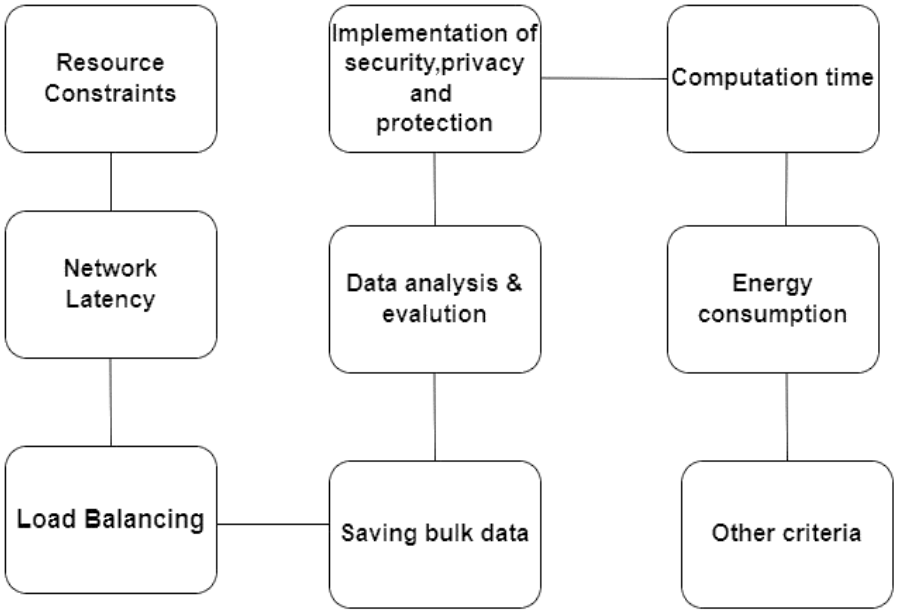

Figure 3.4: Task offloading Criteria (Satyakam, 2021)

Figure 3.4, shows Task offloading is a crucial process in fog computing and edge computing environments, where computational tasks are transferred from resource-constrained IoT devices to more capable fog nodes or cloud servers.

The decision-making process for task offloading considers various conditions, such as resource constraints, network latency, load balancing, security, privacy, data evaluation, storing bulk data, execution time, energy consumption, and other specific criteria. By balancing these factors, task offloading aims to optimize

resource utilization, reduce latency, improve energy efficiency, and enhance overall system performance. Offloading computationally intensive tasks to more powerful nodes, considering network conditions, and ensuring data privacy and security play key roles in achieving efficient and effective task offloading in distributed computing systems.

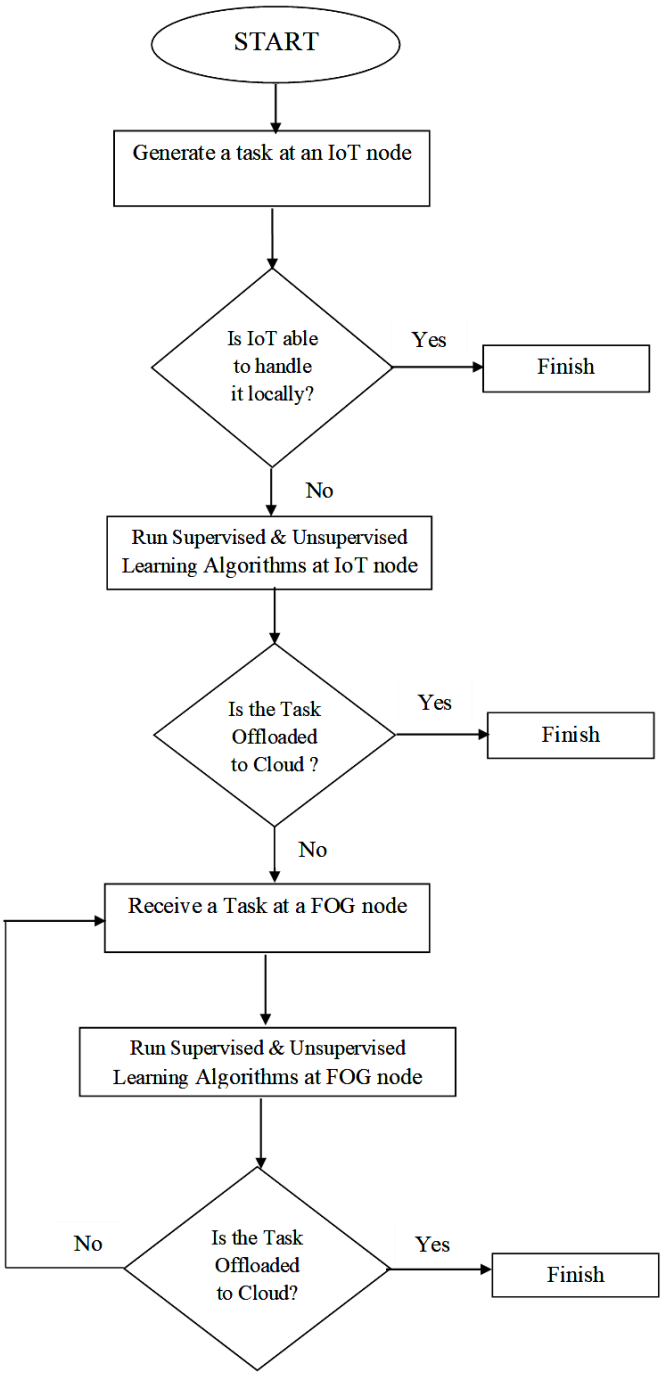

Figure 3.5: Flow Diagram: SMART FOG Task Offloading (Li, 2019)

Figure 3.5 shows the various steps used for the task offloading as shown above in the figure. In the proposed flowchart, a task is generated on an IoT node, and the node evaluates its capability to execute machine learning algorithms locally. If the IoT node lacks resources or the task complexity exceeds its capacity, the task is offloaded to the cloud. Alternatively, if the IoT node can handle the task, it executes the machine learning algorithms. The task is then sent to a fog node, which assesses its resources and executes machine-learning algorithms like classification and clustering. If the fog node is unable to handle the task, it may offload it back to the cloud. After the completion of tasks, results are delivered to the IoT node or end-user, based on application requirements and data privacy considerations. This dynamic process ensures optimal resource utilization, reduced latency, and enhanced performance in the IoT and fog computing environments.

3.9.3 Workflow Diagram

workflow diagram illustrating the progression of research and development in the field of SMART FOG. The workflow diagram illustrates the progression of research and development in the field of SMART FOG. It starts with understanding cloud, IoTs, and fog computing concepts and then delves into analyzing IoT-based architectures and protocols. Next, the focus shifts to evaluating various layers of communication protocols and devising improved task scheduling and allocation techniques for fog computing nodes.

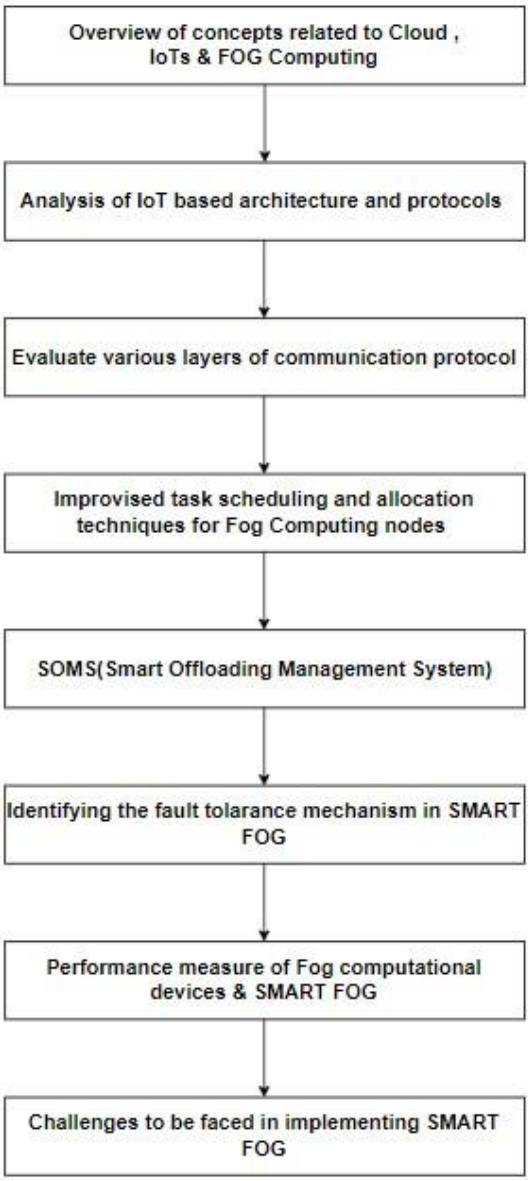

Figure 3.6: Workflow Diagram

Figure 3.6 shows depict the implementation and functioning of the Smart Offloading Management System, which optimizes task offloading decisions. It also highlights the identification and incorporation of fault tolerance mechanisms in SMART FOG to ensure system reliability. Furthermore, the diagram emphasizes the importance of measuring the performance of fog computational devices and the SMART FOG system. Finally, the challenges faced during the implementation of SMART FOG are outlined, underscoring the need to overcome hurdles to achieve successful deployment and operation.

3.10 Performance Metrics for Supervised and Unsupervised Algorithms

Performance metrics for supervised algorithms include accuracy, precision, recall, F1-score, and area under the ROC curve, which assess the model's prediction quality. For unsupervised algorithms, metrics like silhouette score, Davies-Bouldin index, and clustering accuracy evaluate the coherence and separation of clusters. These metrics help determine how well the algorithms perform their respective tasks in various data analysis scenarios.

According to Figure 3.7, various studies to find the quality and performance of the various clustering algorithms various measures are being suggested but finding one is a challenging task in unsupervised learning. Some of the major performance evaluation clustering methods or clustering validity indexes can be classified as external, internal, and relative as shown in the figure below.

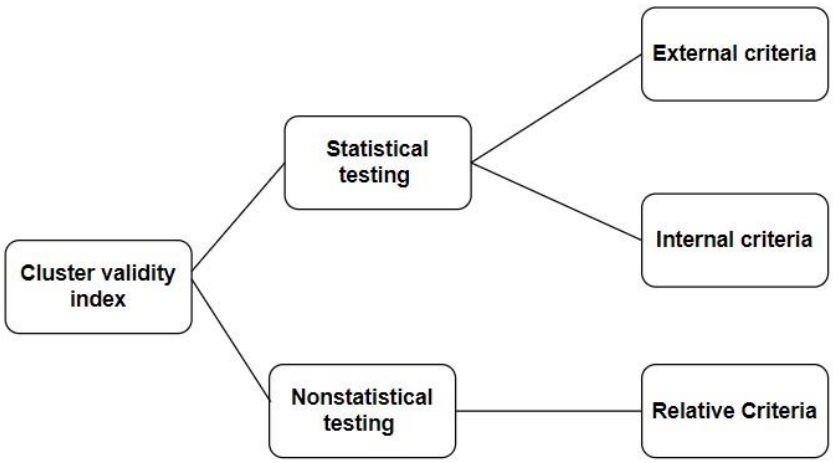

Figure 3.7: Cluster validity index (Wang, 2019)

3.10.1 Internal Validation

Figure 3.8 shows internal validation criteria are being used when we are not having additional information about the datasets. In such cases, the quality of the clustering algorithm can be measured by the two basic approaches partitioned and Hierarchical.

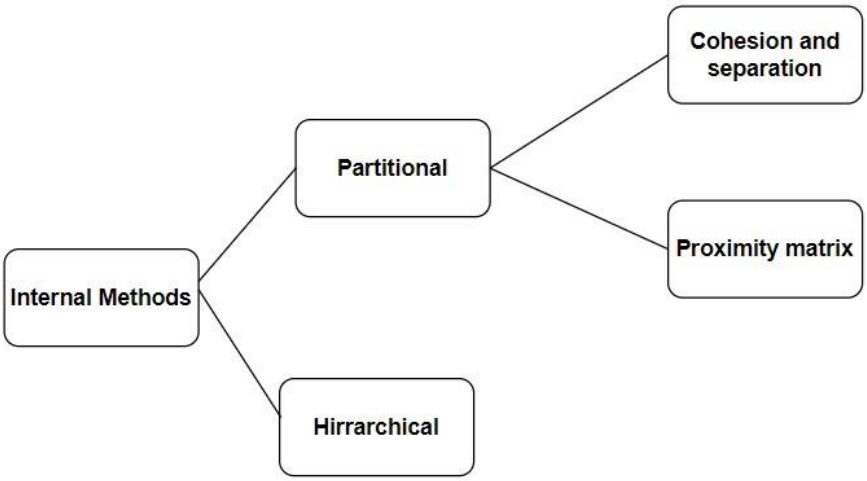

Figure 3.8: Internal Validation Method (Wang, 2019)

In situations where external information or ground truth labels are unavailable, internal validation criteria play a crucial role in assessing the quality of clustering algorithms. These methods evaluate clustering results based solely on the data's characteristics and clustering structure.

3.10.2 External Validation

External validation methods are considered with the supervised learning or classification problems. External validation methods can be also incorporated if additional information or class labels are available in the particular clustering problems that have the class labels for the training sets. For applying external validation various aspects are to be taken into consideration which are as follows

- Required to find clustering tendency for a particular dataset
- Find the correct number of clusters.
- Use internal methods for measuring the quality of clusters first.
- Now compare the internal method results with the external information.
- Make a comparison between the two sets of clusters to find the best one.

Figure 3.9 shows to find the clustering tendency for a given dataset, internal clustering validation methods are utilized to measure the quality of clusters without relying on external information. These methods, such as the Silhouette Score, Davies-Bouldin Index, and Dunn Index, help identify the correct number of clusters that yield the highest quality results. By comparing the internal validation outcomes with external information, obtained through external validation methods like

Adjusted Rand Index or Normalized Mutual Information when available, the clustering results can be evaluated against known class labels or ground truth data. The best clustering solution is determined by considering both internal and external validation results, aiming to achieve consistency and high-quality clusters.

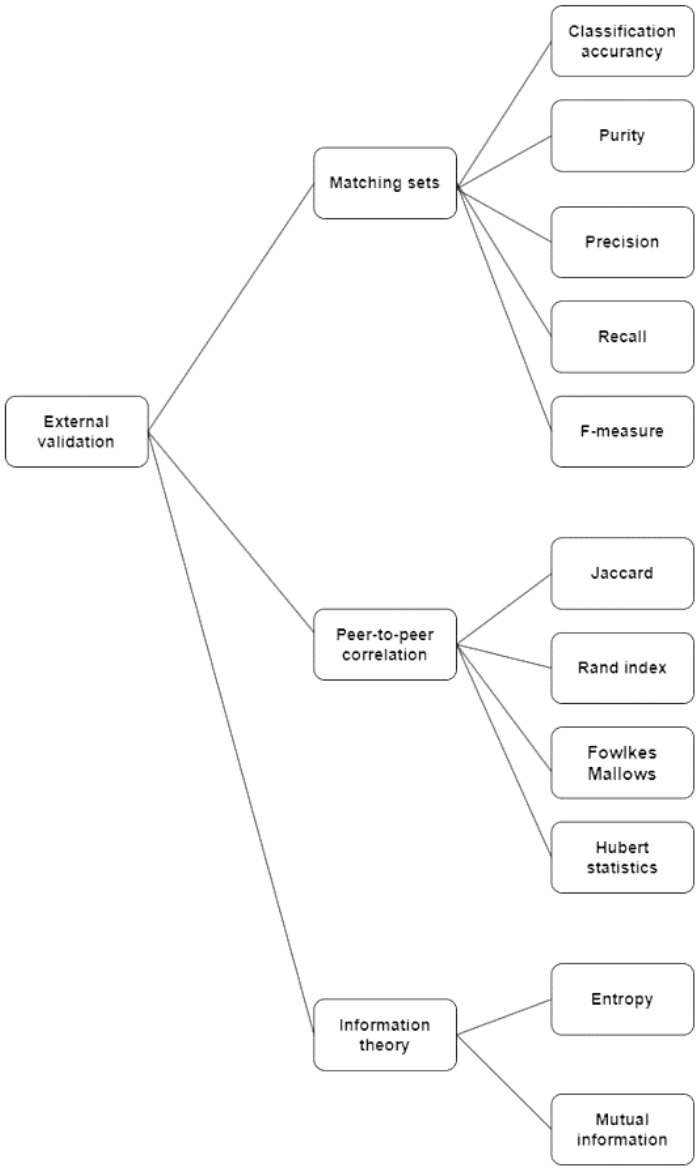

Figure 3.9: External Validation Method (Wang, 2019)

The external criteria are applied as in the clustering algorithm suppose C = {C1, C2…..Cm} represent the clustered partition and P = {P1, P2…. Ps} represent the true partition obtained from expert knowledge or class labels.

TP[59]: The no. of data points found in the same particular cluster, both C and P.

[59] True Positive

FP[60]: The no. of data points found in the same particular cluster in C but in a different cluster.

FN[61]: The no. of data points found in different clusters in C but in the same cluster in P.

TN[62]: The no. of data points found in different clusters, both in C and in P.

The no. of data points found in the same cluster in C:

m1 = TP + FP.

The no. of data points found in the same cluster in P:

m2 = TP + FN.

M = TP + FP + FN + TN.

These external validation methods help assess the accuracy, consistency, and robustness of clustering algorithms by comparing their results with known ground truth information or externally provided criteria. By utilizing these validation techniques, researchers and practitioners can make informed decisions about the suitability and performance of clustering algorithms for their specific applications and datasets.

Matching Sets

The first category in external criteria includes the measuring parameters like recall, precision, TP, TN, FP, FN, error, F- measure, etc. Precision can be calculated by number of the true positives

$$Pr = \frac{TP}{TP + FP} = \frac{TP}{P} = \frac{p_{ij}}{p_i}$$

Recall measures the percentage of data points properly included in the same particular cluster:

$$R = \frac{TP}{TP + FN} = \frac{p_{ij}}{p_j}$$

The F-measure is a combination of precision and recall

[60] False Positive
[61] False Negative
[62] True Negative

$$F_\alpha = \frac{1+\alpha}{\frac{1}{Pr} + \frac{\alpha}{R}}$$

The F-measure, also known as the F1 score, is a metric used to evaluate the accuracy of a classification model, particularly in binary classification tasks. It combines both precision and recall into a single measure, providing a balanced assessment of a model's performance.

Peer-to-peer Correlation

The second category includes the following methods: Peer-to-peer correlation refers to the degree of similarity or correlation between individuals or entities within a peer group or network.

Jaccard coefficient: The Jaccard coefficient is used to find the similarity of the identified clusters C to the true values in P

$$J = \frac{TP}{TP + FP + FN}$$
$$J = \frac{\sum_{ij} \binom{n_{ij}}{2}}{\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2} - \sum_{ij} \binom{n_{ij}}{2}}$$

The Rand coefficient is also similar to the Jaccard coefficient although used to measure considering the total data set (accuracy).

$$Rand = \frac{TP + TN}{M}$$
$$Rand = \frac{\binom{n}{2} - \sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2} - \sum_{ij} \binom{n_{ij}}{2}}{\binom{n}{2}}$$

The Folkes and Mallows coefficient also finds the similarity between the particular clusters generated by particular clustering algorithms as independent markers

$$FM = \sqrt{\frac{TP}{TP + FP} * \frac{TP}{TP + FN}}$$
$$FM == \frac{\sum_{ij} \binom{n_{ij}}{2}}{\sqrt{\sum_i \binom{n_i}{2} * \sum_j \binom{n_j}{2}}}$$

Hubert statistical coefficient

$$\Gamma = \frac{1}{M} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} Y_{ij}$$

Peer-to-peer Correlation includes Jaccard coefficient, Hubert statistical coefficient, Rand coefficient, and Folkes and Mallows coefficient which helps in finding the association between the entities.

Measures Based on Information Theory

Measures based on information theory assess the amount of information present in a system. Key metrics include entropy, reflecting dataset uncertainty, and mutual information, quantifying shared information between variables. Entropy can be considered as the reciprocal of the purity measure to find the degree of disorder among clusters:

$$H = -\sum_i p_i \left(\sum_j \frac{p_{ij}}{p_i} \log \frac{p_{ij}}{p_i} \right)$$

Mutual information is used to measure the reduction in uncertainty in clustering:

$$MI = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$$

These measures find utility across disciplines like machine learning and signal processing for optimizing systems and analyzing data. Mutual information quantifies the reduction in uncertainty about cluster assignments when clustering a dataset. By measuring the amount of shared information between data points and cluster labels, mutual information assesses how well clustering reduces uncertainty by revealing underlying patterns or structures in the data.

Optimization Metrics

Optimization metrics in fog computing are essential for measuring the efficiency and performance of distributed computing at the network edge. These metrics enable the assessment and improvement of resource utilization, latency reduction, and overall system optimization in fog-based architectures.

The goals of resource allocation, task scheduling, and workflow scheduling are to maximize the resources of fog nodes by optimizing the job execution process. Resource allocation, task scheduling, and workflow scheduling in fog computing aim to maximize fog node resources by optimizing job execution. Key objectives include efficient task allocation, minimizing delays, and improving overall system performance

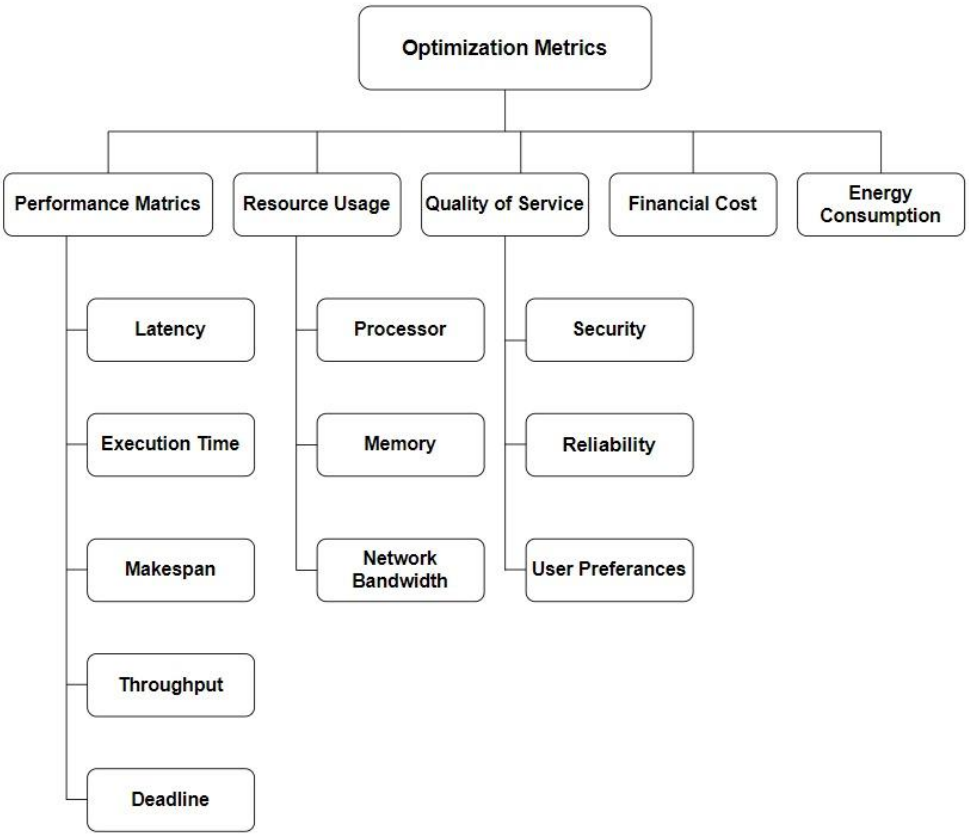

Figure 3.10: Optimization Metrics (Marbukh, 2019)

Figure 3.10 shows optimization metrics such as makespan, latency, throughput, energy consumption, load balancing, and quality of service play a crucial role in achieving these goals. By considering these metrics and employing appropriate algorithms and techniques, fog computing systems can enhance resource utilization, reduce delays, and provide efficient and reliable execution of tasks and workflows.

Optimization metrics in fog computing are critical for achieving optimal performance, resource management, and latency reduction at the network edge. They guide decision-making, ensuring that fog architectures deliver on their promise of efficient and responsive edge computing solutions.

Performance Metrics

Performance metrics in fog computing are vital for evaluating the efficiency and effectiveness of edge computing systems. These metrics provide valuable insights into processing speed, resource utilization, and data transfer rates, enabling the fine-tuning and improvement of fog-based architectures.

Performance metrics in task scheduling for fog computing refer to the parameters used to evaluate and measure the effectiveness and efficiency of the task scheduling process. These metrics provide insights into the performance of the system and help in assessing the quality of the scheduling algorithm or approach. Common performance metrics include makes pan (total time taken to complete all tasks), latency (response time between task submission and completion), throughput (number of tasks completed per unit of time), resource utilization (percentage of resources utilized), and fairness (equitable distribution of resources and workload among fog nodes). These metrics allow for quantitative assessment and comparison of different task scheduling techniques, enabling the selection of optimal approaches for improved performance in fog computing environments. The parameters of performance metrics are as follows:

Latency

One of the most crucial variables for evaluating the effectiveness of any task-scheduling system is latency. Other names for latency include delay and reaction time. The sum of the transmission delay and the computing latency is the total latency.

$$Latency_i = TL_i + CL_i$$

- $Latencyi$: latency,
- TLi: transmission latency
- Computational latency of task ‘i’.

Execution Time

Execution time is the length of time it takes for a system to complete a task. CPU or execution time does not account for the time spent waiting for I/O or other operations to complete.

$$ExeTime_i = FT_i + ST_i$$

- $ExeTimei$: overall execution time,
- FTi: finish time
- STi: the start execution time of task ‘i’.

Make Span

Make span is a key goal of task scheduling that shows how long it will take to execute a process in its entirety.

$$Makespan = CT_l - ST_f$$

- CTl: time when the last task is completed
- STf: starting time of the first task.

Throughput

The number of tasks finished in a system's throughput is measured in units of time.

$$Throughput = \frac{\text{Number of tasks}}{\text{Makespan}}$$

Throughput, in the context of computing and systems, refers to the rate at which tasks or operations are completed or processed within a given time frame. It is a performance metric that measures the efficiency and productivity of a system, indicating how many tasks or units of work are accomplished in a specific period. The measurement of throughput is typically expressed in terms of tasks per second, operations per second, or any other appropriate unit of time. Higher throughput indicates that the system can handle a greater volume of work and is more capable of processing tasks efficiently.

Deadline

The deadline is the amount of time between when a task is submitted and when it must be finished. The completion of each activity at the designated deadline is crucial in real-time applications. Missing a task deadline may be disastrous, especially for difficult real-time applications like air traffic control Jamil (2022). Meeting task deadlines is critical in real-time applications as it ensures timely processing and response. In fog computing, where tasks are distributed across fog nodes, the deadline parameter becomes crucial for efficient task scheduling. The deadline specifies the maximum acceptable delay for task completion, and scheduling algorithms must consider this constraint to ensure tasks are allocated to appropriate nodes that can meet the specified deadlines. Failure to meet task deadlines in time-sensitive applications can have severe consequences, such as compromising safety, system failures, or financial losses. Therefore, in fog computing environments, effective task scheduling algorithms are designed to

prioritize tasks based on their deadlines, optimizing resource allocation and ensuring timely task completion within the given constraints.

Performance metrics in fog computing are instrumental in optimizing edge computing solutions. They facilitate informed decision-making, resource allocation, and system enhancements, ensuring that fog architectures deliver the required speed, efficiency, and responsiveness in the evolving digital landscape.

3.10.3 Simulation Setup

For the predictive construct, both supervised and unsupervised machine learning algorithms were used. A simulated environmental setup was constructed to evaluate and appraise the research model being proposed which is SMART FOG system which included an improvised task offloading approach. The experimental environment used is Anaconda Jupiter Python and R. The model is trained through various supervised and unsupervised learning algorithms which include KNN[63], Decision Tree, MLP[64], Logistic Regression algorithm, Lazy IBK, Naive Bayes, SVM[65] being used.

This research work aims to address various aspects of smart fog systems It encompasses a mixed methods approach, combining quantitative analysis and qualitative insights. The research objectives include studying commuting and cloud issues, analyzing technologies for enhancing computation, addressing implementation issues, and identifying the most appropriate machine learning approach for fog computing. The research begins with a comprehensive literature review to identify gaps in existing research and frameworks. Its performance is evaluated using various performance measures, comparing it with existing models. Recommendations and strategies are proposed to overcome these challenges.

[63]K-Nearest Neighbor
[64]Multilayer Perceptron
[65]Support Vector Machine

Chapter- 4
Smart Fog Protocol-Based Techniques

Across the globe, billions of devices are today communicating and exchanging data with each other. IoT communication protocols protect and ensure the security of the data being exchanged between these devices. This research work covers the most popular protocols in use today.

4.1 Analyzing IoT Infrastructure for Smart Fog Protocol Design

The Smart Fog Protocol-Based Technique involves the utilization of intelligent fog nodes at the edge of the network to process data locally and reduce the burden on centralized cloud servers. To design this technique, various layers of communication protocols are essential. The physical layer focuses on selecting appropriate communication technologies for IoT devices, while the data link layer ensures reliable data transmission. The network layer facilitates efficient communication paths between fog nodes and IoT devices, while the transport layer ensures orderly data delivery and minimizes latency. At the top layer, the application layer enables seamless integration with fog-based services and applications, optimizing real-time data processing at the edge and offering advantages in latency reduction, bandwidth optimization, and scalability for IoT applications.

4.1.1 Message Queue Telemetry Transport Protocol

This publish/subscribe message transport protocol is open source and highly lightweight, making it a great choice for connecting tiny devices to restricted networks. It was developed to function in environments with low bandwidth, such as sensors and mobile devices, and on networks that are not completely stable. Because of this feature, it is the protocol of choice for connecting devices that have a tiny code footprint. It is also the protocol of choice for wireless networks that have different amounts of delay as a result of bandwidth limits or unstable connections. It accomplishes this by operating on top of TCP/IP[66], which is the foundation of the Internet. MQTT is comprised of these three primary parts: Subscriber. Publisher and Broker in this protocol's most fundamental process, the publisher is responsible for creating and sending information to subscribers via a broker. This information is then received by the subscribers. The authorization of subscribers and publishers is checked by the broker as part of the broker's primary responsibility, which is to

[66]Transmission Control Protocol and Internet Protocol

maintain data security. This protocol is favored for usage in IoT devices because it can deliver well-organized information routing features to low-bandwidth networks as well as tiny, low-cost, low-memory, and power devices. MQTT employs three different degrees of quality of service to assure the dependability of its messages.

MQTT is a communication protocol that can go in both directions, meaning that clients may both generate and receive data through the process of publishing messages and subscribing to topics. IoT devices that are equipped with connectivity in both directions can concurrently deliver sensor data and receive configuration information and control commands. MQTT makes it considerably simpler to validate clients using contemporary authentication methods and encrypt communications using TLS[67].

CoAP message flows involve lightweight and efficient communication between IoT devices and servers. CoAP messages include GET, PUT, POST, and DELETE methods, enabling device data exchange. Devices send requests to servers, which respond with corresponding acknowledgments or data. CoAP's simplicity and low overhead make it ideal for resource-constrained IoT devices.

4.1.2 Constrained Application Protocol

CoAP is a Web transfer protocol designed for use in the IoT with restricted devices and networks. It is meant for applications that have a limited capacity to connect utilizing LWM2M[68], such as smart energy and building automation, and it may be implemented through a UDP[69].LWM2M makes it possible to remotely manage IoT devices and provides interfaces for safely monitoring and controlling those devices. The design of CoAP is based on the well-known REST[70] paradigm. According to this model, servers make resources available under a URL[71], and clients may access these resources by utilizing methods such as GET, PUT, POST, and DELETE. Both the CoAP and HTTP protocols have many similarities; however, the CoAP protocol has been improved for the IoT, and more especially for machine-to-machine communication. It has a minimal overhead, combined with the ability to proxy and

[67]Transport Layer Security
[68]Light-Weight Machine-To-Machine Communication
[69]User Datagram Protocol
[70]Representational State Transfer
[71]Uniform Resource Locator

cache messages, and it asynchronously exchanges messages. The architecture of CoAP is broken down into two primary categories: messaging, which is in charge of the dependability and duplication of messages, and request/response, which is in charge of communication between clients and servers.

The message layer sits above UDP and is in charge of the communication protocol that enables IoT devices and the internet to exchange messages with one another. Confirmable messages, non-confirmable messages, acknowledgment messages, and reset messages are the four distinct varieties of CoAP communications. When two endpoints communicate with one another, a CON[72]is a message that can be relied upon. It is repeated until the receiving end sends an acknowledgment message, at which point it is stopped. The message ID of an ACK[73] the message is identical to the message ID of a CON message. If the server is unable to successfully manage the incoming request, it may respond with a RST rather than an ACK. Unreliable NON[74] messages, in which the server does not acknowledge the message, can be used for transferring messages that are not vital to the operation of the system. To avoid sending duplicate messages, NON-messages are given unique message identifiers.

The Request/Response layer is the second tier of the CoAP abstraction layer. Requests can be sent using either CON or NON-messages in this layer. In situations in which a server can instantly react to a request, the request is communicated using a CON message, followed by an ACK message that contains the answer or the error code that was generated by the server. The message ID is not included in either the request or the response's token, which means they have their unique token. When the server is in a position where it is unable to instantly react, it will send an ACK message that has no content as the response. After the response is complete, a new CON message that includes the response is sent back to the client. The client then acknowledges the response that it has received in this new CON message.

[72]Confirmable Message
[73]Acknowledgement
[74]Non-confirmable

4.1.3 Advanced Message Queuing Protocol

Figure 4.1, shows AMQP is an open standard application layer protocol that was developed with the goals of providing increased security and dependability while still being easy to deploy and interoperable. Because TCP is employed as a transport protocol, it is a connection-oriented protocol. This means that to transmit data, both the client and the broker need to first establish a connection with one another. AMQP provides two levels of quality of service for the dependable delivery of messages: the unsettle format, which is comparable to MQTT's QoS0, and the settle format. The primary distinction between AMQP and MQTT standards is that AMQP brokers are composed of two primary parts: exchange and queues. MQTT brokers only have one primary part. Exchange is in charge of both receiving messages from publishers and delivering them to the appropriate queues. Subscribers establish connections to the queues, which in essence stand in for the topics, and begin receiving sensory input as soon as it becomes available.

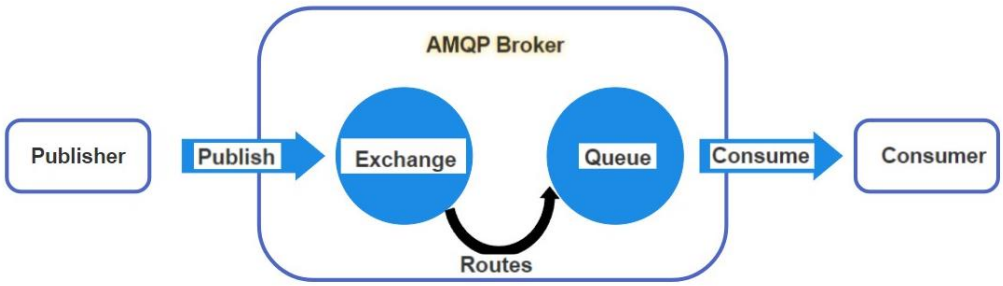

Figure 4.1: AMQP Architecture (Macarulla, 2016)

AMQP architecture is a messaging protocol designed for reliable and efficient message communication between distributed systems. It employs a client-server model with message brokers as intermediaries. Producers send messages to the broker, which then delivers them to appropriate consumers based on routing rules and message queues.

4.1.4 Data Distribution Service

DDS is a middleware protocol for data-centric connection that was developed by the object management group. It offers commercial and mission-critical IoT applications low-latency data communication, exceptional dependability, and a scalable design. This protocol enables the use of multicasting techniques during data transmission and enables high-quality QoS to be provided by applications and devices with a tiny

memory footprint. Both a DCPS[75] layer and a data-local reconstruction layer make up DDS's communications paradigm. These levels are referred to as the interface layers.

Throughout the publish/subscribe process, the DCPS layer is the one that is in charge of binding the values of data objects included inside an application. At the application level, the DLRL[76] is a layer that is used for integrating DDS, but its use is optional.

4.2 Task Scheduling and Allocation in Smart Fog Computing Nodes

The previous survey results were analyzed and a detailed treatment of the fundamentals of scheduling and scheduling types, such as task scheduling, workflow scheduling, resource allocation, and the many optimization measures used to evaluate these methods. Classification and extensive assessment of existing scheduling algorithms, with a special emphasis on intelligent dynamic scheduling strategies based on machine learning, fuzzy logic, reinforcement learning, and deep reinforcement learning, with descriptions of their strengths and shortcomings. Identification of research gaps and problems for task scheduling and resource allocation in fog computing for future research efforts in this subject through the presentation of various simulation settings and tools utilized in diverse studies.

[75]Data-Centric Publish-Subscribe
[76]Data Local Reconstruction Layer

Comparison of Traditional Scheduling Algorithms

Table 4.1, shows compares several traditional scheduling algorithms based on their type and the specific performance measures they optimize.

Table 4.1: Comparison of Traditional Scheduling Algorithms

Name	Type of Scheduling	Optimize the following Performance Measure				
		Latency	Execution Time	Network Usage	Energy Consumption	Cost
FCFS[77]	Task Scheduling	Optimized	Unoptimized	Optimized	Optimized	Unoptimized
PERA[78]	Task Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized
WRR[79]	Task Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Unoptimized
FCFS	Resource Scheduling	Optimized	Unoptimized	Optimized	Optimized	Unoptimized

FCFS task scheduling algorithm when applied in Fog and cloud environment suggests that FCFS in Fog environment optimizes latency, total network usage, and energy consumption when compared with FCFS in cloud environment. PERA, a Priority-based task scheduling algorithm optimizes latency and cost.

FCFS is a task-scheduling algorithm that optimizes latency, network usage, and energy consumption but does not focus on minimizing execution time or cost efficiency. Priority-based Scheduling also deals with task scheduling, optimizing latency and cost, while neglecting execution time, network usage, and energy consumption. Weighted Round Robin, another task scheduling algorithm, primarily optimizes latency without targeting execution time, network usage, energy consumption, or cost. The combined FCFS, Delay Priority, and Concurrent approach, a resource scheduling method, optimizes latency, network usage, and energy consumption but does not focus on execution time or cost efficiency. Each algorithm is tailored to enhance specific aspects of performance, demonstrating the trade-offs inherent in scheduling decisions

[77]First-Come, First-Served
[78]Packetized Ensemble Resource Allocation
[79]Weighted Round Robin

Integer Linear Programming

Table 4.2 compares various Integer Linear Programming scheduling algorithms based on their type and the performance measures they optimize.

Table 4.2: Integer Linear Programming

Name	Type of Scheduling	Optimize the following Performance Measure			
		Latency	Makespan	QoS	Cost
ILP[80]	Resource Scheduling	Optimized	Unoptimized	Optimized	Optimized
MILP[81]	Resource Scheduling	Unoptimized	Unoptimized	Optimized	Optimized

ILP, a resource scheduling algorithm based on integer linear programming when used in Fog environment optimizes the latency, QoS, and cost when compared with a cloud environment. ILP is a resource scheduling algorithm that optimizes latency, Quality of Service, and cost but does not focus on minimizing makespan. Min-CCV and Min-V, also resource scheduling algorithms, prioritize QoS and cost efficiency, without optimizing latency or makespan.

Comparison of Heuristic Scheduling Algorithms

Table 4.3, shows compare various heuristic scheduling algorithms based on their type and optimized performance measures.

Table 4.3: Comparison of Heuristic Scheduling Algorithms

Name	Type of Scheduling	Optimize the following Performance Measure					
		Latency	Makespan	QoS	Cost	Energy Consumption	Network Usage
SJF[82]	Task	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized	Unoptimized
PTPN[83]	Resource	Unoptimized	Unoptimized	Optimized	Unoptimized	Optimized	Unoptimized
MCCV [84]	Resource	Unoptimized	Unoptimized	Optimized	Optimized	Unoptimized	Unoptimized
EDF &LFC[85]	Resource	Optimized	Optimized	Unoptimized	Optimized	Unoptimized	Unoptimized
DOTS[86]	Resource	Optimized	Unoptimized	Unoptimized	Optimized	Unoptimized	Unoptimized
TIPS[87]	Task / Resource	Unoptimized	Unoptimized	Optimized	Unoptimized	Unoptimized	Unoptimized

[80]Integer Linear Programming
[81]Mixed Integer Linear Programming
[82]Shortest Job First
[83]Preemptive Task Priority Network
[84]Minimum Critical-Cycle Variance
[85]Earliest Deadline First and Least Slack Time
[86]Dynamic Optimization of Time Sequences
[87]Time-Invariant Power Scheduling

SJF task scheduling algorithm optimizes latency and energy consumption. Similarly, PTPN resource allocation algorithm highly optimizes QoS and energy consumption.

SJFfor task scheduling optimizes latency and energy consumption but not makespan, QoS, cost, or network usage. PTPN for resource scheduling focuses on optimizing QoS and energy consumption, neglecting other factors.

Min-CCV and Min-V for resource scheduling enhance QoS and cost efficiency but do not optimize latency, makespan, energy consumption, or network usage. EDF & Static LFC for resource scheduling optimize latency, makespan, and cost, leaving QoS, energy consumption, and network usage unoptimized. DOTS for resource scheduling focuses on minimizing latency and cost but does not optimize makespan, QoS, energy consumption, or network usage. Finally, TIPS for both task and resource scheduling prioritizes QoS without addressing latency, makespan, cost, energy consumption, or network usage.

Comparison of Fuzzy-Based Scheduling Algorithms

Table 4.4, shows a Comparison of Fuzzy-Based Scheduling Algorithms Fog computing is not a replacement for cloud computing but instead, an extension of cloud computing that enhances the already established cloud architecture. Here's how While the server nodes of cloud computing are located within the internet, fog computing has them at the edge of the networks. With this parameter, fog computing enhances cloud computing by functionally managing data from mobile devices thus reducing latency and improved response time.

Table 4.4: Comparison of Fuzzy-Based Scheduling Algorithms

Name	Type of Scheduling	Optimize the following Performance Measure					
		Latency	Makespan	QoS	Cost	Energy Consumption	Network Usage
RFN[88]	Resource Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized	Unoptimized
FLPSO[89]	Resource Scheduling	Unoptimized	Unoptimized	Optimized	Unoptimized	Unoptimized	Unoptimized
FPFTS[90]	Resource Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized	Optimized
EDA[91]	Resource Scheduling	Optimized	Unoptimized	Optimized	Unoptimized	Optimized	Optimized

[88] Rule-based Fuzzy Network
[89] Fuzzy Logic and Particle Swarm Optimization
[90] Fuzzy-Possibilistic Fuzzy Time Series
[91] Estimation of Distribution Algorithm

RFN, a fuzzy-based scheduling algorithm is a resource scheduling algorithm that optimizes latency and energy consumption. Similarly, FLPSO algorithm highly optimizes QoS.

Fog computing is not a replacement for cloud computing but instead, an extension of cloud computing that enhances the already established cloud architecture. Here's how – While the server nodes of cloud computing are located within the internet, fog computing has them at the edge of the networks. With this parameter, fog computing enhances cloud computing by functionally managing data from mobile devices thus reducing latency and improved response time.

4.3 Challenges in Implementing Fog Computing

Implementing Fog computing faces challenges such as heterogeneous device integration, security concerns at the edge, resource optimization, reliability maintenance, and scalability issues. Ensuring seamless interoperability between diverse devices, managing security risks at the edge, and optimizing resource allocation are crucial tasks. Additionally, maintaining reliability in a decentralized environment and addressing scalability concerns pose significant challenges that require comprehensive solutions. Fog computing is really necessary. There are, however, many obstacles to overcome to put it into practice:

Data Privacy

By placing fog nodes in the network's periphery, fog computing makes them available to a wider audience of end users. This makes the fog nodes more of a target for cyber-attacks as they collect a greater volume of sensitive data than the distant cloud.

Security

As fog computing requires authentication of devices at several gateways, the possibility of a rogue user using a spoofed IP address to access the data stored in a specific fog node is the most crucial security concern. This resulted in the installation of intrusion detection systems throughout the whole platform.

Network Management

Because they are linked to disparate hardware types, managing the fog's nodes, network, and inter-node connections can be arduous without software-defined networking and network function virtualization approaches.

Positioning the FOG Servers

Positioning Fog servers, or Fog nodes, is essential in Fog Computing architecture to enhance performance by bringing data processing closer to the data sources. This proximity reduces latency, conserves bandwidth, and improves network efficiency, particularly for real-time applications like autonomous vehicles, industrial automation, and smart grids. Effective placement involves a distributed and hierarchical network topology to balance load and prevent bottlenecks, considering workload characteristics and dynamically adjusting based on network conditions. It also requires modular and scalable deployment to accommodate varying demands, ensuring high availability through redundancy. Security is paramount, with robust measures to protect sensitive data and compliance with local regulations. Additionally, energy efficiency is critical, achieved through strategic placement with reliable power sources and green computing practices. Practical scenarios include smart cities for traffic management and public safety, industrial IoT for predictive maintenance and automation control, healthcare for remote monitoring and telemedicine, and retail for in-store analytics and reliable point-of-sale systems. To maximize the service provided by fog computing and reduce maintenance costs, it is necessary to analyze the work performed in each node of the servers before deciding where to arrange the group of fog servers.

Positioning Fog servers effectively is a multifaceted challenge that requires careful consideration of proximity to data sources, network topology, workload distribution, scalability, security, and energy efficiency. By strategically placing these nodes, organizations can leverage the benefits of Fog computing, such as reduced latency, improved bandwidth utilization, enhanced data security, and greater overall network efficiency. This approach is particularly beneficial in applications requiring real-time processing and analysis, making it a vital component of modern distributed computing architectures.

Energy consumption is a critical consideration in Fog Computing systems due to the extensive deployment of fog nodes across distributed environments. These fog nodes, which are responsible for processing and managing data at the edge of the network, often operate in resource-constrained settings with limited power sources. Energy consumption is significant because of the large number of fog nodes used in fog computing systems. Our research work focuses on the above-stated objective which aims to use the computational power of computation-enabled devices to collaboratively perform tasks and speed up the processing.

4.4 Hypothesis Testing Results

The null hypothesis $H_0$1 as stated Smart Fog protocol-based technique to create a Fog Computing environment will not share computational power with IoT devices with low computational power and other aspects are being categorized into various sub-hypotheses $H_0$1, $H_0$2, $H_0$3, $H_0$4, $H_0$5, and $H_0$6 to compare the impact of various aspects related to efficiency and various measures of SMART FOG protocol-based system with the cloud-based system.

$H_0$1: There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure execution time.

An alternative hypothesis is as follows

H_a1: There is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure execution time.

When comparing Fog Computing to Cloud Computing in terms of average execution time, it's essential to consider how each architecture processes tasks and their implications for task completion speed. Fog Computing, which processes tasks closer to the edge of the network, can potentially reduce latency and speed up execution times, particularly for time-sensitive tasks, by minimizing the distance data needs to travel. However, the effectiveness of Fog Computing depends on factors such as task complexity, resource availability at the edge, and network efficiency. Cloud Computing, while offering scalability and computational power, may introduce latency due to the distance between edge devices and centralized data centers, impacting average execution time. The choice between Fog Computing and

Cloud Computing should be based on the specific requirements of the application, considering factors such as task type, network latency, resource availability, and scalability needs.

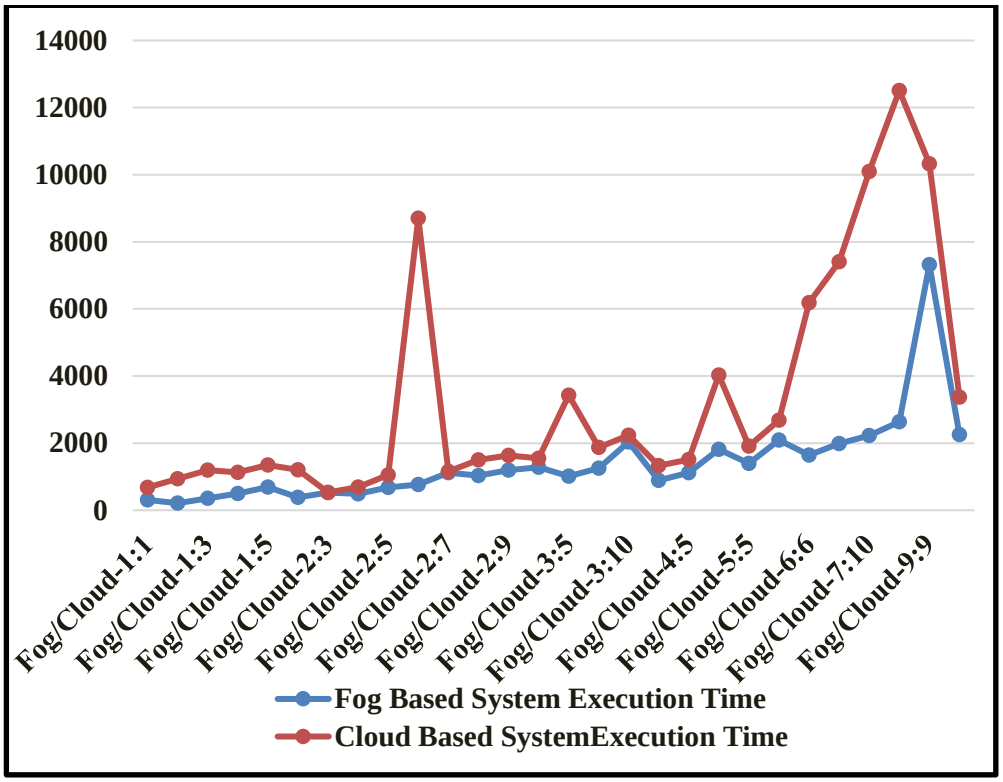

Figure 4.2: Fog Vs Cloud System Based on Average Execution Time

Figure 4.2, shows comparative analysis between Fog fog-based systems and Cloud cloud-based systems based on reduction in execution time as shown below confirms that there is a large reduction in execution time with the use of Smart Fog-based systems as compared to Cloud-based systems.

Table 4.5: Execution Time Reduced due to Fog Computing Environment

System	Fog Based System Execution Time	Cloud Based System Execution Time	Execution Time Reduced Using Fog System
Fog/Cloud-1:1	312	684	372
Fog/Cloud-1:2	210	933	723
Fog/Cloud-1:3	359	1198	839
Fog/Cloud-1:4	502	1133	631
Fog/Cloud-1:5	692	1348	656
Fog/Cloud-2:2	384	1203	819
Fog/Cloud-2:3	525	531	6
Fog/Cloud-2:4	494	690	196
Fog/Cloud-2:5	677	1048	371
Fog/Cloud-2:6	769	8703	7934
Fog/Cloud-2:7	1122	1153	31
Fog/Cloud-2:8	1032	1502	470
Fog/Cloud-2:9	1193	1632	439
Fog/Cloud-2:10	1278	1547	269
Fog/Cloud-3:5	1010	3429	2419
Fog/Cloud-3:6	1253	1877	624
Fog/Cloud-3:10	2036	2237	201
Fog/Cloud-4:4	893	1328	435
Fog/Cloud-4:5	1121	1513	392
Fog/Cloud-4:10	1816	4024	2208
Fog/Cloud-5:5	1400	1908	508
Fog/Cloud-5:10	2091	2686	595
Fog/Cloud-6:6	1648	6181	4533
Fog/Cloud-6:10	1986	7403	5417
Fog/Cloud-7:10	2229	10095	7866
Fog/Cloud-8:10	2636	12508	9872
Fog/Cloud-9:9	7315	10323	3008
Fog/Cloud-10:5	2254	3373	1119

Table 4.5 shows Execution Time Reduced due to Fog Computing Environment in Fog system 8:10, 9:9, 7:10, 6:10, 6:6, 4:10, and 2:6 there is a large reduction in execution time with values 9872, 3008, 7866, 5417, 4533, 4024, and 8703 respectively. So, it is very clear that Fog layer plays an important role in the execution time reduction. The Smart Fog system 9:9 which means 9 areas and 9 cameras takes a lower execution time of 7315 as compared to the cloud system 9:9 with an execution time of 10323. The experimental outcomes are further represented or categorized into high and low as shown below in the crosstabulation table.

Table 4.6 shows the FOG SYSTEM operates over two distinct execution time ranges, categorized into "Low" and "High." The "Low" range includes values from 0 to 500, while the "High" range covers values from 501 to 10000. Similarly, the

CLOUD SYSTEM is categorized into "Low" and "High" ranges, with the "Low" range spanning from 0 to 250 and the "High" range covering 251 to 10000.

Table 4.6: Classification of Fog and Cloud for Execution Time

Obs	Fog System	Execution Time	Rank	Cloud System	Execution Time	Rank
1	Fog-1:1	312	Low	Cloud-1:1	684	High
2	Fog-1:2	210	High	Cloud-1:2	933	High
3	Fog-1:3	359	High	Cloud-1:3	1198	High
4	Fog-1:4	502	High	Cloud-1:4	1133	High
5	Fog-1:5	692	High	Cloud-1:5	1348	High
6	Fog-2:2	384	High	Cloud-2:2	1203	High
7	Fog-2:3	525	Low	Cloud-2:3	531	High
8	Fog-2:4	494	High	Cloud-2:4	690	High
9	Fog-2:5	677	Low	Cloud-2:5	1048	Low
10	Fog-2:6	769	Low	Cloud-2:6	8703	Low
11	Fog-2:7	1122	Low	Cloud-2:7	1153	High
12	Fog-2:8	1032	High	Cloud-2:8	1502	High
13	Fog-2:9	1193	Low	Cloud-2:9	1632	Low
14	Fog-2:10	1278	Low	Cloud-2:10	1547	High
15	Fog-3:5	1010	Low	Cloud-3:5	3429	High
16	Fog-3:6	1253	Low	Cloud-3:6	1877	Low
17	Fog-3:10	2036	High	Cloud-3:10	2237	High
18	Fog-4:4	893	High	Cloud-4:4	1328	High
19	Fog-4:5	1121	High	Cloud-4:5	1513	High
20	Fog-4:10	1816	Low	Cloud-4:10	4024	High
21	Fog-5:5	1400	Low	Cloud-5:5	1908	High
22	Fog-5:10	2091	High	Cloud-5:10	2686	High
23	Fog-6:6	1648	High	Cloud-6:6	6181	High
24	Fog-6:10	1986	High	Cloud-6:10	7403	High
25	Fog-7:10	2229	High	Cloud-7:10	10095	High
26	Fog-8:10	2636	High	Cloud-8:10	12508	High
27	Fog-9:9	7315	High	Cloud-9:9	10323	High
28	Fog-10:5	2254	High	Cloud-10:5	3373	High

Table 4.7 shows specific ranges chosen to comprehensively understand each system's performance across varying operational scenarios. By distinguishing between lower and higher values, managing and optimizing the behaviours of the system becomes easier, ensuring they operate efficiently under different conditions. The "Low" range typically represents scenarios with minimal operational load, while the "High" range accounts for more intensive usage, allowing for tailored strategies to maintain optimal performance.

Table 4.7: Type of System (Fog or Cloud) and Average Execution Time

Crosstabulation: Type of System (Fog or Cloud) and Average Execution Time				
Type		**Average Execution Time**		**Total**
		High	**Low**	
System (Fog or Cloud)	Cloud-Based System	24	4	28
	Fog Based System	17	11	28
Total		**41**	**15**	**56**

Table 4.8 shows the approach for calculating the expected value from the row total of average execution time and column total of type of system (Fog or Cloud) also the total number of observations is 56.

Table 4.8: Expected Frequency

Calculation of Expected Frequency			
Total Average Execution Time	**Total Type (Fog or Cloud)**	**Expected Frequency**	**Expected Frequency**
41	28	(41* 28) / 56	20.5
15	28	(15* 28) / 56	7.5
41	28	(41* 28) / 56	20.5
15	28	(15* 28) / 56	7.5

Table 4.9: χ^2 Calculation

Observed and Expected Frequency for the calculation of $\chi 2$			
Observed Frequency (OF)	**Expected Frequency (EF)**	**$(OF - EF)^2$**	**$(OF - EF)^2 / EF$**
24	20.5	12.25	0.5975
4	7.5	12.25	1.6333
17	20.5	12.25	0.5975
11	7.5	12.5	1.6333
		Total ($\sum$)	**4.4616**

Degree of Freedom = (r-1) * (c-1)

= (2-1) * (2-1)

= 1

Table value @ 5% level of significance = 3.84

Therefore,

The calculated value of Chi-Square is found to be 4.4616

The tabulated value of Chi-Square is found to be 3.84

Accordingly, table 4.9 represents the calculation of Chi-Square test value using the observed and expected frequencies. The results confirm that the calculated value of Chi-Square 4.4616 is greater than the tabulated value of 3.84 at a 5% level of significance. So, it is clear that the null hypothesis is **rejected.**

It concludes that there is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure execution time. A notable contrast in performance, measured by execution time, emerged between the SMART FOG protocol-based and cloud-based systems. The findings reveal that the SMART FOG system exhibited superior performance with notably shorter execution times compared to its cloud-based counterpart.

$H_0$2: There is a significant difference between SMART FOG protocol-based systems and cloud-based systems based on the performance measure latency.

An alternative hypothesis is as follows

H_a2: There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure latency.

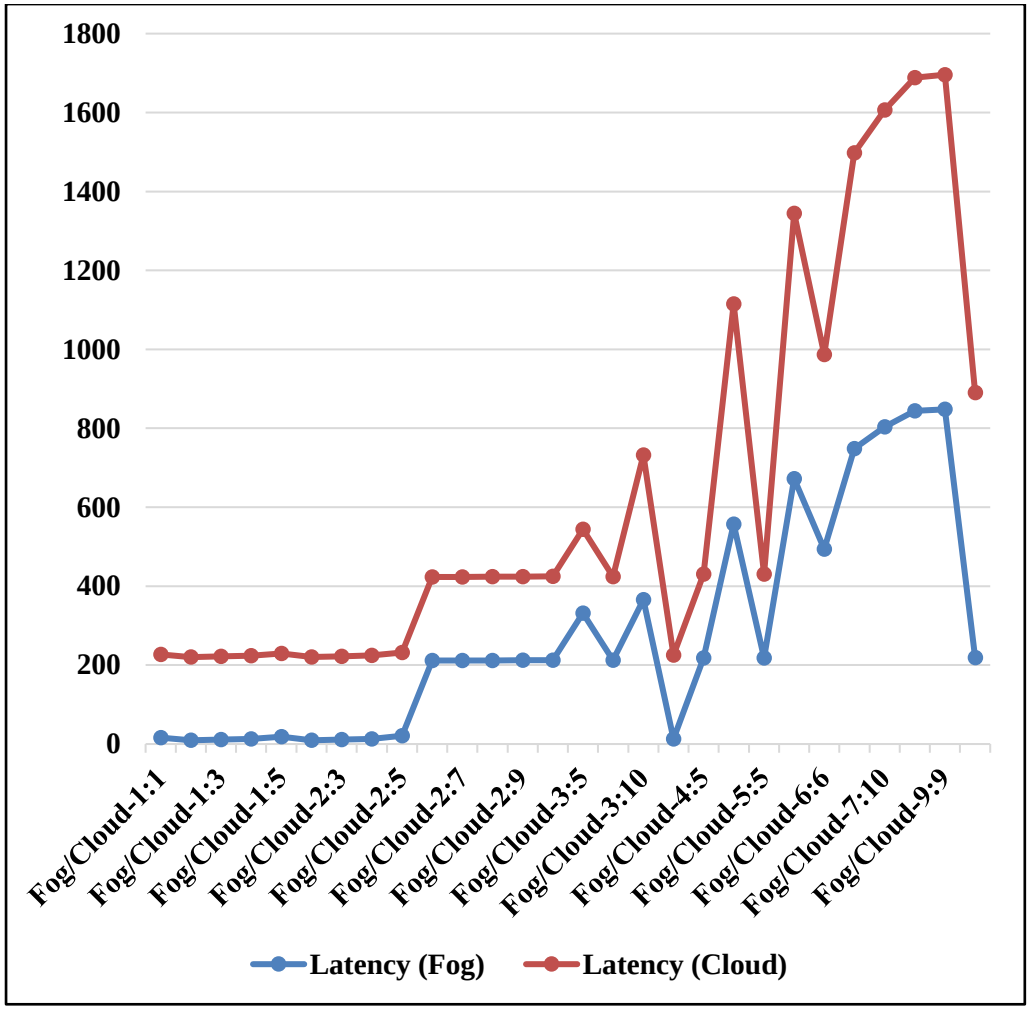

Figure 4.3: Fog Vs Cloud System Based on Latency

Figure 4.2, shows a comparative analysis between Fog fog-based system and Cloud-based system based on reduction in latency, as shown above, confirms that there is a large reduction in latency with use of Smart Fog based systems as compared to Cloud-based systems. In Fog system 10:5, 4:4, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4, 1:3, 1:2, and 1:1 there is a large reduction in latency value such as 453. 523, 198.926, 190.698, 198.131, 199.715, 201.366, 191.913, 197.730, 199.413, 201.161, and 194.086 respectively. So, it is very clear that Fog layer plays an important role in latency reduction.

Table 4.10: Latency Reduced due to Fog Computing Environment

System	Latency (Fog)	Latency (Cloud)	Latency Reduced Using Fog System
Fog/Cloud-1:1	16.414	210.499	194.086
Fog/Cloud-1:2	9.493	210.654	201.161
Fog/Cloud-1:3	11.278	210.692	199.413
Fog/Cloud-1:4	13.064	210.794	197.730
Fog/Cloud-1:5	18.946	210.859	191.913
Fog/Cloud-2:2	9.493	210.859	201.366
Fog/Cloud-2:3	11.278	210.993	199.715
Fog/Cloud-2:4	13.064	211.195	198.131
Fog/Cloud-2:5	20.707	211.405	190.698
Fog/Cloud-2:6	211.577	211.599	0.022
Fog/Cloud-2:7	211.787	211.857	0.070
Fog/Cloud-2:8	211.941	211.965	0.024
Fog/Cloud-2:9	212.107	212.184	0.077
Fog/Cloud-2:10	212.376	212.365	-0.011
Fog/Cloud-3:5	331.999	211.814	-120.186
Fog/Cloud-3:6	212.108	212.150	0.042
Fog/Cloud-3:10	366.026	365.965	-0.062
Fog/Cloud-4:4	13.064	211.990	198.926
Fog/Cloud-4:5	218.450	212.354	-6.096
Fog/Cloud-4:10	557.410	557.302	-0.108
Fog/Cloud-5:5	217.757	212.806	-4.952
Fog/Cloud-5:10	672.095	672.236	0.141
Fog/Cloud-6:6	493.522	493.557	0.035
Fog/Cloud-6:10	748.737	748.728	-0.008
Fog/Cloud-7:10	803.448	803.375	-0.073
Fog/Cloud-8:10	844.404	844.350	-0.054
Fog/Cloud-9:9	847.908	847.990	0.083
Fog/Cloud-10:5	218.625	672.148	453.523

Table 4.10, shows that Smart Fog system 10:5 which means 10 areas and 5 cameras takes a lower latency value of 218.62 as compared to the cloud system 10:5 with a latency value of 672.14. The experimental outcomes are further represented or categorized into high and low as shown below in the crosstabulation table.

Table 4.11 shows latency for the FOG SYSTEM is categorized into "Low" and "High" ranges, with the "Low" range including values from -5600.0000 to 1.0000 and the "High" range covering values from 1.0001 to 2100.0000. Similarly, the CLOUD SYSTEM latency is divided into "Low" and "High" ranges, where the "Low" range spans from 1.0001 to 15000.0000, and the "High" range includes values from -55000 to 1.0000.

Table 4.11: Classification of Fog and Cloud for Latency

Obs	Fog System	Latency	Rank	Cloud System	Latency	Rank
1	Fog-1:1	16.41	Low	Cloud-1:1	210.50	High
2	Fog-1:2	9.49	Low	Cloud-1:2	210.65	High
3	Fog-1:3	11.28	Low	Cloud-1:3	210.69	High
4	Fog-1:4	13.06	Low	Cloud-1:4	210.79	High
5	Fog-1:5	18.95	Low	Cloud-1:5	210.86	High
6	Fog-2:2	9.49	Low	Cloud-2:2	210.86	High
7	Fog-2:3	11.28	Low	Cloud-2:3	210.99	High
8	Fog-2:4	13.06	Low	Cloud-2:4	211.19	High
9	Fog-2:5	20.71	Low	Cloud-2:5	211.41	High
10	Fog-2:6	211.58	High	Cloud-2:6	211.60	High
11	Fog-2:7	211.79	Low	Cloud-2:7	211.86	High
12	Fog-2:8	211.94	Low	Cloud-2:8	211.97	High
13	Fog-2:9	212.11	Low	Cloud-2:9	212.18	High
14	Fog-2:10	212.38	High	Cloud-2:10	212.37	Low
15	Fog-3:5	332.00	High	Cloud-3:5	211.81	Low
16	Fog-3:6	212.11	Low	Cloud-3:6	212.15	High
17	Fog-3:10	366.03	High	Cloud-3:10	365.96	Low
18	Fog-4:4	13.06	Low	Cloud-4:4	211.99	High
19	Fog-4:5	218.45	High	Cloud-4:5	212.35	Low
20	Fog-4:10	557.41	High	Cloud-4:10	557.30	Low
21	Fog-5:5	217.76	High	Cloud-5:5	212.81	Low
22	Fog-5:10	672.10	Low	Cloud-5:10	672.24	High
23	Fog-6:6	493.52	Low	Cloud-6:6	493.56	High
24	Fog-6:10	748.74	High	Cloud-6:10	748.73	High
25	Fog-7:10	803.45	High	Cloud-7:10	803.37	Low
26	Fog-8:10	844.40	High	Cloud-8:10	844.35	Low
27	Fog-9:9	847.91	Low	Cloud-9:9	847.99	High
28	Fog-10:5	218.62	Low	Cloud-10:5	672.15	High

Table 4.12 shows specific ranges are chosen to provide a comprehensive understanding of each system's performance across various latency conditions. By distinguishing between lower and higher latency values, it becomes easier to optimize the systems' behaviors, ensuring they operate efficiently under different scenarios. This categorization aids in tailoring strategies to maintain optimal performance by addressing minimal and intensive latency conditions separately.

Table 4.12: Type of System (Fog or Cloud) and Latency

Crosstabulation: Type of System (Fog or Cloud) and Latency				
Type		**Latency**		**Total**
		High	**Low**	
System (Fog or Cloud)	Cloud-Based System	20	8	28
	Fog Based System	10	18	28
Total		**30**	**26**	**56**

Table 4.13 shows the approach for calculating the expected value from the row total of latency and column total type of system (Fog or Cloud) also the total number of observations is 56.

Table 4.13: Expected Frequency

Calculation of Expected Frequency			
Total of Latency	**Total Type (Fog or Cloud)**	**Expected Frequency**	**Expected Frequency**
30	28	(30 * 28) / 56	15
26	28	(26 * 28) / 56	13
30	28	(30 * 28) / 56	15
26	28	(26 * 28) / 56	13

Table 4.14:χ^2 Calculation

Observed and Expected Frequency for the calculation of χ^2			
Observed Frequency (OF)	**Expected Frequency (EF)**	**$(OF - EF)^2$**	**$(OF - EF)^2$ / EF**
20	15	25	1.67
8	13	25	1.92
10	15	25	1.67
18	13	25	1.92
		Total (∑)	**7.18**

Degree of Freedom = (r-1) * (c-1)

= (2-1) * (2-1)

= 1

Table value @ 5% level of significance = 3.841

Therefore,

The calculated value of Chi-Square is found to be 7.18.

The tabulated value of Chi-Square is found to be 3.841.

Accordingly, table 4.14 represents the calculation of Chi-Square test value using the observed and expected frequencies. The results confirm that the calculated value of Chi-Square 7.18 is greater than the tabulated value of 3.841 at a 5% level of significance. So, it is clear that the null hypothesis is **accepted.**

It concludes that there is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure latency. A notable contrast in performance, measured by latency, emerged between the SMART FOG protocol-based and cloud-based systems. The findings reveal that the SMART FOG system exhibited superior performance with notably shorter latency compared to its cloud-based counterpart.

$H_0$3: There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure energy consumed.

An alternative hypothesis is as follows

H_a3: There is significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure energy consumed.

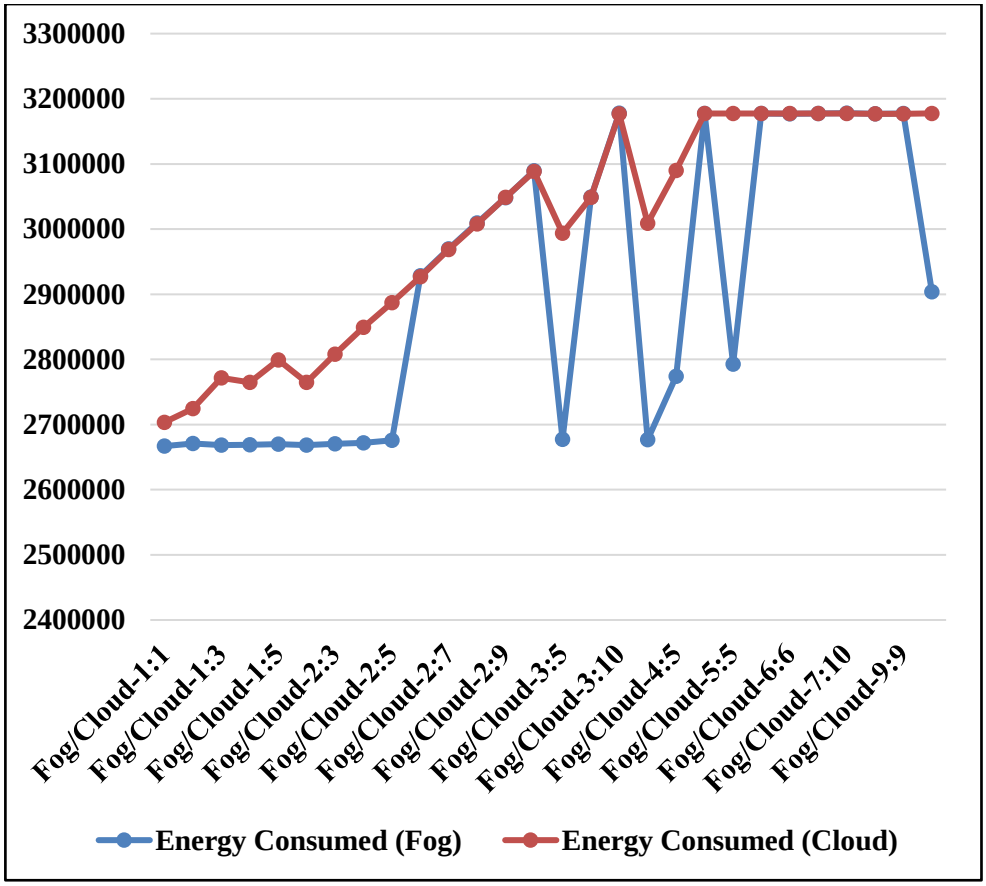

Figure 4.4: Fog Vs Cloud System Based on Energy Consumption (Joules)

Figure 4.4, shows a comparative analysis between Fog fog-based system and Cloud-based system based on reduction in energy consumption as shown below confirming that there is a large reduction in energy consumption with the use of a Smart Fog based system as compared to Cloud-based systems. In Fog system 10:5, 5:5, 4:5, 4:4, 3:5, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4, 1:3, 1:2, and 1:1 there is a large reduction in energy consumption such as 273694.729, 384391.984, 316112.779, 331766.905, 211288.876, 177375.370, 137333.715, 96247.432, 129245.955, 96035.138, 103230.811, 53272.127 and 36660.736 respectively.

Table 4.15: Energy Consumption Reduced due to Fog Computing Environment

System	Energy Consumed (Fog)	Energy Consumed (Cloud)	Energy Consumption Reduced Using Fog
Fog/Cloud-1:1	2666906.9783	2703567.7143	36660.736
Fog/Cloud-1:2	2670956.3531	2724228.4804	53272.127
Fog/Cloud-1:3	2668402.4564	2771633.2679	103230.811
Fog/Cloud-1:4	2668904.0258	2764939.1634	96035.138
Fog/Cloud-1:5	2669934.3309	2799180.2862	129245.955
Fog/Cloud-2:2	2668603.1406	2764850.5729	96247.432
Fog/Cloud-2:3	2670441.5028	2807775.2177	137333.715
Fog/Cloud-2:4	2671749.8597	2849125.2299	177375.370
Fog/Cloud-2:5	2675762.8380	2887051.7140	211288.876
Fog/Cloud-2:6	2928104.6330	2926944.7498	-1159.883
Fog/Cloud-2:7	2969367.4582	2968703.5879	-663.870
Fog/Cloud-2:8	3009438.8046	3007902.8161	-1535.989
Fog/Cloud-2:9	3048411.7523	3048899.5858	487.834
Fog/Cloud-2:10	3089402.5999	3088658.5915	-744.008
Fog/Cloud-3:5	2677098.6434	2993796.5055	316697.862
Fog/Cloud-3:6	3049029.0903	3048668.7878	-360.303
Fog/Cloud-3:10	3178035.5090	3176881.5438	-1153.965
Fog/Cloud-4:4	2676983.1865	3008750.0915	331766.905
Fog/Cloud-4:5	2773838.5932	3089951.3720	316112.779
Fog/Cloud-4:10	3177518.9069	3177179.7693	-339.138
Fog/Cloud-5:5	2792816.8478	3177208.8314	384391.984
Fog/Cloud-5:10	3177556.9957	3177196.9814	-360.014
Fog/Cloud-6:6	3176917.6581	3177369.9354	452.277
Fog/Cloud-6:10	3177409.1509	3177226.8895	-182.261
Fog/Cloud-7:10	3177797.6631	3177507.3385	-290.325
Fog/Cloud-8:10	3177042.9305	3177065.2212	22.291
Fog/Cloud-9:9	3177160.0221	3177118.2621	-41.760
Fog/Cloud-10:5	2903894.7132	3177589.4426	273694.729

Table 4.15 shows that The Smart Fog system 10:5 which means 10 areas and 5 cameras takes a lower energy consumption of 2903894.713 as compared to the cloud system 10:5 with an energy consumption of 3177589.443. The experimental outcomes are further represented or categorized into very high to very low as shown below in the crosstabulation table.

Table 4.16 shows that energy consumption ranges for both FOG SYSTEM and CLOUD SYSTEM are tailored to categorize their respective usage levels effectively. FOG SYSTEM's categories range from "Very High" (below -1150.0000) for extremely low consumption to "Very Low" (400.0000 to 400000.0000) for higher usage scenarios. In contrast, CLOUD SYSTEM starts with "Very Low" (below -1550.0000) and goes up to "Very High" (400.0000 to 400000.0000).

Table 4.16: Classification of Fog and Cloud for Energy Consumption

Obs.	Fog System	Energy Consum-ption	Rank	Cloud System	Energy Consum-ption	Rank
1	Fog-1:1	2666906.98	Very Low	Cloud-1:1	2703567.71	Very High
2	Fog-1:2	2670956.35	Very Low	Cloud-1:2	2724228.48	Very High
3	Fog-1:3	2668402.46	Very Low	Cloud-1:3	2771633.27	Very High
4	Fog-1:4	2668904.03	Very Low	Cloud-1:4	2764939.16	Very High
5	Fog-1:5	2669934.33	Very Low	Cloud-1:5	2799180.29	Very High
6	Fog-2:2	2668603.14	Very Low	Cloud-2:2	2764850.57	Very High
7	Fog-2:3	2670441.50	Very Low	Cloud-2:3	2807775.22	Very High
8	Fog-2:4	2671749.86	Very Low	Cloud-2:4	2849125.23	Very High
9	Fog-2:5	2675762.84	Very Low	Cloud-2:5	2887051.71	Very High
10	Fog-2:6	2928104.63	High	Cloud-2:6	2926944.75	Low
11	Fog-2:7	2969367.46	High	Cloud-2:7	2968703.59	Low
12	Fog-2:8	3009438.80	Very High	Cloud-2:8	3007902.82	Low
13	Fog-2:9	3048411.75	Very Low	Cloud-2:9	3048899.59	Very High
14	Fog-2:10	3089402.60	High	Cloud-2:10	3088658.59	Low
15	Fog-3:5	2677098.64	Very Low	Cloud-3:5	2993796.51	Very High
16	Fog-3:6	3049029.09	High	Cloud-3:6	3048668.79	Low
17	Fog-3:10	3178035.51	High	Cloud-3:10	3176881.54	Low
18	Fog-4:4	2676983.19	Very Low	Cloud-4:4	3008750.09	Very High
19	Fog-4:5	2773838.59	Very Low	Cloud-4:5	3089951.37	Very High
20	Fog-4:10	3177518.91	High	Cloud-4:10	3177179.77	Low
21	Fog-5:5	2792816.85	Very Low	Cloud-5:5	3177208.83	Very High
22	Fog-5:10	3177557.00	High	Cloud-5:10	3177196.98	Low
23	Fog-6:6	3176917.66	Very Low	Cloud-6:6	3177369.94	Very High
24	Fog-6:10	3177409.15	High	Cloud-6:10	3177226.89	Low
25	Fog-7:10	3177797.66	High	Cloud-7:10	3177507.34	Low
26	Fog-8:10	3177042.93	Low	Cloud-8:10	3177065.22	High
27	Fog-9:9	3177160.02	High	Cloud-9:9	3177118.26	Low
28	Fog-10:5	2903894.71	Very Low	Cloud-10:5	3177589.44	Very High

Table 4.17 shows, specific ranges are chosen to provide a comprehensive understanding of each system's performance across various Energy Consumption conditions. By distinguishing between Very Low, Low, High, Very High values, it becomes easier to optimize the systems' behaviors, ensuring they operate efficiently under different scenarios.

Table 4.17: Type of System (Fog or Cloud) and Energy Consumption

Crosstabulation: Type of System (Fog or Cloud) and Energy Consumption						
Count						
Type		Energy Consumption				Total
		Very Low	Low	High	Very High	
System (Fog or Cloud)	Cloud	0	11	1	16	28
	Fog	16	1	10	1	28
Total		**16**	**12**	**11**	**17**	**56**

Table 4.18, shows the approach for calculating the expected Frequency value from the row total of energy consumption and column total of type of system (Fog or Cloud) also the total number of observations is 56.

Table 4.18: Expected Frequency

Calculation of Expected Frequency			
Total of Energy Consumption	**Total Type (Fog or Cloud)**	**Expected Frequency**	**Expected Frequency**
16	28	(16 * 28) / 56	8.0
12	28	(12 * 28) / 56	6.0
11	28	(11 * 28) / 56	5.5
17	28	(17 * 28) / 56	8.5
16	28	(16 * 28) / 56	8.0
12	28	(12 * 28) / 56	6.0
11	28	(11 * 28) / 56	5.5
17	28	(17 * 28) / 56	8.5

Table 4.19: χ^2 Calculation

Observed and Expected Frequency for the calculation of χ^2			
Observed Frequency (OF)	**Expected Frequency (EF)**	**$(OF - EF)^2$**	**$(OF - EF)^2 / EF$**
0	8.0	64.00	8.00
11	6.0	25.00	4.17
1	5.5	20.25	3.68
16	8.5	56.25	6.62
16	8.0	64.00	8.00
1	6.0	25.00	4.17
10	5.5	20.25	3.68
1	8.5	56.25	6.62
		Total (∑)	**44.93**

Degree of Freedom =(r-1) * (c-1)

= (2-1) * (4-1)

= 3

Table value @ 5% level of significance = 7.81

Therefore,

The calculated value of Chi-Square is found to be 44.93

The tabulated value of Chi-Square is found to be 7.81

Accordingly, table 4.19 represents the calculation of Chi-Square test value using the observed and expected frequencies. The results confirm that the calculated value of Chi-Square 44.93 is much greater than the tabulated value of 7.81 at a 5% level of significance. So, it is clear that the null hypothesis is **rejected.**

It concludes that there is significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure energy consumed. A notable contrast in performance, measured by energy consumption, emerged between the SMART FOG protocol-based and cloud-based systems. The findings reveal that the SMART FOG system exhibited superior performance with notably lower energy consumption as compared to its cloud-based counterpart.

$H_0$4: There is significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure cost of execution.

An alternative hypothesis is as follows

H_a4: There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure cost of execution.

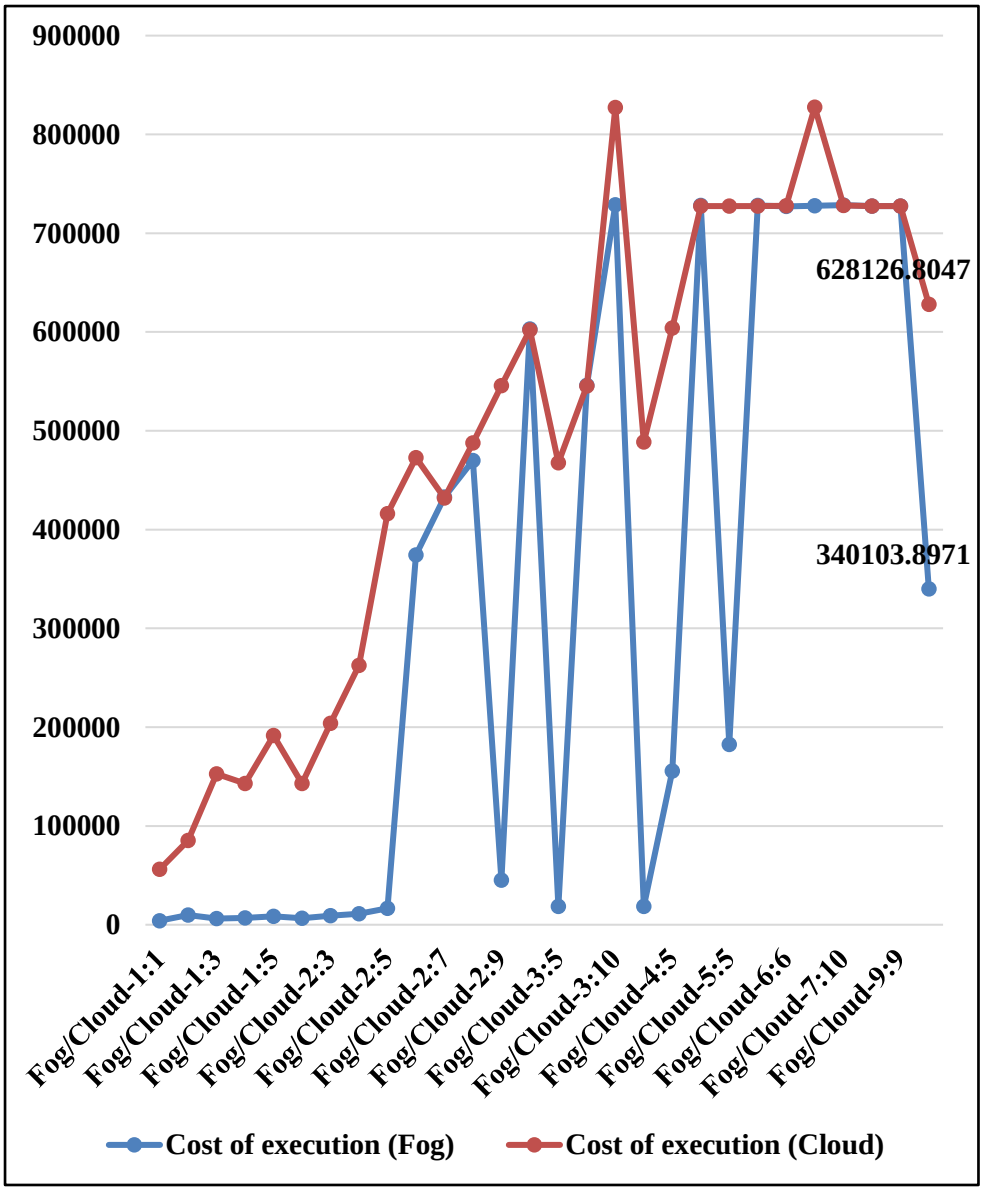

Figure 4.5: Fog Vs Cloud System Based on Cost of Execution (ms)

Figure 4.5, shows comparative analysis between Fog based system and Cloud based system based on a reduction in cost of execution as shown below confirms that there is a large reduction in cost of execution with the use of a Smart fog-based system as compared to Cloud-based systems.

Table 4.20: Cost of Execution Reduced due to Fog Computing Environment

System	Cost of execution (Fog)	Cost of execution (Cloud)	Cost of execution Reduced Using Fog System
Fog/Cloud-1:1	4121.285714	56096	51974.71429
Fog/Cloud-1:2	9862.171429	85387.21272	75525.04129
Fog/Cloud-1:3	6241.457143	152594	146352.5429
Fog/Cloud-1:4	6952.542857	143103.6241	136151.0813
Fog/Cloud-1:5	8413.228571	191648.0007	183234.7721
Fog/Cloud-2:2	6525.971429	142978.0275	136452.056
Fog/Cloud-2:3	9132.257143	203833.2201	194700.9629
Fog/Cloud-2:4	10987.14286	262456.0221	251468.8792
Fog/Cloud-2:5	16676.42857	416225.2147	399548.7862
Fog/Cloud-2:6	374426.8214	472782.4301	98355.60871
Fog/Cloud-2:7	432926.0167	431984.8335	-941.1832586
Fog/Cloud-2:8	469736.0268	487558.4228	17822.39598
Fog/Cloud-2:9	44988.81339	545680.4254	500691.6121
Fog/Cloud-2:10	603102.4201	602047.6234	-1054.796652
Fog/Cloud-3:5	18570.22857	467559.6027	448989.3741
Fog/Cloud-3:6	545864.0268	545353.2181	-510.8087055
Fog/Cloud-3:10	728759.2027	827123.2013	98363.99866
Fog/Cloud-4:4	18406.54286	488759.6234	470353.0806
Fog/Cloud-4:5	155720.5371	603880.4261	448159.889
Fog/Cloud-4:10	728026.8047	727546.002	-480.8026788
Fog/Cloud-5:5	182626.4171	727587.204	544960.7869
Fog/Cloud-5:10	728080.804	727570.404	-510.3999999
Fog/Cloud-6:6	727174.4013	727815.6047	641.2033483
Fog/Cloud-6:10	727871.2013	827612.8054	99741.60402
Fog/Cloud-7:10	728422.0033	728010.404	-411.5993303
Fog/Cloud-8:10	727352.0027	727383.6047	31.60200911
Fog/Cloud-9:9	727518.006	727458.802	-59.20401781
Fog/Cloud-10:5	340103.8971	628126.8047	288022.9075

Table 4.20 shows Fog system 10:5, 6:10, 5:5, 4:5, 4:4, 3:10, 3:5, 2:9, 2:8, 2:6, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4, 1:3,1:2 and 1:1 there is large reduction in cost of execution such as 288022.9075, 99741, 60402, 544960.7869, 448159.0806, 98363.99866, 448989.3741, 500691.6121, 98355.60871, 399548.7862, 251468.8792, 194700.9629, 136452.056, 183234.7721, 183234.7721, 136151.0813, 146352.5429, 75525.04129 and 51974.7142 respectively. The Smart Fog system 10:5 which means 10 areas and 5 cameras takes a lower cost of execution of 340103.8971 as compared to the cloud system 10:5 with a cost of execution of 628126.8047. The experimental outcomes are further represented or categorized into very high to very low as shown below in the crosstabulation table.

Table 4.21 shows the FOG SYSTEM's "Very High" (below -950.0000) indicates exceptionally low costs due to optimized processes. "High" (-950.0001 to 30.0000)

suggests moderate expenses with efficient operations, while "Low" (30.0001 to 90000.0000) represents typical costs within budget. "Very Low" (90001.0000 to 600000.0000) signifies higher expenses possibly from less optimized setups.

For the CLOUD SYSTEM, "Very Low" (below -950.0000) and "Low" (-950.0001 to 31.0000) denote economical costs and efficient management. "High" (31.0001 to 100000.0000) reflects standard expenses akin to FOG SYSTEM's "Low" range, while "Very High" (100000.0001 to 600000.0000) indicates higher costs due to complex tasks.

Table 4.21: Classification of Fog and Cloud for Execution

Obs.	Fog System	Cost of Execution	Classi-fication	Cloud System	Cost of Execution	Rank
1	Fog-1:1	4121.29	Low	Cloud-1:1	56096.00	High
2	Fog-1:2	9862.17	Low	Cloud-1:2	85387.21	High
3	Fog-1:3	6241.46	Very Low	Cloud-1:3	152594.00	Very High
4	Fog-1:4	6952.54	Very Low	Cloud-1:4	143103.62	Very High
5	Fog-1:5	8413.23	Very Low	Cloud-1:5	191648.00	Very High
6	Fog-2:2	6525.97	Very Low	Cloud-2:2	142978.03	Very High
7	Fog-2:3	9132.26	Very Low	Cloud-2:3	203833.22	Very High
8	Fog-2:4	10987.14	Very Low	Cloud-2:4	262456.02	Very High
9	Fog-2:5	16676.43	Very Low	Cloud-2:5	416225.21	Very High
10	Fog-2:6	374426.82	Very Low	Cloud-2:6	472782.43	Very High
11	Fog-2:7	432926.02	High	Cloud-2:7	431984.83	Low
12	Fog-2:8	469736.03	Low	Cloud-2:8	487558.42	High
13	Fog-2:9	44988.81	Very Low	Cloud-2:9	545680.43	Very High
14	Fog-2:10	603102.42	Very High	Cloud-2:10	602047.62	Very Low
15	Fog-3:5	18570.23	Very Low	Cloud-3:5	467559.60	Very High
16	Fog-3:6	545864.03	High	Cloud-3:6	545353.22	Low
17	Fog-3:10	728759.20	Very Low	Cloud-3:10	827123.20	Very High
18	Fog-4:4	18406.54	Very Low	Cloud-4:4	488759.62	Very High
19	Fog-4:5	155720.54	Very Low	Cloud-4:5	603880.43	Very High
20	Fog-4:10	728026.80	High	Cloud-4:10	727546.00	Low
21	Fog-5:5	182626.42	Very Low	Cloud-5:5	727587.20	Very High
22	Fog-5:10	728080.80	High	Cloud-5:10	727570.40	Low
23	Fog-6:6	727174.40	Low	Cloud-6:6	727815.60	High
24	Fog-6:10	727871.20	Very Low	Cloud-6:10	827612.81	Very High
25	Fog-7:10	728422.00	High	Cloud-7:10	728010.40	Low
26	Fog-8:10	727352.00	Low	Cloud-8:10	727383.60	High
27	Fog-9:9	727518.01	High	Cloud-9:9	727458.80	Low
28	Fog-10:5	340103.90	Very Low	Cloud-10:5	628126.80	Very High

Table 4.22 These classifications help ranges guide cost-effective strategies and resource allocation cost of execution based on operational needs. specific ranges are chosen to provide a comprehensive understanding of each system's performance across various Energy Consumption conditions. By distinguishing between Very Low, Low, High, Very High values, it becomes easier to optimize the systems' behaviors, ensuring they operate efficiently under different scenarios.

Table 4.22: Type of System (Fog or Cloud) and Cost of Execution

Crosstabulation: Type of System (Fog or Cloud) and Cost of Execution						
Count						
Type		**Cost of Execution**				**Total**
		Very Low	**Low**	**High**	**Very High**	
System (Fog or Cloud)	Cloud	1	6	5	16	28
	Fog	16	5	6	1	28
Total		17	11	11	17	56

Table 4.23 shows the approach for calculating the expected value from the row total of cost of execution and column total of type of system (Fog or Cloud) also the total number of observations is 56.

Table 4.23: Expected Frequency

Calculation of Expected Frequency			
Total Cost of Execution	**Total Type (Fog or Cloud)**	**Expected Frequency**	**Expected Frequency**
17	28	(17 * 28) / 56	8.5
11	28	(11 * 28) / 56	5.5
11	28	(11 * 28) / 56	5.5
17	28	(17 * 28) / 56	8.5
17	28	(17 * 28) / 56	8.5
11	28	(11 * 28) / 56	5.5
11	28	(11 * 28) / 56	5.5
17	28	(17 * 28) / 56	8.5

Table 4.24: χ^2 Calculation

Observed and Expected Frequency for the calculation of χ^2			
Observed Frequency (OF)	**Expected Frequency (EF)**	**(OF - EF)2**	**(OF - EF)2 / EF**
1	8.5	56.25	6.62
6	5.5	00.25	0.05
5	5.5	00.25	0.05
16	8.5	56.25	6.62
16	8.5	56.25	6.62
5	5.5	00.25	0.05
6	5.5	00.25	0.05
1	8.5	56.25	6.62
		Total (∑)	**26.65**

Degree of Freedom = (r-1) * (c-1)

= (2-1) * (4-1)

= 3

Table value @ 5% level of significance = 7.81

Therefore,

The calculated value of Chi-Square is found to be 26.65

The tabulated value of Chi-Square is found to be 7.81

Accordingly, table 4.24 represents the calculation of the Chi-Square test value using the observed and expected frequencies. The results confirm that the calculated value of Chi-Square 26.65 is greater than the tabulated value of 7.81 at a 5% level of significance. So, it is clear that the null hypothesis is **accepted.**

It concludes that there is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure cost of execution. A notable contrast in performance, measured by cost of execution, emerged between the SMART FOG protocol-based and cloud-based systems. The findings reveal that the SMART FOG system exhibited superior performance with a notably lower cost of execution as compared to its cloud-based counterpart.

$H_0$5: There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure of total network usage.

An alternative hypothesis is as follows

H_a5: There is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure of total network usage.

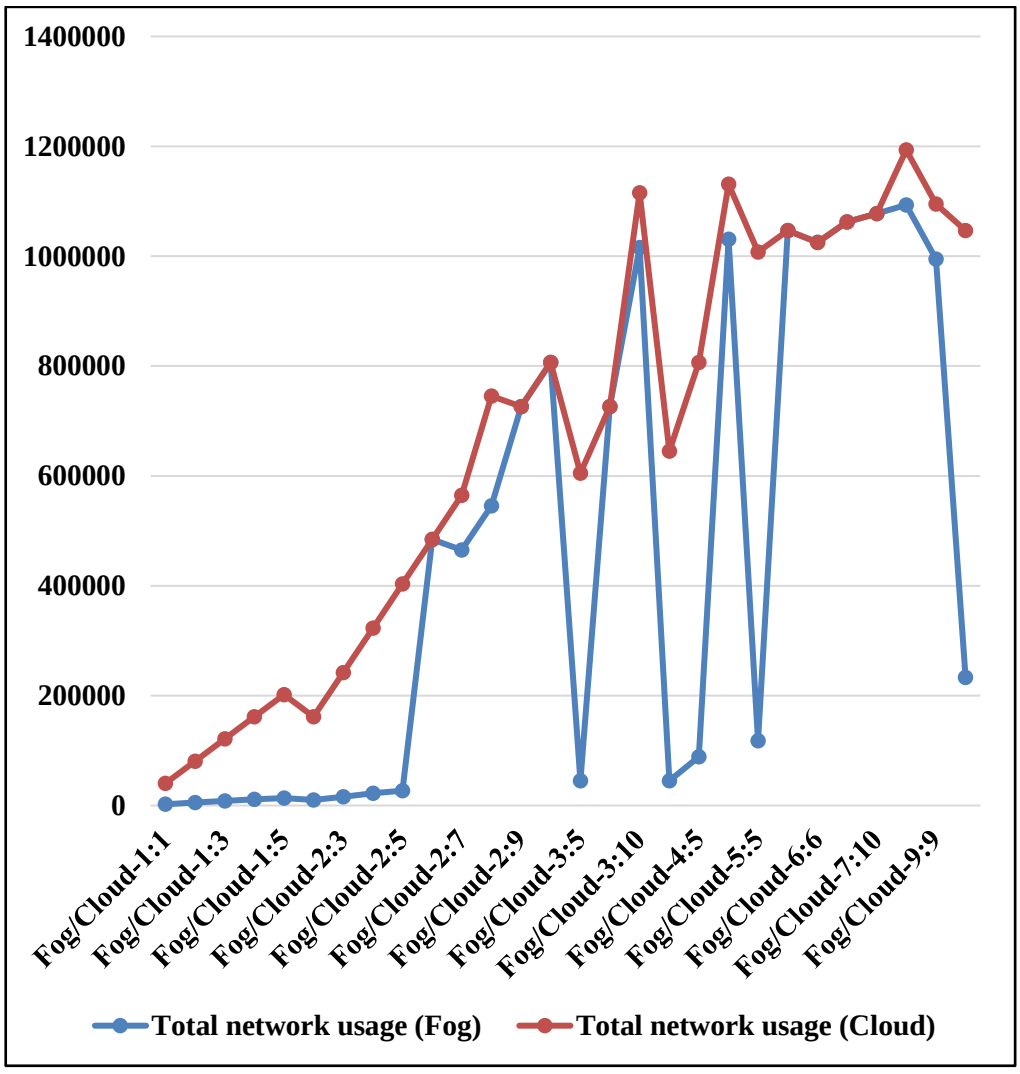

Figure 4.6: Fog Vs Cloud System Based on Total Network Usage (B/s)

Figure 4.6, shows that comparative analysis between Fog-based system and Cloud based system based on reduction in total network usage as shown confirms that there is large reduction in total network usage with use of Smart Fog based system as compared to Cloud-based systems.

Table 4.25: Total Network Usage Reduced due to Fog Computing Environment

System	Total network usage (Fog)	Total network usage (Cloud)	Reduction in Total Network Usage Using Fog System
Fog/Cloud-1:1	2309.9	40452.6	38142.7
Fog/Cloud-1:2	5537.8	80808.4	75270.6
Fog/Cloud-1:3	8357.7	121164.2	112806.5
Fog/Cloud-1:4	11383.6	161520	150136.4
Fog/Cloud-1:5	13887.4	201875.8	187988.4
Fog/Cloud-2:2	10055.6	161521.8	151466.2
Fog/Cloud-2:3	16103.4	242233.4	226130
Fog/Cloud-2:4	22457.2	322945	300487.8
Fog/Cloud-2:5	27266.8	403656.6	376389.8
Fog/Cloud-2:6	484368.2	484368.2	0
Fog/Cloud-2:7	464879.8	564879.8	100000
Fog/Cloud-2:8	545391.4	745391.4	200000
Fog/Cloud-2:9	725903	725903	0
Fog/Cloud-2:10	806414.6	806414.6	0
Fog/Cloud-3:5	44826.2	605137.4	560311.2
Fog/Cloud-3:6	725904.8	725904.8	0
Fog/Cloud-3:10	1015474.4	1115474.4	100000
Fog/Cloud-4:4	44812.4	645395	600582.6
Fog/Cloud-4:5	88728.2	806418.2	717690
Fog/Cloud-4:10	1031034.2	1131034.2	100000
Fog/Cloud-5:5	118114	1007699	889585
Fog/Cloud-5:10	1046594	1046594	0
Fog/Cloud-6:6	1024814.6	1024814.6	0
Fog/Cloud-6:10	1062153.8	1062153.8	0
Fog/Cloud-7:10	1077713.6	1077713.6	0
Fog/Cloud-8:10	1093273.4	1193273.4	100000
Fog/Cloud-9:9	994831	1094831	100000
Fog/Cloud-10:5	233479	1046603	813124

Table 4.25 Total Network Usage Reduced due to Fog Computing Environment Fog system10:5, 9:9, 8:10, 5:5, 4:10, 4:5, 4:4, 3:10, 3:5, 2:8, 2:7, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4, 1:3, 1:2 and 1:1 there is large reduction in total network usage such as 813124, 100000, 100000, 889585, 100000, 717690, 600582.6, 100000, 560311.2, 200000, 100000, 376389.8, 300487.8, 226130, 151466.2, 187988.4, 150136.4, 112806.6, 75270.6, and 38142.7 respectively.

The Smart Fog system 10:5 which means 10 number of areas and 5 cameras reduces total network usage of 233479 as compared to the cloud system 10:5 with high total

network usage of 1046603. The experimental outcomes are further represented or categorized into very high to very low as shown below in the crosstabulation table.

Table 4.26 shows the network usage ranges for both FOG SYSTEM and CLOUD SYSTEM effectively categorize their activity levels. In FOG SYSTEM, "No Change" denotes 0 usage, typical during idle periods. "Low" (38000.0001 to 150000.0000) indicates moderate usage for regular data exchanges. "Very Low" (150000.0001 to 900000.0000) suggests increased activity, possibly due to extensive data processing. "High" (900000.0001 to 1000000.0000) represents intensified data transfer or operational demands. "Very High" (above 1000000.0000) indicates extensive network activity or intensive data processing.

Similarly, in CLOUD SYSTEM, "No Change" signifies 0 usage, "High" (38000.0001 to 150000.0000) denotes typical activity levels, "Very High" (150000.0001 to 900000.0000) indicates significant traffic, "Low" (900000.0001 to 1000000.0000) suggests reduced activity, and "Very Low" (above 1000000.0000) signifies minimal network use or efficient management.

Table 4.26: Classification of Fog and Cloud for Total Network Usage

Obs	Fog System	Total Network Usage	Rank	Cloud System	Total Network Usage	Rank
1	Fog-1:1	2309.90	Low	Cloud-1:1	40452.60	High
2	Fog-1:2	5537.80	Low	Cloud-1:2	80808.40	High
3	Fog-1:3	8357.70	Low	Cloud-1:3	121164.20	High
4	Fog-1:4	11383.60	Very Low	Cloud-1:4	161520.00	Very High
5	Fog-1:5	13887.40	Very Low	Cloud-1:5	201875.80	Very High
6	Fog-2:2	10055.60	Very Low	Cloud-2:2	161521.80	Very High
7	Fog-2:3	16103.40	Very Low	Cloud-2:3	242233.40	Very High
8	Fog-2:4	22457.20	Very Low	Cloud-2:4	322945.00	Very High
9	Fog-2:5	27266.80	Very Low	Cloud-2:5	403656.60	Very High
10	Fog-2:6	484368.20	No Change	Cloud-2:6	484368.20	No Change
11	Fog-2:7	464879.80	Low	Cloud-2:7	564879.80	High
12	Fog-2:8	545391.40	Very Low	Cloud-2:8	745391.40	Very High
13	Fog-2:9	725903.00	No Change	Cloud-2:9	725903.00	No Change
14	Fog-2:10	806414.60	No Change	Cloud-2:10	806414.60	No Change
15	Fog-3:5	44826.20	Very Low	Cloud-3:5	605137.40	Very High
16	Fog-3:6	725904.80	No Change	Cloud-3:6	725904.80	No Change
17	Fog-3:10	1015474.40	Low	Cloud-3:10	1115474.40	High
18	Fog-4:4	44812.40	Very Low	Cloud-4:4	645395.00	Very High
19	Fog-4:5	88728.20	Very Low	Cloud-4:5	806418.20	Very High
20	Fog-4:10	1031034.20	Low	Cloud-4:10	1131034.20	High
21	Fog-5:5	118114.00	Very Low	Cloud-5:5	1007699.00	Very High
22	Fog-5:10	1046594.00	No Change	Cloud-5:10	1046594.00	No Change
23	Fog-6:6	1024814.60	No Change	Cloud-6:6	1024814.60	No Change
24	Fog-6:10	1062153.80	No Change	Cloud-6:10	1062153.80	No Change
25	Fog-7:10	1077713.60	No Change	Cloud-7:10	1077713.60	No Change
26	Fog-8:10	1093273.40	Low	Cloud-8:10	1193273.40	High
27	Fog-9:9	994831.00	Low	Cloud-9:9	1094831.00	High
28	Fog-10:5	233479.00	Very Low	Cloud-10:5	1046603.00	Very High

Table 4.27 These classifications help ranges guide cost-effective strategies and resource allocation cost of execution based on Total Network Usage. specific ranges are chosen to provide a comprehensive understanding of each system's performance across various Energy Consumption conditions. By distinguishing between Very Low, Low, High, Very High values, it becomes easier to optimize the systems' behaviors, ensuring they operate efficiently under different scenarios.

Table 4.27: Type of System (Fog or Cloud) and Total Network Usage

Crosstabulation: Type of System (Fog or Cloud) and Total Network Usage							
Count							
Type		Total Network Usage					Total
		Very Low	Low	No Change	High	Very High	
System (Fog or Cloud)	Cloud	0	0	8	8	12	28
	Fog	12	8	8	0	0	28
Total		**12**	**8**	**16**	**8**	**12**	**56**

Table 4.28 shows the approach for calculating the expected value from the row total of total network usage and column total of type of system (Fog or Cloud) also the total number of observations is 56.

Table 4.28: Expected Frequency

Calculation of Expected Frequency			
Total Network Usage	**Total Type (Fog or Cloud)**	**Expected Frequency**	**Expected Frequency**
12	28	(12 * 28) / 56	6
8	28	(8 *28) / 56	4
16	28	(16 * 28) / 56	8
8	28	(8 * 28) / 56	4
12	28	(12 * 28) / 56	6
12	28	(12 * 28) / 56	6
8	28	(8 * 28) / 56	4
16	28	(16 * 28) / 56	8
8	28	(8 * 28) / 56	4
12	28	(12 * 28) / 56	6

Table 4.29: χ^2 Calculation

Observed and Expected Frequency for the calculation of χ^2			
Observed Frequency (OF)	**Expected Frequency (EF)**	**(OF - EF)2**	**(OF - EF)2 / EF**
0	6	36	6.00
0	4	16	4.00
8	8	0	0.00
8	4	16	4.00
12	6	36	6.00
12	6	36	6.00
8	4	16	4.00
8	8	0	0.00
0	4	16	4.00
0	6	36	6.00
		Total (∑)	**30.00**

Degree of Freedom = (r-1) * (c-1)

= (2-1) * (5-1)

= 4

Table value @ 5% level of significance = 9.49

Therefore,

The calculated value of Chi-Square is found to be 30.00.

The tabulated value of Chi-Square is found to be 9.49

Accordingly, table 4.29 represents the calculation of the Chi-Square test value using the observed and expected frequencies. The results confirm that the calculated value of Chi-Square 30 is much greater than the tabulated value of 9.49 at a 5% level of significance. So, it is clear that the null hypothesis is **rejected.**

It concludes that there is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure of total network usage. A notable contrast in performance, measured by total network usage, emerged between the SMART FOG protocol-based and cloud-based systems. The findings reveal that the SMART FOG system exhibited superior performance with notably lower total network usage as compared to its cloud-based counterpart.

$H_0$6: There is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure computational power consumed.

An alternative hypothesis is as follows

H_a6: There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure computational power consumed.

The comparative analysis between Fog based system and Cloud based system based on a reduction in computational power consumed as shown below confirms that there is large reduction in computational power consumed with use of Smart Fog based system as compared to cloud-based systems.

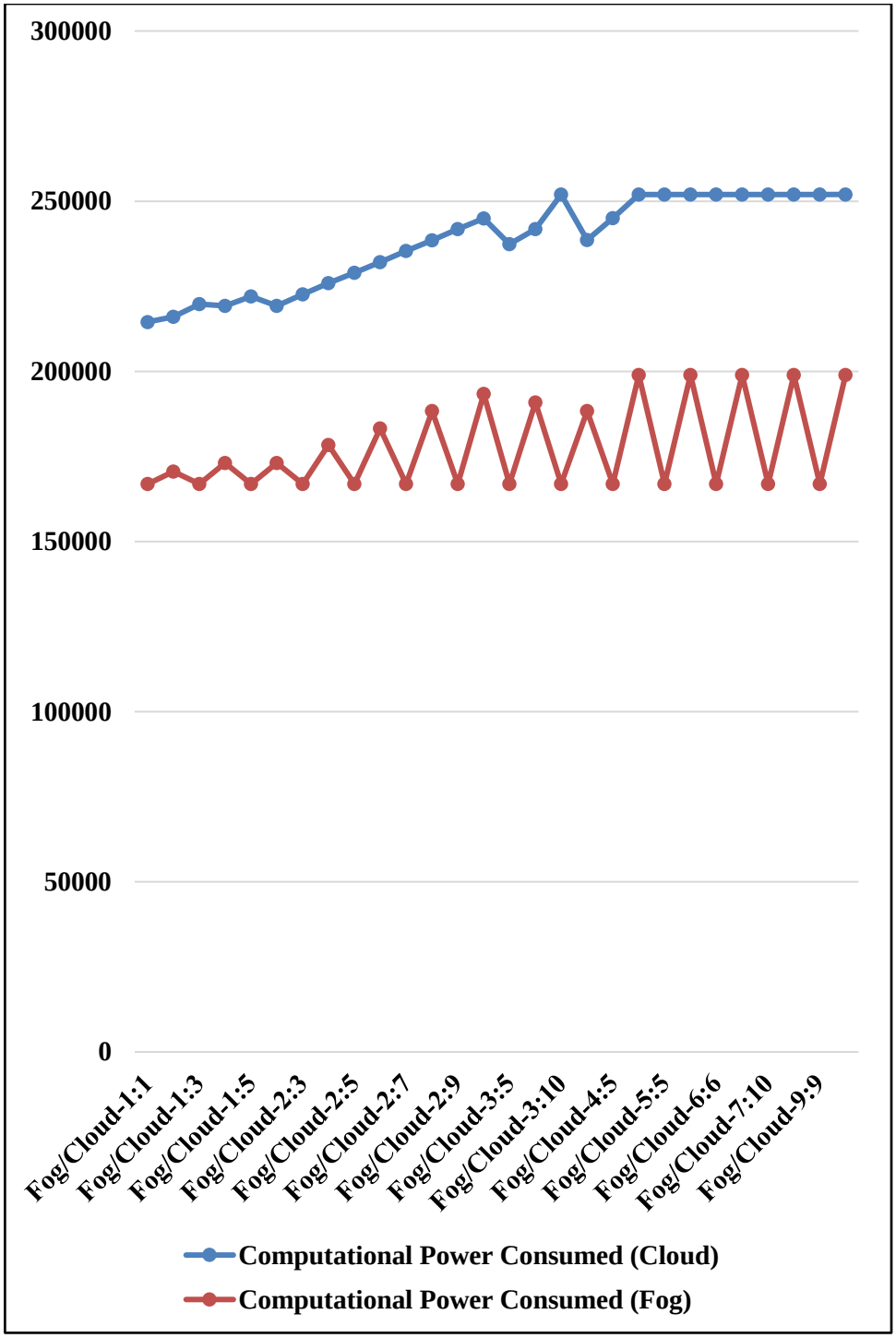

Figure 4.7: Fog Vs Cloud System Based on Computational Power (W)

Figure 4.7, shows there is a large reduction in computational power consumed in all cases for Fog system as compared to cloud-based system so based on the results it can be concluded that there is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure computational power consumed by Fog devices in comparison to Cloud devices.

Table 4.30: Computational Power Reduced due to Fog Computing Environment

System Devices	Computational Power Consumed (Cloud)	Computational Power Consumed (Fog)	Reduction in Computational Power by Fog
Fog/Cloud-1:1	214499.73	166866.599	47633.131
Fog/Cloud-1:2	216031.3185	170536.7029	45494.61562
Fog/Cloud-1:3	219790.5181	166867.599	52922.91914
Fog/Cloud-1:4	219259.6757	173085.1916	46174.48403
Fog/Cloud-1:5	221974.9967	166868.599	55106.39769
Fog/Cloud-2:2	219252.6504	173079.6459	46173.00457
Fog/Cloud-2:3	222656.5748	166869.599	55786.97577
Fog/Cloud-2:4	225935.6307	178355.2394	47580.39134
Fog/Cloud-2:5	228943.2009	166870.599	62072.60192
Fog/Cloud-2:6	232106.7187	183226.7413	48879.97732
Fog/Cloud-2:7	235418.1945	166871.599	68546.59552
Fog/Cloud-2:8	238526.6933	188294.7163	50231.97703
Fog/Cloud-2:9	241777.7372	166872.599	74905.13815
Fog/Cloud-2:10	244930.6263	193350.0278	51580.59848
Fog/Cloud-3:5	237408.0629	166873.599	70534.46388
Fog/Cloud-3:6	241759.4349	190846.6661	50912.76876
Fog/Cloud-3:10	251926.7064	166874.599	85052.10742
Fog/Cloud-4:4	238593.8823	188347.7557	50246.12653
Fog/Cloud-4:5	245033.1438	166875.599	78157.5448
Fog/Cloud-4:10	251950.3557	198891.4536	53058.90215
Fog/Cloud-5:5	251952.6603	166876.599	85076.06133
Fog/Cloud-5:10	251951.7206	198892.531	53059.18959
Fog/Cloud-6:6	251965.4359	166877.599	85087.83688
Fog/Cloud-6:10	251954.0923	198894.4033	53059.68905
Fog/Cloud-7:10	251976.3319	166878.599	85097.73295
Fog/Cloud-8:10	251941.272	198884.2828	53056.98919
Fog/Cloud-9:9	251945.4782	166879.599	85065.87919
Fog/Cloud-10:5	251982.8428	198917.0991	53065.74369

Figure 4.30 shows that The Smart Fog system 10:5 which means 10 areas and 5 cameras reduces computational power of 198917.0991 as compared to the cloud system 10:5 with a high computational power of 251982.8428. The experimental outcomes are further represented or categorized into very high to very low as shown below in the crosstabulation table.

Table 4.31 shows that the computational power consumed ranges for both FOG SYSTEM and CLOUD SYSTEM effectively categorize their operational intensity. In FOG SYSTEM, "Low" (45000.0000 to 48000.0000) signifies modest computational demands, likely involving basic processing tasks. "Very Low" (48000.0001 to 86000.0000) indicates slightly higher power consumption, potentially due to more complex computations or increased workload. Moving to "High" (86000.0001 to 100000.0000), it denotes significant computational power usage, indicative of intensive processing requirements or larger-scale operations. "Very High" (above 100000.0000) suggests extensive power consumption, possibly involving complex simulations or heavy data analytics.

Similarly, in CLOUD SYSTEM, "High" (45000.0000 to 48000.0000) and "Very High" (48000.0001 to 86000.0000) reflect varying degrees of computational intensity. "Low" (86000.0001 to 100000.0000) suggests reduced demands, while "Very Low" (above 100000.0000) indicates minimal power usage or highly efficient computational management.

Table 4.31: Classification of Fog and Cloud for Computational Power

Obs.	Fog System	Compu-tational Power	Rank	Cloud System	Compu-tational Power	Rank
1	Fog-1:1	166866.60	Low	Cloud-1:1	214499.73	High
2	Fog-1:2	170536.70	Low	Cloud-1:2	216031.32	High
3	Fog-1:3	166867.60	Very Low	Cloud-1:3	219790.52	Very High
4	Fog-1:4	173085.19	Low	Cloud-1:4	219259.68	High
5	Fog-1:5	166868.60	Very Low	Cloud-1:5	221975.00	Very High
6	Fog-2:2	173079.65	Low	Cloud-2:2	219252.65	High
7	Fog-2:3	166869.60	Very Low	Cloud-2:3	222656.57	Very High
8	Fog-2:4	178355.24	Low	Cloud-2:4	225935.63	High
9	Fog-2:5	166870.60	Very Low	Cloud-2:5	228943.20	Very High
10	Fog-2:6	183226.74	Very Low	Cloud-2:6	232106.72	Very High
11	Fog-2:7	166871.60	Very Low	Cloud-2:7	235418.19	Very High
12	Fog-2:8	188294.72	Very Low	Cloud-2:8	238526.69	Very High
13	Fog-2:9	166872.60	Very Low	Cloud-2:9	241777.74	Very High
14	Fog-2:10	193350.03	Very Low	Cloud-2:10	244930.63	Very High
15	Fog-3:5	166873.60	Very Low	Cloud-3:5	237408.06	Very High
16	Fog-3:6	190846.67	Very Low	Cloud-3:6	241759.43	Very High
17	Fog-3:10	166874.60	Very Low	Cloud-3:10	251926.71	Very High
18	Fog-4:4	188347.76	Very Low	Cloud-4:4	238593.88	Very High
19	Fog-4:5	166875.60	Very Low	Cloud-4:5	245033.14	Very High
20	Fog-4:10	198891.45	Very Low	Cloud-4:10	251950.36	Very High
21	Fog-5:5	166876.60	Very Low	Cloud-5:5	251952.66	Very High
22	Fog-5:10	198892.53	Very Low	Cloud-5:10	251951.72	Very High
23	Fog-6:6	166877.60	Very Low	Cloud-6:6	251965.44	Very High
24	Fog-6:10	198894.40	Very Low	Cloud-6:10	251954.09	Very High
25	Fog-7:10	166878.60	Very Low	Cloud-7:10	251976.33	Very High
26	Fog-8:10	198884.28	Very Low	Cloud-8:10	251941.27	Very High
27	Fog-9:9	166879.60	Very Low	Cloud-9:9	251945.48	Very High
28	Fog-10:5	198917.10	Very Low	Cloud-10:5	251982.84	Very High

Table 4.32 These classifications help ranges guide cost-effective strategies and resource allocation cost of execution based on Computational Power. specific ranges are chosen to provide a comprehensive understanding of each system's performance across various Energy Consumption conditions. By distinguishing between Very Low, Low, High, Very High values, it becomes easier to optimize the systems' behaviors, ensuring they operate efficiently under different scenarios.

Table 4.32: Type of System (Fog or Cloud) and Computational Power

Type of System (Fog or Cloud) and Computational Power Crosstabulation						
Count						
Type		Computational Power				Total
		Very Low	Low	High	Very High	
System (Fog or Cloud)	Cloud	0	0	5	23	28
	Fog	23	5	0	0	28
Total		**23**	**5**	**5**	**23**	**56**

Table 4.33 shows the approach for calculating the expected value from the row total of computational power and column total of type of system (Fog or Cloud) also the total number of observations is 56.

Table 4.33: Expected Frequency

Calculation of Expected Frequency			
Total of Total Computational Power	**Total Type (Fog or Cloud)**	**Expected Frequency**	**Expected Frequency**
23	28	(23 * 28) / 56	11.50
5	28	(5 * 28) / 56	02.50
5	28	(5 * 28) / 56	02.50
23	28	(23 * 28) / 56	11.50
23	28	(23 * 28) / 56	11.50
5	28	(5 * 28) / 56	02.50
5	28	(5 * 28) / 56	02.50
23	28	(23 * 28) / 56	11.50

Table 4.34: χ^2 Calculation

Observed and Expected Frequency for the calculation of χ^2			
Observed Frequency (OF)	**Expected Frequency (EF)**	**(OF - EF)2**	**(OF - EF)2 / EF**
0	11.5	132.25	11.50
0	2.5	6.25	02.50
5	2.5	6.25	02.50
23	11.5	132.25	11.50
23	11.5	132.25	11.50
5	2.5	6.25	02.50
0	2.5	6.25	02.50
0	11.5	132.25	11.50
		Total (∑)	**56.00**

Degree of Freedom =(r-1) * (c-1)

= (2-1) * (4-1)

= 3

Table value @ 5% level of significance = 7.81

Therefore,

The calculated value of Chi-Square is found to be 56.00

The tabulated value of Chi-Square is found to be 7.81

Accordingly, table 4.34 represents the calculation of Chi-Square test value using the observed and expected frequencies. The results confirm that the calculated value of Chi-Square 56 is greater than the tabulated value of 7.81 at a 5% level of significance. So, it is clear that the null hypothesis is **accepted**.

This concludes that there is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure computational power. A notable contrast in performance, measured by computational power, emerged between the SMART FOG protocol-based and cloud-based systems. The findings reveal that the SMART FOG system exhibited superior performance with notably lower computational power as compared to its cloud-based counterpart.

4.5 Multiple Regression Model

To find the association between energy consumed and several devices, execution time, average loop delay, CPU[92] delay, latency, cost execution, and total network usage multiple regression analysis is being conducted the results of the analysis are shown below in the tables.

The descriptive analysis is shown below in the table

Table 4.35 shows that the dataset constructed from experimental values encompasses comprehensive metrics across fog and cloud computing environments. It includes data points for latency, execution time, energy consumption, power consumption, cost of execution, and total network usage. Each metric is recorded under varying experimental conditions, such as different numbers of tasks and nodes. The dataset is

[92] Central Processing Unit

designed to facilitate thorough analysis and evaluation of system performance and resource utilization in both fog and cloud computing scenarios. Utilizing 10-fold cross-validation ensures rigorous testing and validation of models trained on this dataset, enhancing reliability and robustness in assessing the effectiveness of computational frameworks in real-world applications. Descriptive and multiple regression analyses conducted using Excel provide valuable insights into relationships between variables in the dataset.

Table 4.35: Descriptive Summary of Various Measures

Descriptive Statistics			
	Mean	**Std. Deviation**	**N**
Energy Consumed	2906053.09	220658.21	28
No. of Areas	3.57	00002.54	28
Number of Cameras Per Area	6.21	00002.89	28
Execution Time	2686.17	5335.31	28
Average Loop Delay: Motion Object Detector	197.06	0255.13	28
Average Loop Delay: Object Tracker, PTZ[93] Control	065.26	0050.46	28
CPU Delay: Motion Video Stream	001.61	0001.65	28
CPU Delay: Detected Object	000.15	0000.09	28
CPU Delay: Object Location	011.93	0059.39	28
CPU Delay: Camera	002.10	0	28
Latency	276.03	0283.45	28
Cost of execution	325306.73	315146.58	28
Total network usage	466288.21	455181.89	28

The mean value of energy consumed is found to be 2906053.0944 and the standard deviation is found to be 220658.21578.

Table 4.36 shows the energy consumed is considered a dependent variable and No. of Area, Number of Cameras Per Area, Execution Time, Average Loop Delay: Motion Detector, Object Detector, Object Tracker, Average Loop Delay: Object Tracker, PTZ Control, CPU Delay: Motion Video Stream, CPU Delay: Detected Object, CPU Delay: Object Location, CPU Delay: Camera, Latency, Cost of execution and Total network usage are the independent variables.

[93]Pan-Tilt-Zoom

Table 4.36: Variables Considered & Removed

Variables Entered/ Removed [a]			
Model	**Variables Entered**	**Variables Removed**	**Method**
1	Total network usage, Execution Time, CPU Delay: Detected Object, No. of Areas, Average Loop Delay: Object Tracker, PTZ Control, CPU Delay: Motion Video Stream, Number of Cameras Per Area, Latency, Cost of execution, CPU Delay: Object Location [b]	. Time	Enter
a. Dependent Variable: Energy Consumed			
b. Tolerance = .00 limit reached.			

Table 4.37, shows the developed model is shown below in the table which confirms there is a strong correlation between the dependent and independent variables as the calculated R-Square value is 0.99.

Table 4.37: Regression Model Summary

Model Summary [b]									
Model	**R**	**R Square**	**Adjusted R Square**	**Std. Error of the Estimate**	**Change Statistics**				
					R Square Change	**F Change**	**df1**	**df2**	**Sig. F Change**
1	.99[a]	.99	.99	9579.24	.99	1430.95	10	17	.00
a. Predictors: (Constant), Total network usage, Execution Time, CPU Delay: Detected Object, No. of Areas, Average Loop Delay: Object Tracker, PTZ Control, CPU Delay: Motion Video Stream, Number of Cameras Per Area, Latency, Cost of execution, CPU Delay: Object Location									
b. Dependent Variable: Energy Consumed									

The statistical analysis of the model shows high goodness-of-fit measures, indicating a strong relationship between the dependent variable and the independent variables. The coefficient of determination R Square is 0.99, indicating that approximately 99.9% of the variability in the dependent variable can be explained by the independent variables in the model. The adjusted R Square, which accounts for the number of predictors in the model, is 0.99, suggesting that the model is a good fit and not overfitting the data. The standard error of the estimate is 9579.24, indicating the average difference between the observed values and the predicted values by the model.

Table 4.38: ANOVA Statistics

ANOVA[a]						
Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1e	Regression	1313071347215.98	10	131307134721.59	1430.95	.00[b]
	Residual	0001559953982.11	17	000091761998.94	1430.91	.00[b]
	Total	1314631301198.09	27	0131398896720.55	1430.95	.00[b]
a. Dependent Variable: Energy Consumed						
b. Predictors: (Constant), Total network usage, Execution Time, CPU Delay: Detected Object, No. of Areas, Average Loop Delay: Object Tracker, PTZ Control, CPU Delay: Motion Video Stream, Number of Cameras Per Area, Latency, Cost of execution, CPU Delay: Object Location						

Table 4.38 shows the ANOVA Statistics table presents the results of the analysis of variance for the model. The mean square for the model is 131,307,134,721.598, which represents the variance explained by the independent variables in the model. The F-statistic is 1430.953, indicating that the variance explained by the model is significantly greater than what would be expected by chance alone. The p-value (Sig. = .000) is less than the typical significance level of 0.05, indicating that the model's overall effect is statistically significant.

Model 1

The model finds the association between energy consumed and number of areas, number of cameras per area, execution time, average loop delay, CPU delay, latency, cost of execution, and total network usage as shown below in the coefficient table and model summary table.

Energy Consumed ⟶ No. of Areas

Energy Consumed ⟶ No. of Cameras per Area

Energy Consumed ⟶ Execution Time

Energy Consumed ⟶ Average Loop Delay

Energy Consumed ⟶ CPU Delay

Energy Consumed ⟶ Latency

Energy Consumed ⟶ Cost of Execution

Energy Consumed ⟶ Total Network Usage

Table 4.39 shows that the coefficient Values represent the impact of each independent variable on the dependent variable (Energy Consumed). Among the predictors, "Average Loop Delay: Object Tracker, PTZ Control" and "Total network usage" exhibit the most substantial influence, with positive coefficients indicating a positive relationship with energy consumption. Conversely, "Execution Time" and "Latency" demonstrate significant but negative coefficients, suggesting that higher values of these variables are associated with lower energy consumption. Other predictors show relatively weaker associations with energy consumption.

Table 4.39: Coefficient Values

Coefficients[a]						
Model		**Unstandardized Coefficients**		**Standardized Coefficients**	**t**	**Sig.**
		B	**Std. Error**	**Beta**		
1	(Constant)	2647256.69	9801.57	0.007	270.08	0.00
	No. of Areas (X1)	1161.67	2503.32	0.013	.46	0.64
	Number of Cameras Per Area (X2)	-164.37	1693.19	-0.002	-.09	0.92
	Execution Time (X3	4.02	1.58	0.09	2.53	0.02
	Average Loop Delay Object Tracker, PTZ Control (X4)	731.35	86.66	0.16	8.43	0.00
	CPU Delay: Motion Video Stream (X5)	3779.44	2591.37	0.02	1.45	0.16
	CPU Delay: Detected Object (X6)	7324.10	26165.35	0.01	0.28	0.78
	CPU Delay: Object Location (X7)	-288.19	142.47	-0.07	-2.02	0.06
	Latency (X8)	-87.49	26.41	-0.11	-3.31	0.01
	Cost of execution (X9)	0.01	0.02	0.02	0.72	0.48
	Total network usage (X10)	0.45	0.02	0.94	18.54	0.00
a. Dependent Variable: Energy Consumed (Y)						

The variables Execution Time, Average Loop Delay: Object Tracker, PTZ Control, Latency, and total network usage are found to be significant as the calculated p-value is greater than the standard alpha value of 0.05.

Table 4.40: Excluded Measures

Excluded Variables [a]						
Model		Beta In	t	Sig.	Partial Correlation	Collinearity Statistics
						Tolerance
1	Average Loop Delay: Motion Detector, Object Detector, Object Tracker	b	-	-	-	.00
a. Dependent Variable: Energy Consumed						
b. Predictors in the Model: (Constant), Total network usage, Execution Time, CPU Delay: Detected Object, No. of Areas, Average Loop Delay: Object Tracker, PTZ Control, CPU Delay: Motion Video Stream, Number of Cameras Per Area, Latency, Cost of execution, CPU Delay: Object Location						

Table 4.40 shows that collinearity statistics section shows a tolerance value of 0.00 for the "Average Loop Delay" variable. A tolerance value of 0 indicates that there is perfect collinearity between this independent variable and other variables in the model. This suggests a high degree of correlation between "Average Loop Delay" and other predictors, which may lead to multicollinearity issues.

Table 4.41: Residual Statistics of Model

Residuals Statistics [a]					
	Minimum	Maximum	Mean	Std. Deviation	N
Predicted Value	2659408.50	3199812.25	2906053.09	220527.25	28
Residual	-22652.21	11547.76	.00	7601.05	28
Std. Predicted Value	-1.11	1.33	.00	1.00	28
Std. Residual	-2.36	1.21	.00	0.79	28
a. Dependent Variable: Energy Consumed					

Table 4.41 shows that Overall residual statistics provide an understanding of the accuracy and variability of the predictions for the "Energy Consumed" dependent variable in the model. The model seems to have a reasonably accurate prediction with minor variations between observed and predicted values.

The mathematical representation of the model:

Y (Energy Consumption) = 1161.675X1 (No. of Areas) -164.373 X2 (Number of Cameras Per Area) + 4.023 X3 (Execution Time) + 731.359 X4 (Average Loop Delay: Object Tracker, PTZ Control) + 3779.441 X4 (CPU Delay: Motion Video Stream) +7324.104X5 (CPU Delay: Detected Object) - 288.190 X6 (CPU Delay: Object Location)-87.494 X7 (Latency) + 0.015 X8 (Cost of execution) +0.456 X9 (Total network usage)

4.6 Use of Machine Learning Approaches in Task Scheduling

Machine learning approaches are playing an increasingly vital role in task scheduling, revolutionizing the efficiency and performance of task allocation and resource management in cloud computing, edge computing, and IoT environments. These techniques offer the ability to predict and forecast task demands, enabling proactive resource allocation and reducing bottlenecks. Dynamic task scheduling becomes possible with real-time data analysis, ensuring agile adaptations to changing conditions. Load balancing benefits from machine learning's insights to distribute tasks optimally across resources. Task prioritization becomes smarter, and energy efficiency is enhanced by choosing energy-conscious resources. Multi-objective optimization enables simultaneous consideration of conflicting objectives, and learning from user behaviour facilitates personalized task scheduling. In essence, the integration of machine learning in task scheduling empowers intelligent, adaptive, and efficient resource allocation, leading to superior system performance, minimized response times, and optimal resource utilization across diverse computing environments. Mainly in supervised learning classification-based algorithms were being used for task scheduling. The algorithms being considered for task scheduling were Logistic Regression, IBK, K-Star, and AdaBoostM1.

Experiment 1: Number of Tasks: 40 and Nodes: 4

In Experiment 1, the number of tasks was set to 40, and the number of nodes was set to 4. The evaluation of the model was performed using 10-fold cross-validation, a common technique to assess the performance of machine learning algorithms. In this approach, the dataset is divided into 10 subsets, and the model is trained and tested 10 times, each time using a different subset as the test set and the remaining subsets as the training set.

4.6.1 Logistic Regression

Table 4.42, shows that the evaluation of logistic regression through 10-fold cross-validation, performance measures provide valuable insights into the model's classification accuracy and predictive capabilities. Accuracy, precision, recall (sensitivity), and F1 score offer comprehensive assessments of the model's correctness in classifying instances and its ability to avoid false positives and negatives.

Table 4.42: Performance Measures for Logistic Regression (LR[94]) at 10-fold Cross-Validation

Measures	Values
Correctly Classified Instances	176 (88%)
Incorrectly Classified Instances	24 (12%)
Kappa statistic	0.83
Mean absolute error	0.0599
Root mean squared error	0.2353
Relative absolute error	16.64%
Root relative squared error	55.47%
Total Number of Instances	200
Time taken to build a model:	0.01 seconds

Table 4.43 Detailed Accuracy by Class: Accuracy class-wise for the LR classifier refers to the accuracy of the model in classifying instances within each individual class. It provides insights into how well the model performs for each specific class in the classification task.

Table 4.43: Accuracy Class Wise (LR Classifier)

Sr. No.	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC[95]	ROC[96] Area	PRC[97] Area	Class
1	0.95	0.04	0.93	0.95	0.94	0.91	0.99	0.99	Node1
2	0.50	0.02	0.83	0.50	0.63	0.59	0.96	0.78	Node2
3	1	0.09	0.72	1	0.84	0.81	0.99	0.99	Node3
4	1	0	1	1	1	1	1	1	Node4
Wt. Avg.	0.88	0.04	0.89	0.88	0.88	0.84	0.98	0.95	

Table 4.44 shows Confusion Matrix: The confusion matrix provides a detailed and clear evaluation of the model's accuracy and misclassification patterns for each class, offering valuable insights into the model's classification capabilities for the given dataset.

[94]Logistic Regression
[95]Matthews Correlation Coefficient
[96]Receiver Operating Characteristic
[97]Precision-Recall Curve

Table 4.44: Confusion Matrix (LR)

a b c d ← classified as
76 4 0 0 | a = Node1
5 20 15 0 | b = Node2
0 0 40 0 | c = Node3
0 0 0 40 | d = Node4

It was found that in case of logistic regression, the correctly classified instances were about 88% which was quite higher than the considered classification techniques such as IBK and AdaBoostM1. Similarly, the precision, recall, and F-measure values of 0.89, 0.88, and 0.88 respectively and the FP rate value 0.04.

4.6.2 IBK (Stratified Cross-Validation: 10-fold)

The performance of IBK classification algorithm at configuration setting: stratified 10-fold cross-validation. Accordingly, the performance measures included are correctly classified instances, incorrectly classified instances, kappa statistic, mean absolute error, root mean squared error, relative absolute error, root relative squared error, total number of instances, and time taken to build a model.

Table 4.45: Performance Measures for IBK at 10-fold Cross-Validation

Measures	Values
Correctly Classified Instances	117 (58.5%)
Incorrectly Classified Instances	83 (41.5%)
Kappa statistic	0.39
Mean absolute error	0.21
Root mean squared error	0.45
Relative absolute error	58.65%
Root relative squared error	106.44%
Total Number of Instances	200
Time taken to build model:	0.001 seconds

Table 4.45 IBK model was built using 10-fold cross-validation on a dataset containing a total of 200 instances. The time taken to build the model was 0.001 seconds, indicating the model's efficiency in training.

Detailed Accuracy by Class

Based on table 4.46 which shows accuracy class-wise shown below it can be concluded that Node 3 and Node 4 have shown higher precision as compared to other two nodes. Similarly, the recall value is found to be higher in case of Node 4 and Node 1 with values 1, and 0.95 respectively.

Table 4.46: Accuracy Class Wise (IBK)

Sr. No.	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1	0.95	0.35	0.64	0.95	0.77	0.59	0.66	0.63	Node1
2	0	0.26	0	0	0	-0.25	0.09	0.16	Node2
3	0.03	0	1	0.02	0.05	0.14	0.58	0.24	Node3
4	1	0	1	1	1	1	1	1	Node4
Wt.Avg.	0.59	0.19	0.65	0.59	0.52	0.41	0.60	0.53	

Table 4.47 shows Confusion Matrix: The confusion matrix for the IBK (Instance-Based k-nearest Neighbor) model shows its performance in classifying instances into different classes (Node1, Node2, Node3, and Node4). It reveals that Node1 has 76 true positives and 4 false positives, while Node2 has all 40 instances misclassified as Node1 (false negatives). Node3 has 1 true positive, 2 false positives, and 37 false negatives, and Node4 has all 40 instances correctly classified as true positives. The matrix provides a comprehensive evaluation of the model's accuracy and misclassification patterns for each class, offering valuable insights into its classification capabilities using the IBK algorithm.

Table 4.47: Confusion Matrix (IBK)

a b c d ← classified as
76 4 0 0 | a = Node1
40 0 0 0 | b = Node2
2 37 1 0 | c = Node3
0 0 0 40 | d = Node4

Accordingly, it was found that in case of IBK, the correctly classified instances were about 58.5% which is quite less showing low level of accuracy as compared with other classification techniques such as Logistic Regression, K-Star, and AdaBoostM1. Similarly, the precision, recall, and F-measure values of 0.65, 0.58, and 0.51

respectively were lower in comparison to other classifiers being considered also the mean absolute error value was found to be 0.21, and FP rate value 0.19.

4.6.3 K-Star (Stratified Cross-Validation: 10-fold)

The performance measures for the K-Star model at 10-fold cross-validation provide valuable insights into its classification accuracy and predictive capabilities. Common metrics such as accuracy, precision, recall (sensitivity), and F1 score offer a comprehensive assessment of the model's correctness in predicting class labels and its ability to avoid false positives and negatives.

Table 4.48: Performance Measures for K-Star at 10-fold Cross-Validation

Measures	Values
Correctly Classified Instances	182(91%)
Incorrectly Classified Instances	18 (9%)
Overall Accuracy	91%
Kappa statistic	0.87
Mean absolute error	0.04
Root mean squared error	0.19
Relative absolute error	13.54%
Root relative squared error	44.24%
Total Number of Instances	200
Time taken to build model:	0.001 seconds

Table 4.48 shows K-Star model was built using 10-fold cross-validation on a dataset containing a total of 200 instances. The time taken to build the model was 0.001 seconds, indicating the model's efficiency in training.

Detailed Accuracy by Class

Based on the table 4.49 accuracy class-wise shown below it can be concluded that Node 2, Node 3, and Node 4 have shown higher precision as compared to Node 1. Similarly, the recall value is found to be higher in case of Node 1 and Node 2 with values 1 and, 1 respectively.

Table 4.49: Accuracy Class Wise (K-Star)

S. No.	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1	1	0.15	0.81	1	0.90	0.83	1	1	Node1
2	0.575	0	1	0.57	0.73	0.72	1	1	Node2
3	0.975	0	1	0.97	0.98	0.98	1	1	Node3
4	1	0	1	1	1	1	1	1	Node4
Wt. Avg.	0.91	0.06	0.92	0.91	0.90	0.87	1	1	

Table 4.50 shows Confusion Matrix: The K-Star model's confusion matrix shows excellent performance in correctly classifying instances into their respective classes, particularly for Node4, with all 40 instances correctly classified. It has minimal misclassifications for Node1 and Node3. However, there are 17 misclassifications for Node2, where 17 instances were classified as Node1 instead.

Table 4.50: Confusion Matrix (K-Star)

a b c d ← classified as
80 0 0 0 | a = Node1
17 23 0 0 | b = Node2
1 0 39 0 | c = Node3
0 0 0 40 | d = Node4

It was found that in case of K-Star classifier being used for task scheduling correctly classified instances were about 91% which was quite higher than the considered classification techniques such as IBK, Logistic Regression, and AdaBoostM1. Similarly, the precision, recall, and F-measure values of 0.927, 0.91, and 0.903 respectively were higher in comparison to IBK, Logistic Regression, and AdaBoostM1 also the mean absolute error value was found to be 0.05 and FP rate value 0.04.

4.6.4 AdaBoostM1 (Stratified Cross-Validation: 10-fold)

AdaBoostM1 is an ensemble learning method based on table 4.51, AdaBoost algorithm, and stratified 10-fold cross-validation is a popular technique used to evaluate its performance. In this evaluation, the dataset is divided into ten subsets, ensuring that each subset has a similar distribution of classes as the original dataset. The AdaBoostM1 model is trained and tested ten times, each time using a different subset as the test set and the remaining nine subsets as the training set.

Table 4.51: Performance Measures forAdaBoostM1 at 10-fold Cross-Validation

Measures	Values
Correctly Classified Instances	120(60%)
Incorrectly Classified Instances	80 (40%)
Kappa statistic	0.41
Mean absolute error	0.32
Root mean squared error	0.38
Relative absolute error	88.46%
Root relative squared error	88.54%
Total Number of Instances	200
Time taken to build a model:	0.03 seconds

The AdaBoostM1 model was built using 10-fold cross-validation on a dataset containing a total of 200 instances. The time taken to build the model was 0.03 seconds which is higher than IBK and K-Star.

Detailed Accuracy by Class

Based on table 4.52 shows accuracy class-wise below it can be concluded that Node 1, Node 2, Node 3, and Node 4 have shown higher precision with value 1. Similarly, the recall value is found to be higher in case of Node 1 and Node 2 with values 1 and, 1 respectively.

Table 4.52: Accuracy Class Wise (AdaBoostM1)

S. No.	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1	1	0.33	0.66	1	0.8	0.66	1	1	Node1
2	1	0.25	0.5	1	0.66	0.61	1	1	Node2
3	0	0	-	0	-	-	1	1	Node3
4	0	0	-	0	-	-	1	1	Node4
Weighted Avg.	0.60	0.18	-	0.6	-	-	1	1	

Table 4.53 shows Confusion Matrix: The confusion matrix for the AdaBoostM1 model shows perfect performance in correctly classifying instances into their respective classes, with 80 instances correctly classified as Node1, 40 instances as Node2, 40 instances as Node3, and 40 instances as Node4. There are no misclassifications observed in the model's predictions for any of the classes.

Table 4.53: Confusion Matrix (AdaBoostM1)

ab c d ← classified as
80 0 0 0 | a = Node1
0 40 0 0 | b = Node2
40 0 0 0 | c = Node3
0 40 0 0 | d = Node4

Accordingly, it was found that in case of AdaBoostM1 the correctly classified instances were about 60% which is quite less showing low level of accuracy as compared with other classification techniques such as Logistic Regression and K-Star. Similarly, the precision, recall, and F-measure were lower in comparison to other classifiers such as Logistic Regression and K-Star and FP rate value 0.18.

4.6.5 Comparative Analysis of Classification Algorithms

In the performance-wise analysis of classification algorithms using 10-fold cross-validation with 40 tasks and 4 nodes, various performance metrics were evaluated to assess the effectiveness of the algorithms in classifying instances.

Experiment 1: Number of Tasks: 40 and Nodes: 4:

In Experiment 1, the number of tasks was set to 40, and the number of nodes was set to 4. The evaluation of the model was performed using 10-fold cross-validation, a common technique to assess the performance of machine learning algorithms. In this approach, the dataset is divided into 10 subsets, and the model is trained and tested 10 times, each time using a different subset as the test set and the remaining subsets as the training set.

Table 4.54: Performance-Wise Analysis of Classification Algorithms (10 folds, Number of Tasks: 40 and Nodes: 4)

Performance Measure	Logistic Regression	K-Star	IBK	AdaBoostM1
Accuracy	0.88	0.91	0.58	0.60
Precision	0.88	0.92	0.65	-
Recall	0.88	0.91	0.58	0.60
F-Measure	0.87	0.90	0.51	-
ROC Area	0.98	1.00	0.60	1.00
Mean absolute error	0.05	0.04	0.21	0.32
Execution Time Model	15ms	10ms	10ms	30ms

Based on table 4.54, which provided performance measures, K-Star appears to be the best-performing algorithm, achieving the highest accuracy and precision among the four. Logistic Regression also shows respectable performance with high accuracy and precision. On the other hand, IBK and AdaBoostM1 have lower accuracy scores, making them less suitable choices for the given classification tasks.

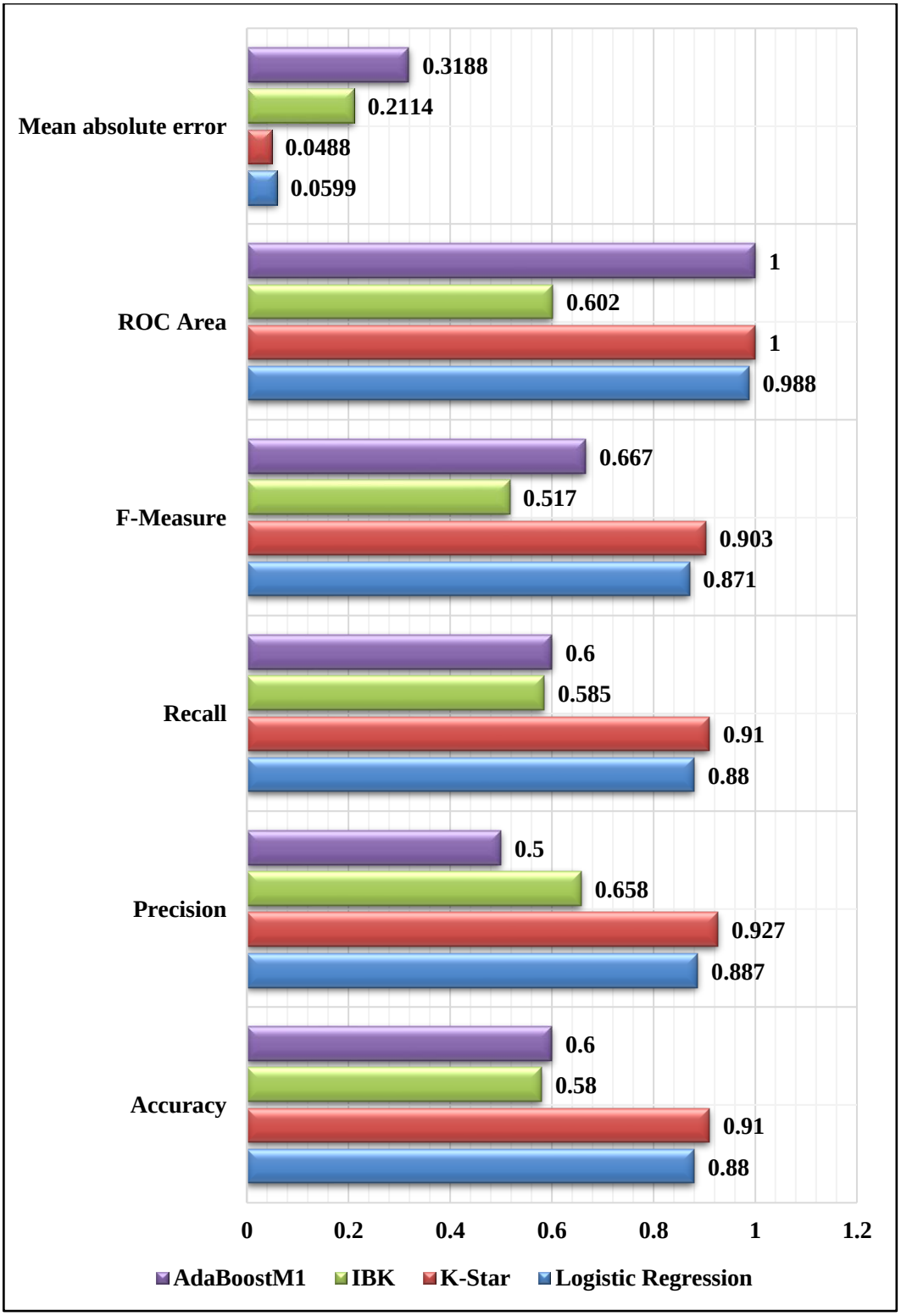

Figure 4.8: Evaluation of Classifier at 10-fold Cross-Validation based on Various Performance Measures

From the above Figure 4.8, it is clear that Logistic Regression and K-star are the most appropriate algorithms for task scheduling while considering the configuration setting; cross-validation 10 folds.

Cross-validation – 25-fold: In the performance-wise analysis of classification algorithms for task allocation and resource management in an IoT environment with 40 tasks and 4 nodes, using 25-fold cross-validation, the evaluation provides a comprehensive understanding of the effectiveness of different algorithms in this specific scenario.

Table 4.55: Performance-Wise Analysis of Classification Algorithms (25 folds, Number of Tasks: 40 and Nodes: 4)

Performance Measure	Logistic Regression	K-Star	IBK	AdaBoostM1
Accuracy	0.89	0.92	0.64	0.52
Precision	0.91	0.94	0.68	0.46
Recall	0.89	0.93	0.64	0.53
F-Measure	0.89	0.92	0.56	0.46
ROC Area	0.99	1.00	0.48	0.95
Mean absolute error	0.05	0.04	0.18	0.32
Execution Time Model Building	15ms	10ms	10ms	35ms

Table 4.55 shows the 25-fold cross-validation involves dividing the dataset of 40 tasks into 25 equal subsets (folds). Each classification algorithm is trained on 24 folds and then tested on the remaining fold. This process is repeated 25 times, with each fold serving as the testing set once.

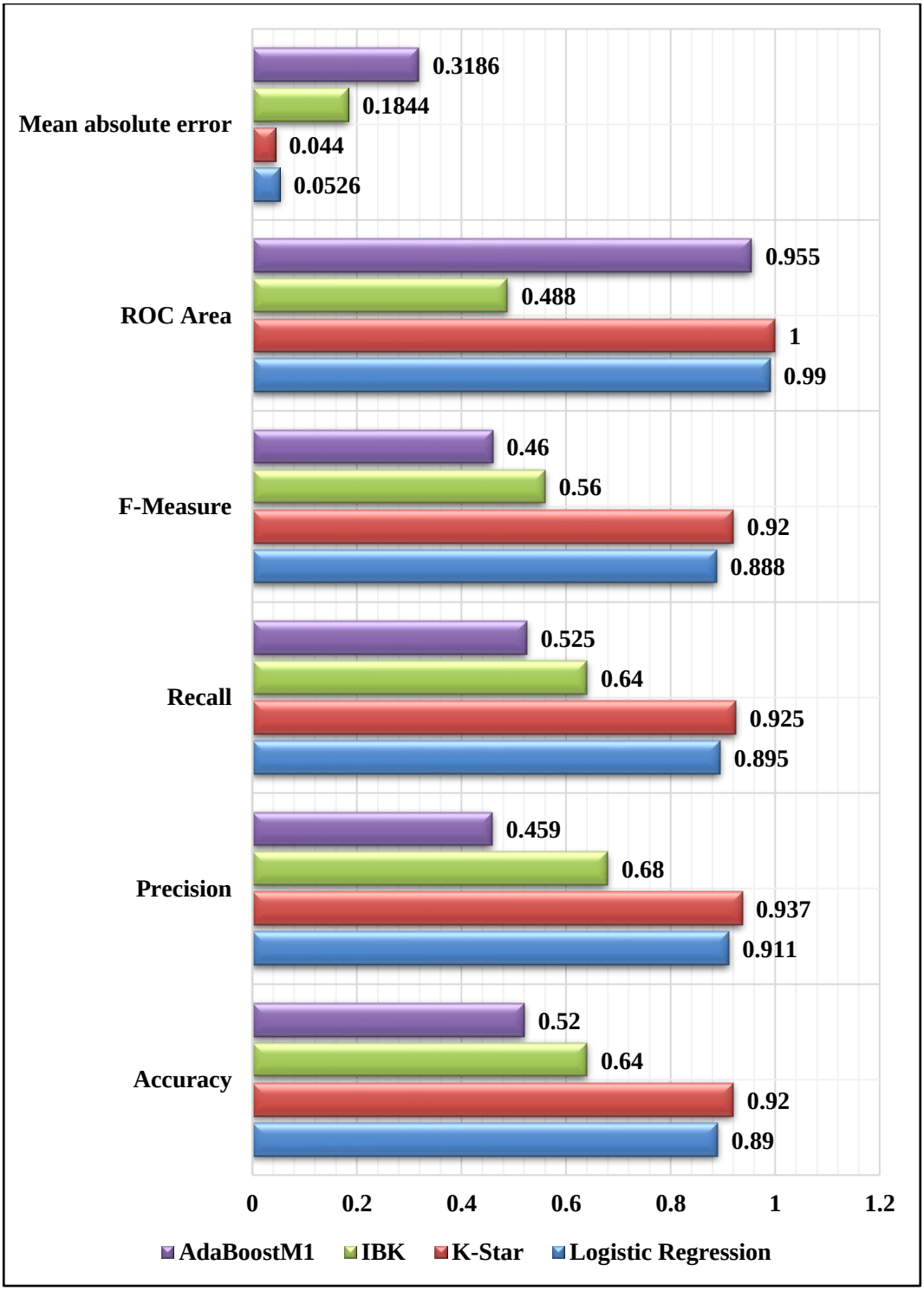

Figure 4.9: Evaluation of Classifier at 25-fold Cross-Validation based on various Performance Measures

From the above Figure 4.9, it is clear that Logistic Regression and K-star are the most appropriate algorithms for task scheduling with a mean absolute error of 0.044 while considering the configuration setting; cross-validation 25 folds.

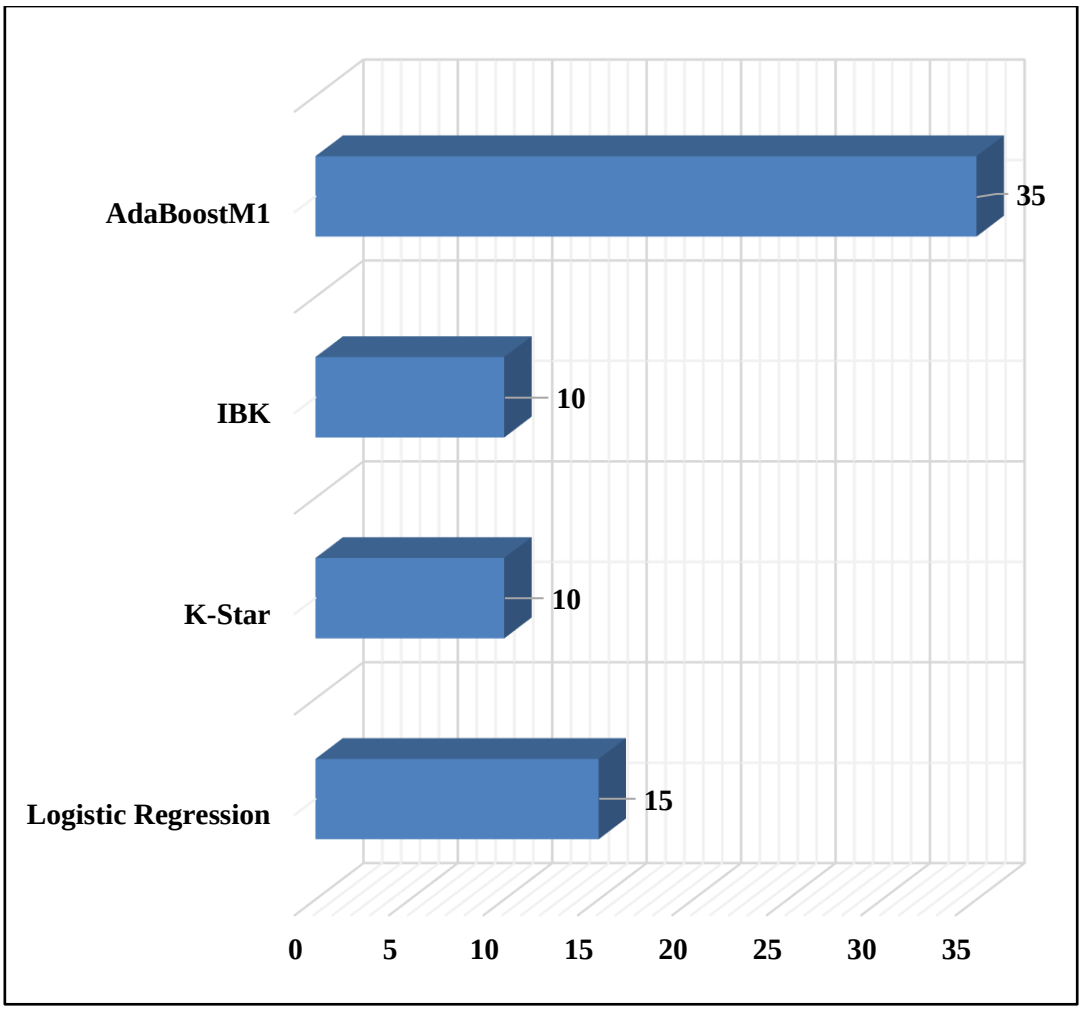

Figure 4.10: Average Execution Time (ms): 25 folds

From Figure 4.10, the average execution time of the most appropriate algorithms is found to be IBK, K-Star, and Logistic Regression while considering 40 tasks and 4 nodes and cross-validation 25 folds. These three algorithms as the most appropriate ones are based on their ability to achieve satisfactory classification performance while offering faster average execution times. The 25-fold cross-validation ensures a robust evaluation of the algorithms' performance, considering different subsets of the data for training and testing.

By considering execution time as an important criterion, the analysis aims to select algorithms that can handle task allocation and resource management efficiently in real-time IoT environments with 40 tasks and 4 nodes.

Experiment 2: Number of Tasks: 160 and Nodes: 4

Cross-validation – 10 folds: The performance-wise analysis of classification algorithms for task allocation and resource management in an IoT environment with 10-fold cross-validation, 160 tasks, and 4 nodes provides valuable insights into the effectiveness of different algorithms in this specific scenario. Using the 10-fold cross-validation, the dataset of 160 tasks is divided into ten equal subsets (folds).

Table 4.56: Performance-Wise Analysis of Classification Algorithms (10 folds,160number of tasks and Nodes: 4)

Performance Measure	Logistic Regression	K-Star	IBK	AdaBoostM1
Accuracy	0.81	0.90	0.25	0.50
Precision	0.83	0.91	0.26	-
Recall	0.81	0.90	0.26	0.50
F-Measure	0.82	0.90	0.26	-
ROC Area	0.95	0.96	0.50	0.83
Mean absolute error	0.09	0.07	0.37	0.25
Execution Time Model Building	1660ms	20ms	20ms	25ms

Each algorithm is trained on 10 folds and tested on the remaining fold. This process is repeated ten times, with each fold serving as the testing set once and the results are shown above in table 4.56.

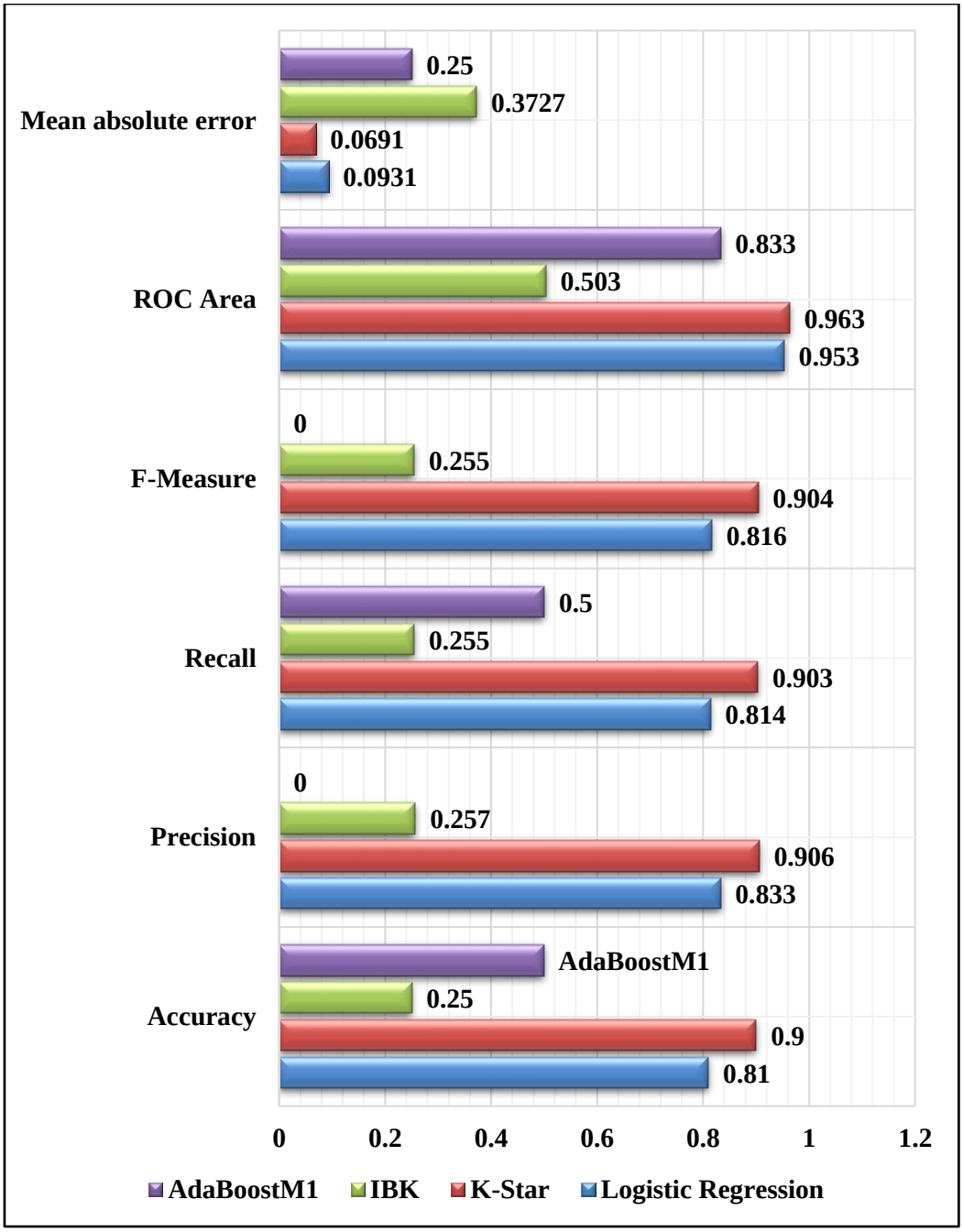

Figure 4.11: Evaluation of Classifier at 10-fold Cross-Validation Based on Various Performance Measures (Number of Tasks: 160 and Nodes: 4)

Figure 4.11, it is clear that Logistic Regression and K-star are the most appropriate algorithms for task scheduling while considering the configuration setting; cross-validation 10 folds and 160 tasks and 4 nodes.

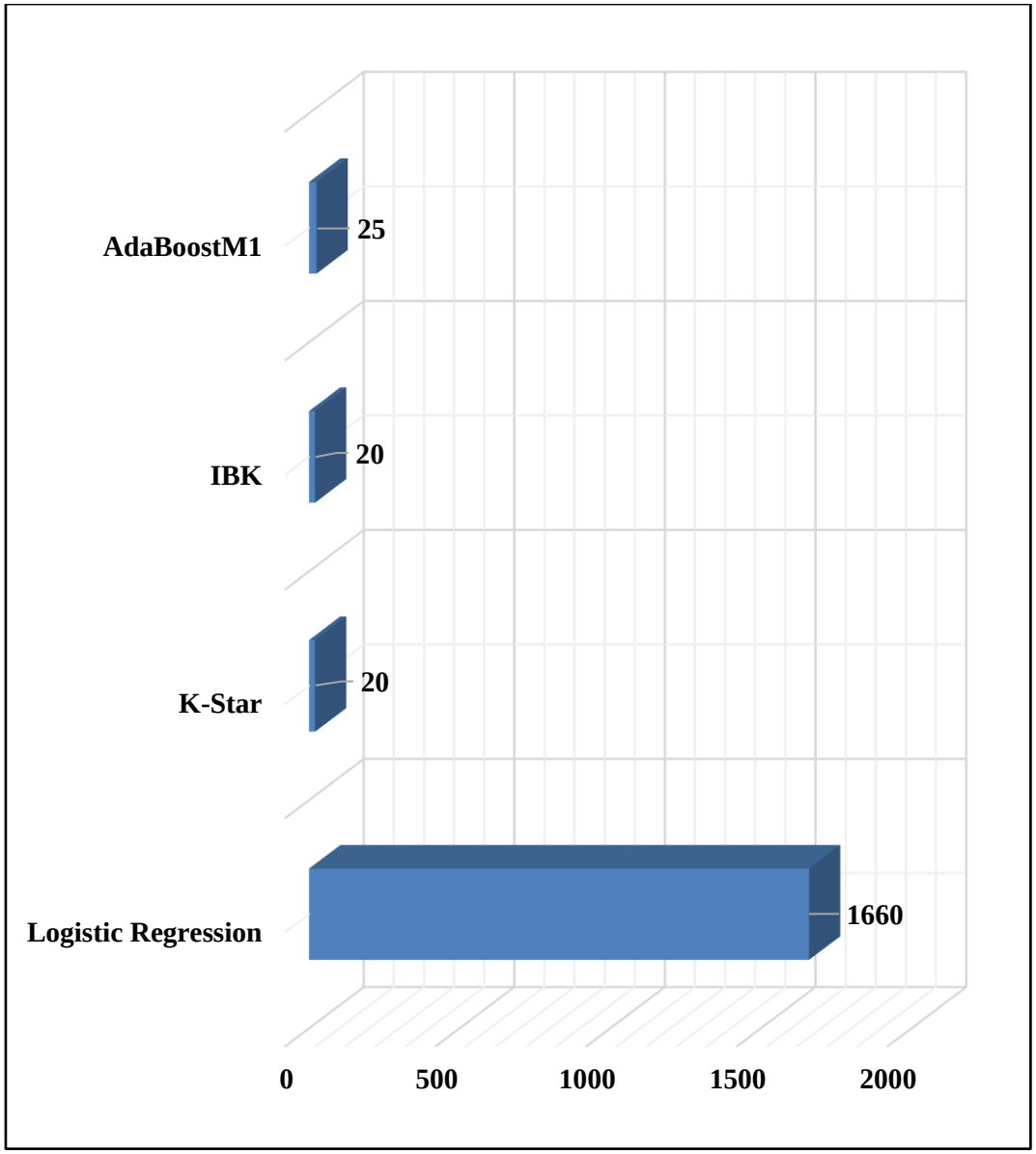

Figure 4.12: Average Execution Time (ms): 10 folds (Number of Tasks: 160 and Nodes: 4)

From the Figure 4.12, for average execution time, the most appropriate algorithms are found to be IBK and K-Star while considering 160 number of tasks and 4 nodes and cross validation 10 folds. In a Fog Computing environment with 160 tasks across 4 nodes and using 10-fold cross-validation, IBK, and K-Star algorithms are identified as optimal based on average execution time known for efficiency in classification tasks, both algorithms demonstrate effective task processing and classification with relatively low execution times, making them suitable choices for distributed Fog Computing scenarios.

Cross-validation - 25 folds: In Table 4.34, the performance-wise analysis of classification algorithms is presented using 25-fold cross-validation with 160 tasks and 4 nodes.

Table 4.57: Performance-Wise Analysis of Classification Algorithms (25 folds, 160 number of tasks and Nodes: 4)

Performance Measure	Logistic Regression	K-Star	IBK	AdaBoostM1
Accuracy	0.89	0.90	0.25	0.47
Precision	0.90	0.91	0.25	0.47
Recall	0.89	0.91	0.25	0.47
F-Measure	0.89	0.91	0.25	0.45
ROC Area	0.98	0.96	0.50	0.80
Mean absolute error	0.05	0.07	0.38	0.29
Execution Time Model Building	1860ms	20ms	20ms	25ms

Table 4.57 shows updated performance measures, K-Star remains the best-performing algorithm, achieving the highest accuracy and precision among the four. Logistic Regression also shows respectable performance with high accuracy and precision scores. However, both IBK and AdaBoostM1 have significantly lower accuracy and precision values, making them less suitable choices for the given classification tasks.

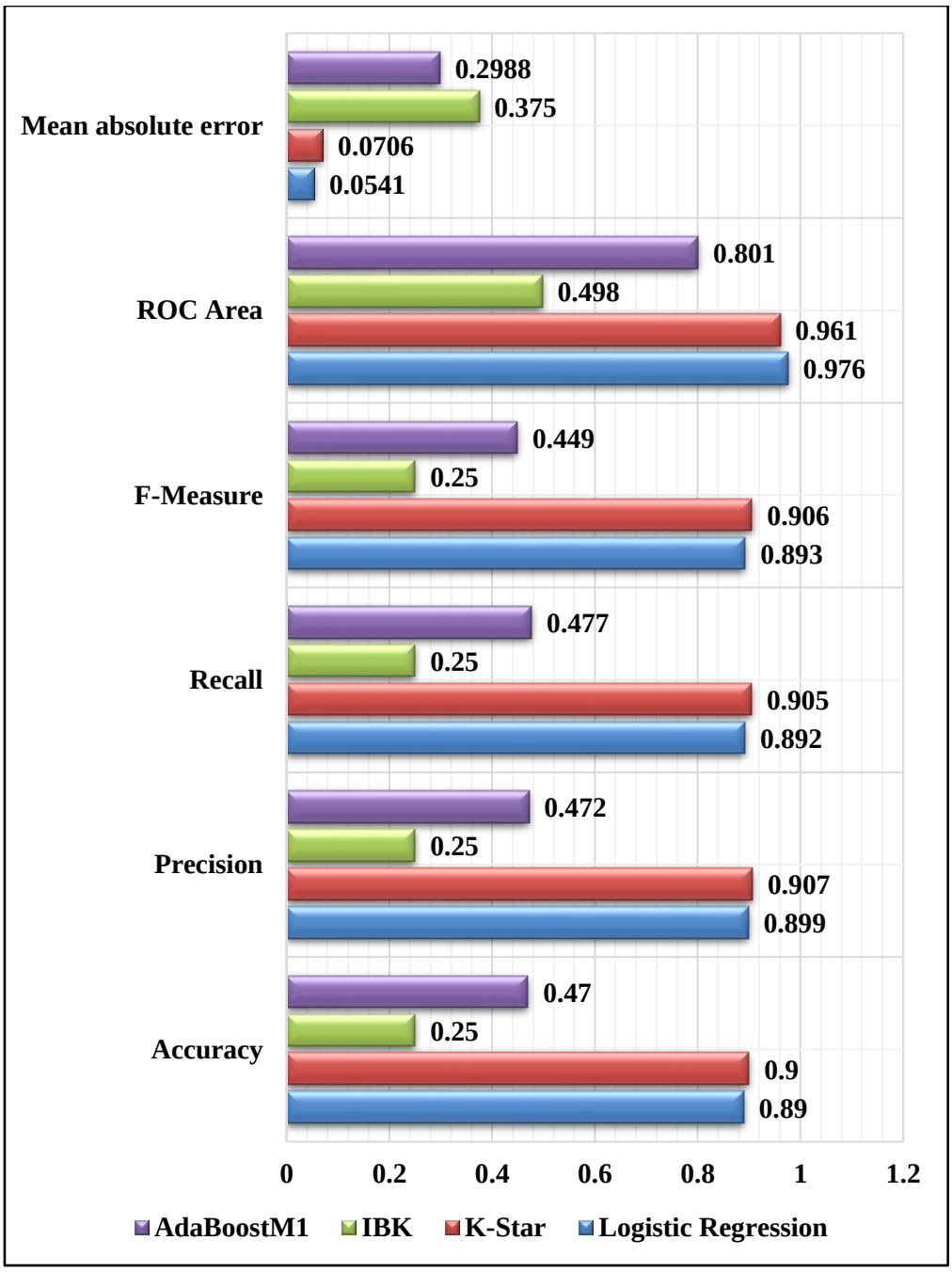

Figure 4.13: Evaluation of classifier at 25-fold Cross-Validation based on Various Performance Measures (Number of Tasks: 160 and Nodes: 4)

Figure 4.13, it is clear that Logistic Regression and K-star are the most appropriate algorithms for task scheduling while considering the configuration setting; cross-validation 25 folds and 160 tasks and 4 nodes.

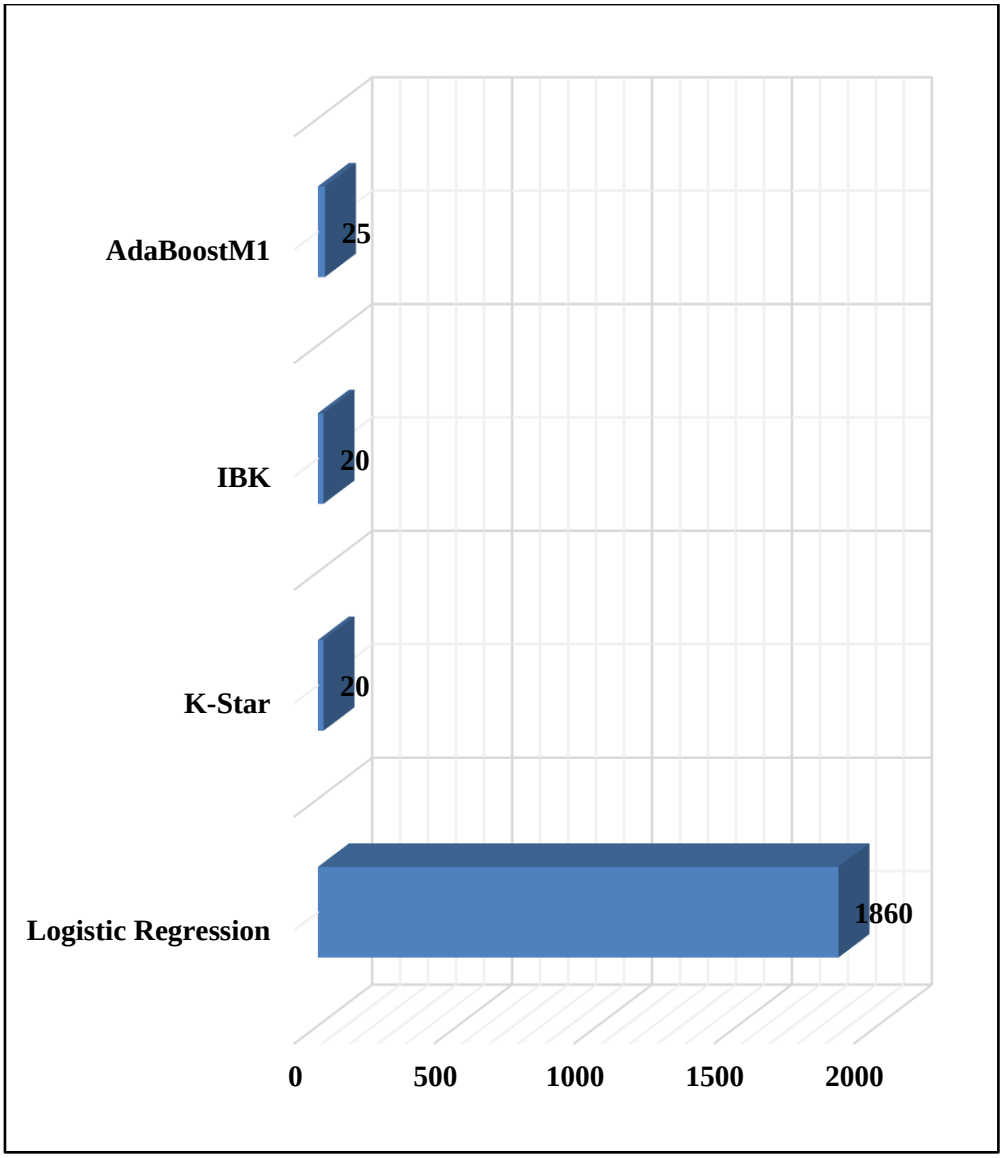

Figure 4.14: Average Execution Time (ms): 25 folds (Number of Tasks: 160 and Nodes: 4)

Based on the above Figure 4.14, average execution time the most appropriate algorithms are found to be K-Star and IBK while considering 160 number of tasks and 4 nodes and cross-validation 25 folds. The consistent performance in minimizing average execution time underscores their suitability for real-time task execution and classification in resource-constrained environments. This reinforces their selection as optimal choices for achieving efficient task processing in Fog Computing systems.

4.7 Clustering Algorithms Used for Task Scheduling

Cloud computing offers several benefits, including immense processing power, ample storage, a massive network connecting processing nodes and data sources, and a pay-per-use approach. Cloud computing is a strong technology that provides these paradigms as well as many other benefits such as flexibility, cheaper costs, scalability, and ease of software installation. However, despite these benefits, Cloud computing has certain disadvantages. Some of the disadvantages include: the client and Cloud layer may be geographically separated, which can cause transmission delays; there may be a scarcity of resources for task execution; many resources may be idle even if tasks must be performed instantly and so on.

Virtualized Fog computing technology is used to solve these issues. Fog is a layer that sits between end users and cloud data centers. Fog computing can be useful for executing applications that require low latency and real-time responses, depending on the location of the data producer. This layer can include a large number of virtual servers to handle incoming requests. "Resource allocation is the systematic approach of allocating available resources to the needed Cloud clients over the Internet," according to Agarwal, Yadav, and Yadav. The timing and order in which resources are allotted are critical for maximizing the benefits of employing a virtual server, since the system's throughput may be increased while customers are not overcharged. The availability of resources should ensure that high-priority jobs do not wind up at the bottom of the task queue. This might result in inefficient utilization of virtual servers and possibly company loss. As a result, allocating resources in a prioritized manner to maximize profit is a critical and promising study topic. Furthermore, ML, an important field, has made significant advances in a variety of academic areas, including robotics, neuromorphic computing, computer graphics, NLP[98], decision-making, and speech recognition. Several researches have been presented to look at ways to use machine learning to solve fog computing issues. In recent years, there has been an increase in the use of ML to improve fog computing applications and deliver fog services, such as efficient resource management, security, latency and energy reduction, and traffic modeling.

[98] Natural Language Processing

There are many different types of fog computing devices, sensors, and objects, and each one generates a large amount of data that must be processed. Real-time processing has the potential to improve efficiency. In some cases, it may be necessary. Sensors, devices, and by sending requests, objects will completely utilize resources. As a result, fog computing requires resource management and should be implemented with caution. In this section, we looked at studies that used ml algorithms to manage fog computing resources. This paper proposes a Scheduling Algorithm which is used to schedule tasks at fog level. A task is scheduled to the VM that plays a role in the execution of request / response model in fog computing. We use a K-means clustering algorithm for scheduling fog devices. The default resource scheduler in the simulator equally divides fog device's resources among all active application modules. Clustering makes it easy to find a set of tasks for VM with minimum cost. Therefore, the integration of ML method i.e. Clustering in scheduling tasks in fog computing will give a better quality of services (QoS) with low execution cost and low network usage. The study includes:

1. Presentation of Clustering Scheduling in Fog Computing.
2. Implementation of proposed algorithm in iFogsim.
3. Reduction of Execution Cost.

Clustering algorithms group data points based on their similarity or proximity. Common types include K-means, which partitions data into K clusters; DBSCAN, which identifies clusters based on density; and Hierarchical clustering, which builds a tree-like structure of nested clusters.

4.7.1 Canopy Clustering

Table 4.58 shows that Canopy Clustering is a pre-processing technique used in data clustering to reduce the computational complexity of subsequent clustering algorithms. It acts as a data summarization step by creating overlapping regions (canopies) that cover subsets of data points based on a similarity threshold. Data points falling within each canopy are then passed to another clustering algorithm for further refinement.

Table 4.58: Accuracy Canopy Clustering

Measures	Values
Correctly Classified Instances	149 (74.5%)
Incorrectly Classified Instances	51 (25.5%)
Overall Accuracy	74.5%
Total Number of Instances	200
Time taken to build a model	0.001 seconds

Figure 4.15, shows that the accuracy results for Canopy Clustering show that the model correctly classified 149 instances, representing 74.5% of the total instances in the dataset. There were 51 instances misclassified, amounting to 25.5% error. The overall accuracy of 74.5% indicates its effectiveness in classifying data points, and the model was built efficiently in just 0.001 seconds for a total of 200 instances.

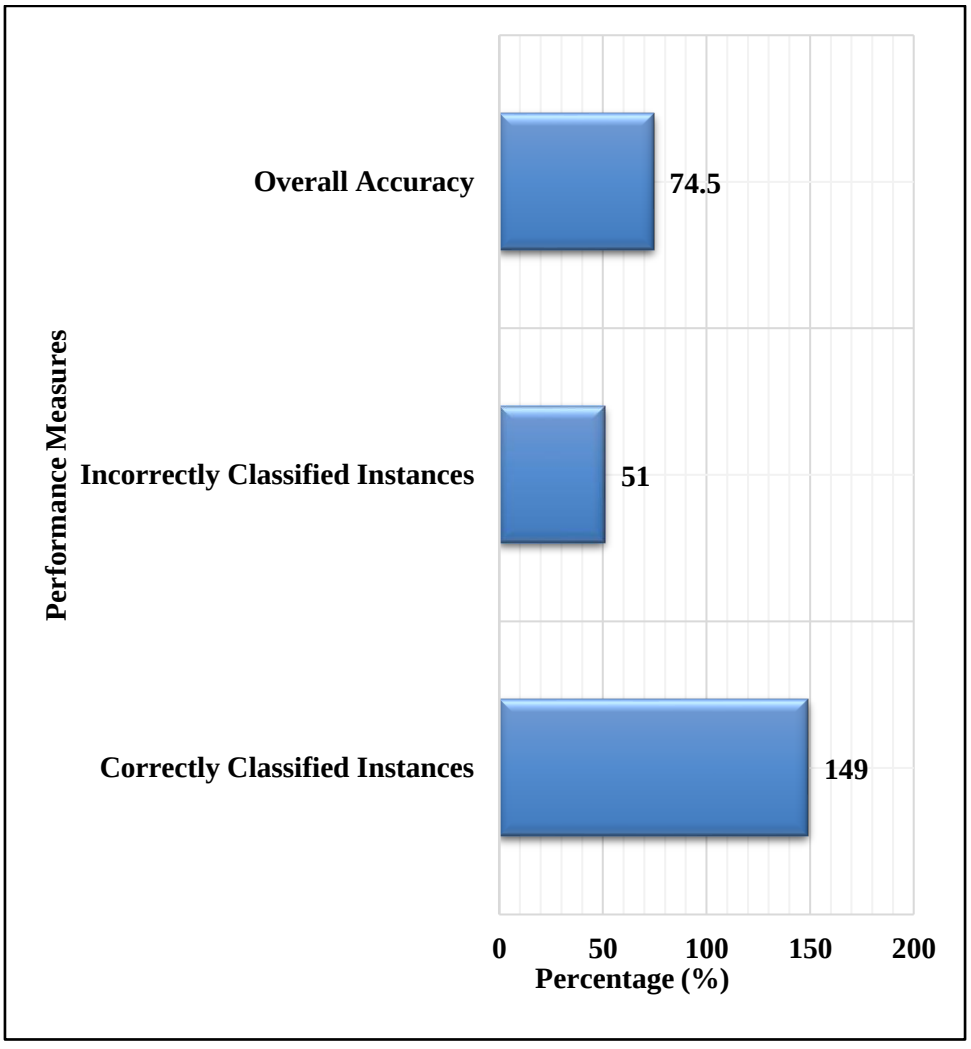

Figure 4.15: Overall Accuracy Canopy Clustering

Accordingly, Figure 4.15, it was found that in case of Canopy Clustering the correctly classified instances were about 74.5% which is quite high and showing high level of

accuracy as compared with other clustering techniques such as Hierarchical Clustering and Density-Based Clustering. Similarly, the precision, recall, and F-measure values of 0.75, 0.70, and 0.70 respectively were higher in comparison to other clustering techniques such as Hierarchical Clustering and Make Density Based Clustering.

Table 4.59: Performance Measure Class Wise (Canopy Clustering)

S. No.	n (truth)	n (classified)	Accuracy	Precision	Recall	F1 Score	Class
1	86	80	0.77	0.75	0.70	0.72	Node1
2	30	40	0.76	0.28	0.37	0.31	Node2
3	44	40	0.96	0.95	0.86	0.90	Node3
4	40	40	1.00	1.00	1.00	1.00	Node4

Based on the above table 4.59, it can be concluded that Node 4 has shown higher precision with value 1. Similarly, the recall value is found to be higher in case of Node 1.

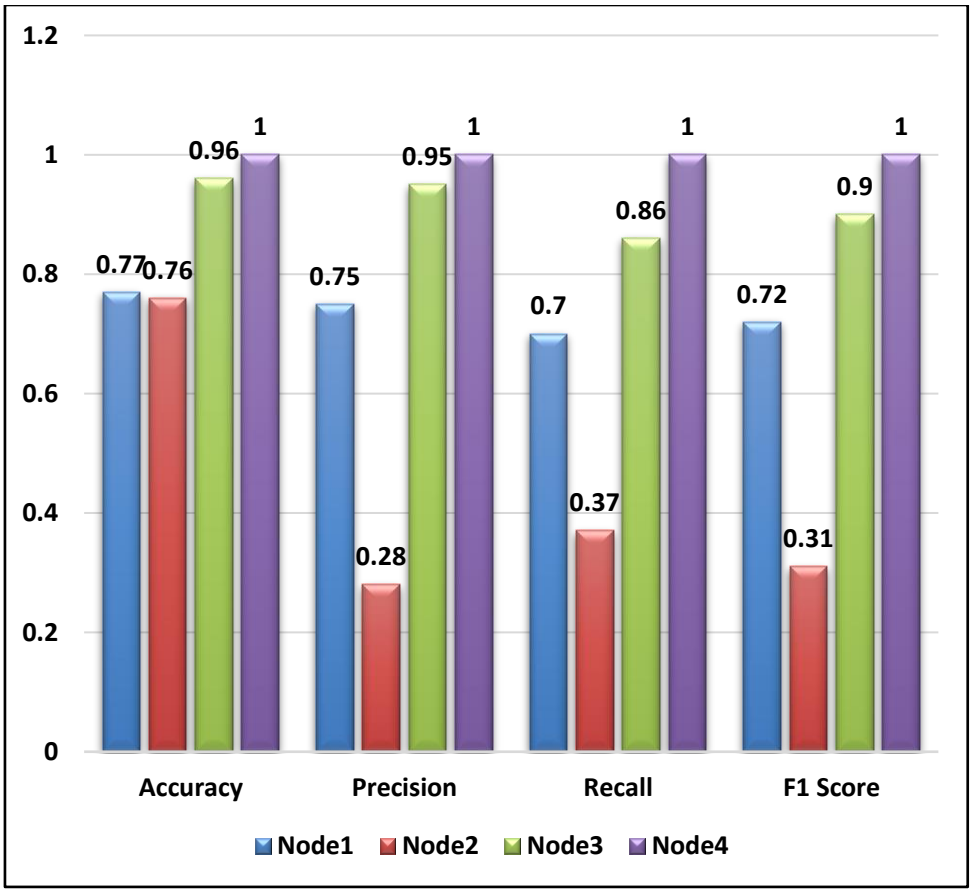

Figure 4.16: Class-wise performance measures

As shown in Figure 4.16, Class-wise performance measures the accuracy of the Node 4 is found to be highest with the value of 1 whereas the accuracy of Node 2 is found to be lowest with the value of 0.28.

Table 4.60: Confusion Matrix (Canopy Clustering)

0 1 2 3 ←assigned to cluster
60 18 2 0 | Cluster 0: Node1
25 11 4 0 | Cluster 1: Node2
1 1 38 0 | Cluster 2: Node3
0 0 0 40 | Cluster 3: Node4

From Table 4.60 the confusion matrix for Canopy Clustering shows the distribution of data points across clusters. It reveals correct and incorrect cluster assignments, helping assess the algorithm's performance. Cluster 0 (Node1) has 60 correct, 18, and 2 incorrect assignments; Cluster 1 (Node2) has 25 correct, 11 and 4 incorrect; Cluster 2 (Node3) has 1 correct, 1 and 38 incorrect; and Cluster 3 (Node4) has 40 correct assignments.

4.7.2 Hierarchical Clustering

Table 4.61: Overall Accuracy Hierarchical Clustering

Measures	Values
Correctly Classified Instances	118 (59%)
Incorrectly Classified Instances	82 (41%)
Overall Accuracy	38.14%
Total Number of Instances	200
Time taken to build a model	0.03 seconds

Table 4.61 the overall accuracy of Hierarchical Clustering is 38.14%, indicating that only 38.14% of the instances were correctly classified, while the remaining instances were misclassified. This relatively low accuracy suggests that the clustering algorithm may not be performing well on the given dataset.

Table 4.62: Class or Node-wiseHierarchical Clustering Performance Measures

S. No.	n (truth)	n (classified)	Accuracy	Precision	Recall	F1 Score	Class
1	150	74	58	0.97	0.48	0.64	Node1
2	4	40	78	0.03	0.25	0.05	Node2
3	39	40	59	0.00	0.00	0.00	Node3
4	1	40	79	0.03	1.00	0.05	Node4

Table 4.62 shows performance metrics for different classes in a classification task. Node1 achieved high accuracy 58% and precision 0.97 but lower recall 0.48 and F1 Score 0.64. Node2 had good accuracy 78% but low precision of 0.03 and recall 0.25.

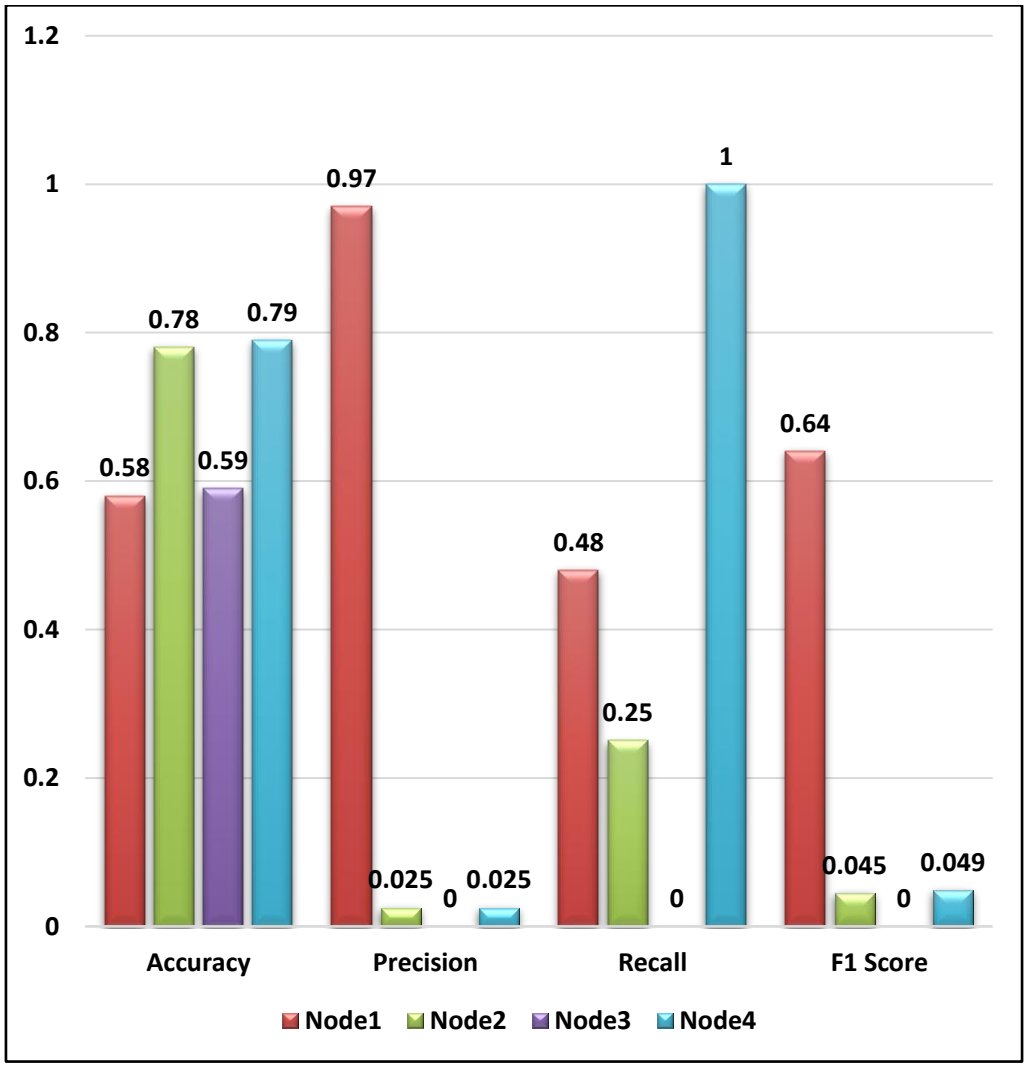

Figure 4.17: Class or Node-wiseHierarchical Clustering Performance Measures

As shown in Figure 4.17, Node3 showed moderate accuracy 59% but had no precision, recall, or F1 Score due to zero true positives. Node4 had high accuracy 79% and recall 1 but low precision 0.03 and F1 Score 0.05. The evaluation highlights the varying strengths and weaknesses of each class's classification performance.

Confusion Matrix: It was found that in case of Hierarchical Clustering the correctly classified instances were about 59% which is quite less and shows a low level of accuracy as compared with other clustering techniques such as Canopy Clustering.

Table 4.63: Confusion Matrix (Hierarchical Clustering)

0 1 2 3 ←assigned to cluster
7220 0 | Cluster 0: Node1
39 1 0 0 | Cluster 1: Node2
39 1 0 0 | Cluster 2: Node3
0 0 391| Cluster 3: Node4

Similarly, From Table 4.63 the precision and F-measure values of 0.02 and 0.04 respectively were lower in comparison to other clustering techniques.

4.8.3 Make Density-Based Clustering

The overall accuracy of Density-Based Clustering is 19.5%, indicating that only 19.5% of the instances were correctly classified.

Table 4.64: Overall Accuracy Make Density-Based Clustering

Measures	Values
Correctly Classified Instances	97 (48.5%)
Incorrectly Classified Instances	103 (51.5%)
Overall Accuracy	19.5%
Total Number of Instances	200
Time taken to build model	0.01 seconds

From Table 4.64 the classification model achieved an accuracy of 19.5%, with 97 instances correctly classified and 103 instances incorrectly classified out of a total of 200 instances. This indicates that the model's performance is relatively poor, as it correctly classified less than half of the instances. This low accuracy suggests that the clustering algorithm may not be performing well on the given dataset.

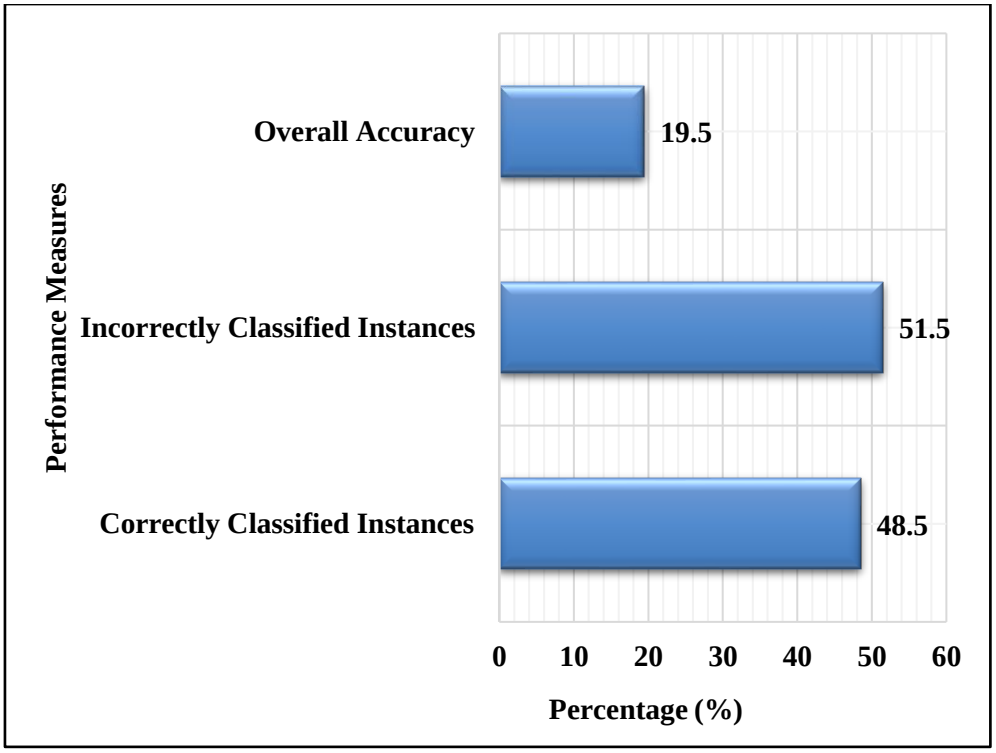

Figure 4.18: Overall Accuracy Make Density-Based Clustering

As shown in Figure 4.18, The Density-Based Clustering model has a relatively high error rate, with 103 instances and 51.5% being incorrectly classified. However, it is important to note that the model was built quickly, taking only 0.01 seconds to complete.

Table 4.65: Class or Node wise Make Density-Based Clustering Performance Measures

S. No.	n (truth)	n (classified)	Accuracy	Precision	Recall	F1 Score	Class
1	50	80	61	0.33	0.52	0.40	Node1
2	40	40	60	0.00	0.00	0.00	Node2
3	55	40	65	0.33	0.24	0.27	Node3
4	55	40	52	0.00	0	0.00	Node4

Table 4.65 Node1 achieved moderate accuracy 61% with relatively low precision 0.33 and recall 0.52, resulting in an F1 Score of 0.40. Node2 had a similar accuracy 60%, but it had no precision, recall, or F1 Score due to zero true positives.

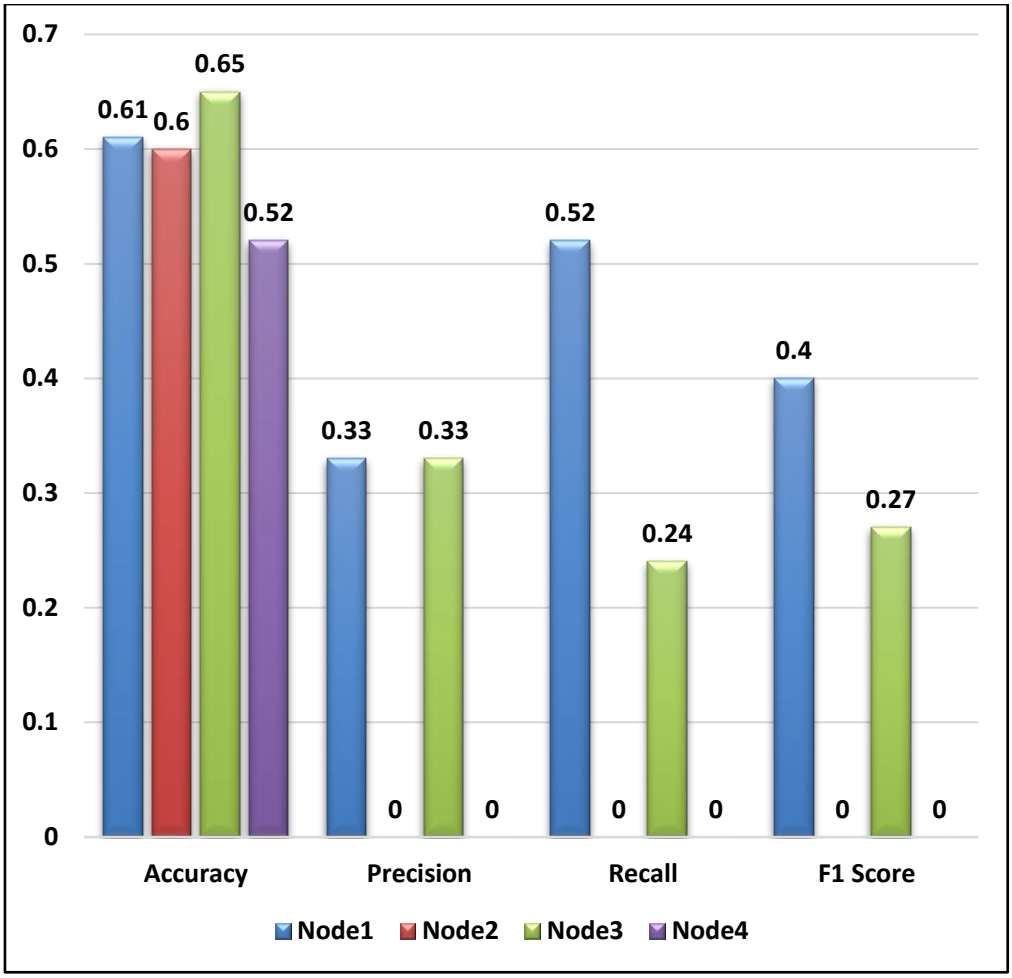

Figure 4.19: Class or Node wise Make Density-Based Clustering Performance Measures

As shown in Figure 4.19, it observes that Node3 performed slightly better with higher accuracy 65% and precision 0.33, but its recall 0.24, and F1 Score 0.27 remained relatively low. Node4 had the lowest accuracy 52%, and its precision, recall, and F1 Score were all zero.

Confusion Matrix

The confusion matrix for the Density-Based Clustering shows the distribution of data points across clusters. Cluster 0 (Node1) contains 2,602,628 data points correctly assigned to it. Cluster 1 (Node2) contains 1, 101, and 613 data points correctly assigned to it. Cluster 2 (Node3) has 130 data points correctly assigned, but 1, 314 data points were mistakenly placed in other clusters. Cluster 3 (Node4) contains 40 data points correctly assigned to it. The matrix provides valuable insights into the clustering performance, with most data points correctly clustered in Cluster 0 and Cluster 1, but some misclassifications in Cluster 2.

Table 4.66: Confusion Matrix (Make Density-Based Clustering)

0 1 2 3 ←assigned to cluster
2602628 | Cluster 0: Node1
1101613 | Cluster 1: Node2
130 1314| Cluster 2: Node3
0 40 0 0| Cluster 3: Node4

Accordingly, Table 4.66 found that in case of Make Density Clustering the correctly classified instances were about 48.5% which is quite low and shows a low level of accuracy as compared with other clustering techniques such as Hierarchical Clustering and Canopy Clustering. Similarly, the precision, recall, and F-measure values were lower in comparison to other clustering techniques such as Hierarchical Clustering and Canopy Clustering.

Chapter-5
Allocation and Scheduling of Computational Power

5.1 Task Offloading

5.2 Task Offloading and Resource Management System

5.3 Comparative Analysis Based on Cross-Validation 10 Folds

5.4 Comparative Analysis Based on Cross-Validation 20 Folds

5.5 Comparative Analysis Based on Split 33%

5.6 Overall Performance of Classification Algorithms

The task offloading & allocation and scheduling of computational power is going to go through the process of allocating and scheduling the computing resources that are shared among IoT devices. The many different categorization algorithms that are based on machine learning are now being investigated, and the most effective methods that are most suitable for fog computing are being identified. During testing, the impact that the proposed work would have on the latency issue that the existing system is experiencing will be evaluated. The implementation of a SMART FOG protocol-based approach to the creation of a fog environment that enables the sharing of computing resources with IoT devices is the major emphasis of this research work.

5.1 Task Offloading

Intelligent systems and smart applications that are self-sufficient, adaptable, and knowledge-based are currently being created. Among them are aerospace, healthcare, IoT, emergency and disaster management, and mobile apps, which are revolutionizing the computer industry. Applications with a high number of expanding devices have made the centralized cloud existing design unworkable. Despite the usage of 5G technology, delay-sensitive apps and the cloud cannot operate simultaneously owing to certain characteristics, such as latency, bandwidth, reaction time, etc., surpassing threshold levels. The use of middleware demonstrates that it is a more effective way to address these problems and yet adhere to the strict rules for job offloading. Middleware that uses fog computing is advised in this due to the services being offered at the network's edge, delay-sensitive applications can be efficiently used with this study article. Contrarily, fog nodes have a finite number of resources, which means they might not be able to handle all jobs, particularly those from computation-intensive applications. Moreover, fog is not a replacement for the cloud but rather an addition to it. Both technologies function similarly and provide services by job requirements, although fog computing is closer to the devices than the cloud is. The issue occurs when a decision must be made on what should be offloaded: data, particularly where to offload the computer or application in the cloud or the fog as well as how much to offload. When it comes to task-related characteristics like task size, duration, arrival rate, and needed resources, fog-cloud collaboration is stochastic.

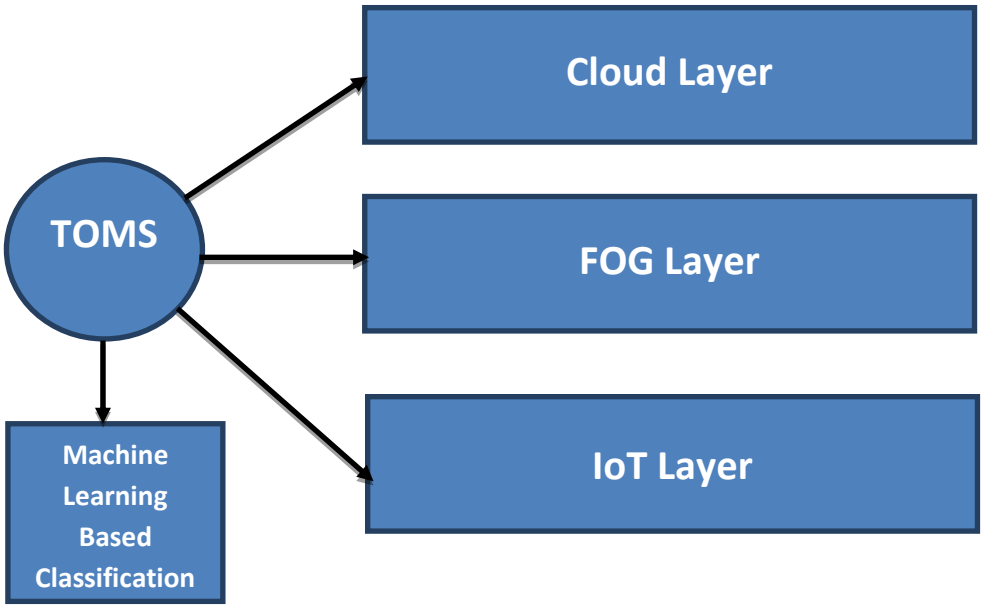

Figure 5.1: Proposed Task Offloading Management System (Li, 2019)

To better utilize the resources at the fog and cloud to improve QoS, dynamic task offloading becomes essential. Due to the complexity of this job-offloading policy creation, the research work addresses this issue and suggests an intelligent task-offloading model.

5.2 Task Offloading and Resource Management System

The Task Offloading & Resource Management System is a sophisticated framework designed to optimize task allocation and resource distribution within an IoT and fog computing environment. By leveraging real-time monitoring, machine learning-based analysis, and a policy repository for offloading criteria, the system intelligently determines when and where to offload computational tasks from IoT devices to fog nodes. Efficient resource management ensures that tasks are allocated to the most suitable nodes based on factors such as task urgency and available resources, leading to reduced latency and improved overall system performance. Through rigorous performance evaluation, the system ensures the reliability and effectiveness of the classification algorithms used for task allocation, contributing to seamless task distribution and optimal resource utilization throughout the network. The system consists of the following five main characteristics

The system consists of the following five main characteristics:

1. Task offloading criteria details: policy repository
2. Status of Fog layer: devices
3. Analysing the offloading and resource allocation using ML – approaches like various classification algorithms.
4. Using various performance measures to evaluate classification algorithms.
5. Suggest the best predictive construct.

The system comprises five main characteristics: a policy repository for task offloading criteria details, real-time monitoring of the fog layer status through devices, machine learning-based analysis using various classification algorithms to determine task offloading and resource allocation decisions, evaluation of classification algorithms using multiple performance measures, and the suggestion of the best predictive construct. These features together enable efficient task allocation, resource utilization, and decision-making in IoT and fog computing environments, optimizing system performance and improving overall efficiency.

Experimental Setup

The iFogSim simulator is being used for developing the Smart Fog environment. The dataset is being constructed recording the various values of attributes like No. of Fog system, Areas, Number of Cameras Per Area, Execution Time, ALD: motion_detector, object_detector, object_tracker, ALD: object_tracker, PTZ_CONTRO, CPU Delay: MOTION_VIDEO_STREAM, CPU Delay: DETECTED_OBJECT, CPU Delay: OBJECT_LOCATION, CPU Delay: CAMERA, Latency, Energy Consumed, Cost of execution, Total network usage, MIPS Million instructions per second, Number of processing elements, RAM, Priority, Previous Time etc.

Algorithm Executed

1) Load D

2) Pre-processing D

3) Train D, test D, split D

4) Classification modeling (IBK, K-Star, MLP, Logistic Regression, Bagging…)

i) Task offloading prediction

for i =0 to EOF ()

```
for j = 0 to (X. length-1)
calculate Z
return Z
p = f (z)
if (p == 1)
{
offloads to Fog
}
Else
{
offloads to Cloud
}
}
```

ii) Evaluation of predictive model using:

Accuracy ( )

Confusion matrix ( )

Average Execution Time ( )

iii) Comparative Analysis:

Comparative Eval (IBK, K-Star, MLP, Logistic Regression, Bagging…)

iv) Identify the most appropriate classifier or predictive model.

5) Implement the Constructed Predictive Model.

The suggested fog-cloud intelligent task offloading paradigm is evaluated and assessed using a simulated environment for machine learning Weka 3.8.4, a data science platform for data scientists, IT specialists, and business executives, has been used to carry out the simulation. A variety of machine learning techniques are used to train the model, with the recommended approach being LR, along with K-Nearest Neighbor, Nave Bayes, Decision Tree, Support Vector Machine, and MLP[99].

[99]Multiple Layer Perceptron

5.3 Comparative Analysis Based on Cross-Validation 10-Folds

Cross-validation – 10-fold: The 10-fold cross-validation provides a robust estimation of each classifier's generalization ability, as it tests the algorithms on different subsets of data, ensuring that the results are less sensitive to the specific data partitioning. For each fold, the classifiers are trained on nine folds and then tested on the remaining fold. This process is repeated ten times, with each fold serving as the testing set exactly once.

Table 5.1: Comparative Analysis of Classifiers Used for Task Offloading and Resource Allocation: 10-fold Cross Validation

Performance Measure	Logistic Regression	K-Star	IBK	J48	Bagging	MLP
Accuracy	0.82	0.53	0.55	0.75	0.69	0.91
Kappa Statistic	0.64	0.07	0.10	0.50	0.39	0.82
TP Rate	0.82	0.54	0.55	0.75	0.35	0.91
FP Rate	0.18	0.46	0.45	0.25	0.69	0.09
Precision	0.83	0.54	0.76	0.79	0.30	0.91
Recall	0.82	0.54	0.55	0.75	0.71	0.91
F-Measure	0.82	0.53	0.44	0.74	0.69	0.91
ROC Area	0.81	0.60	0.57	0.79	0.69	0.98
Mean Absolute Error	0.22	0.44	0.44	0.28	0.85	0.11
Execution Time Model Building	60ms	20ms	20ms	30ms	30ms	80ms

The performance metrics, such as accuracy, precision, recall, F1 score, and area under the receiver operating characteristic ROC curve, are calculated for each fold as shown in table 5.1.

Comparing the performance of the classifier based on Accuracy, Kappa statistics, TP rate, FP Rate, Precision, Recall, F-Measure, ROC Area, Mean Absolute Error, and Execution Time Model Building used for task offloading and resource allocation confirms that at configuration setting of cross-validation 10 folds in case of SMART FOG environment.

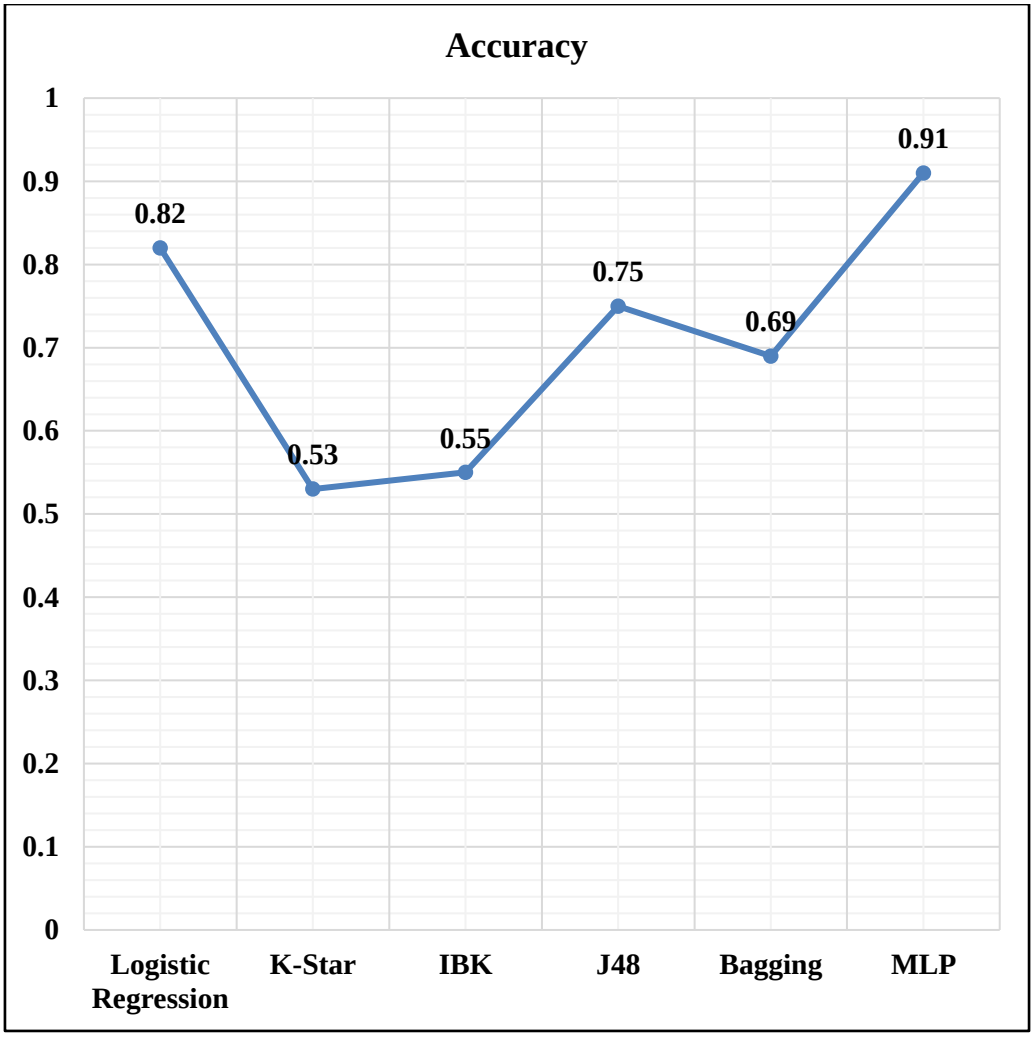

Figure 5.2: Accuracy Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.2, confirms that at configuration setting of cross-validation 10 folds the accuracy of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.82. The other classification algorithms had an accuracy of about 0.75 in case of J48 classifier, 0.69, 0.55, and 0.53 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure accuracy were found to be MLP and LR.

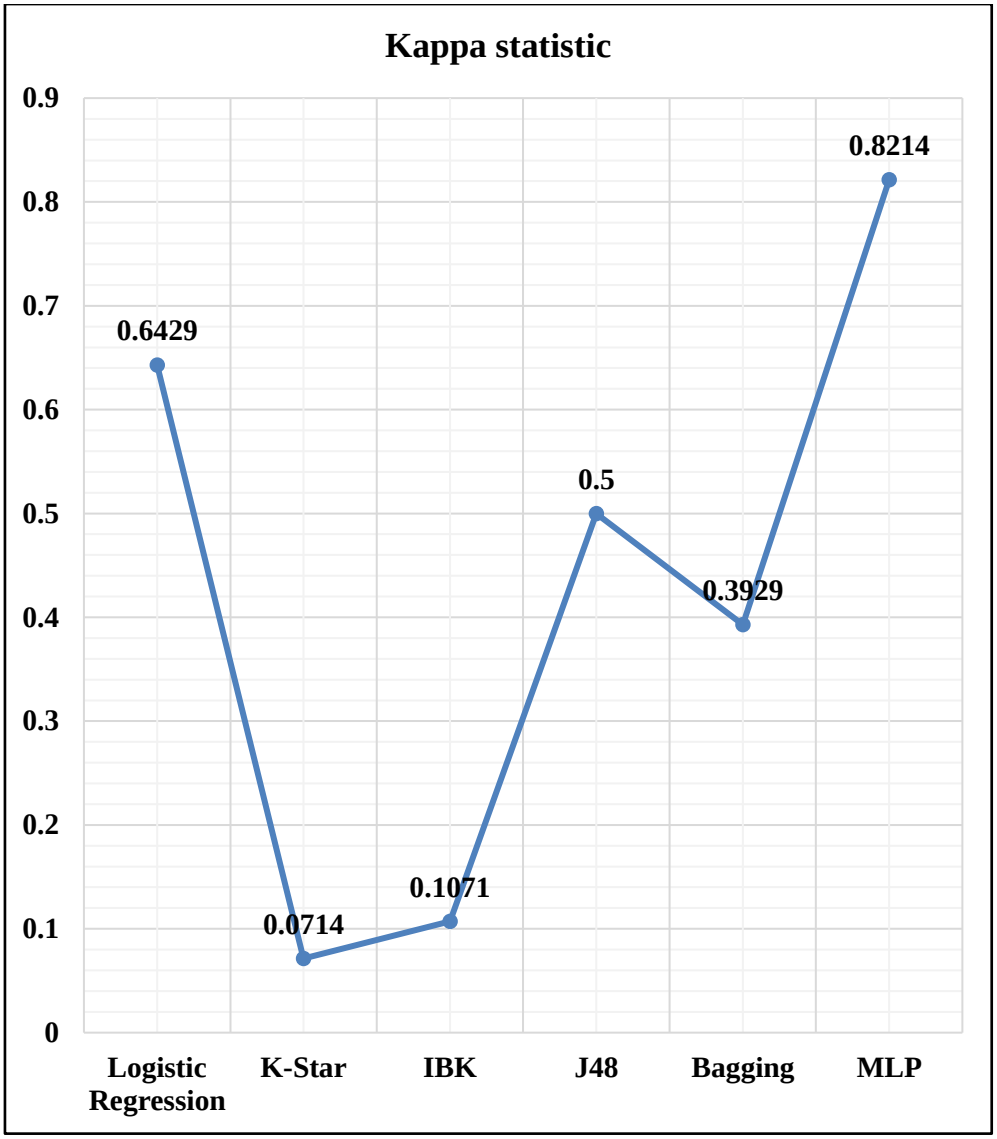

Figure 5.3: Kappa Statistic Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.3, shows the Comparison the performance of the classifier based on Kappa statistics used for task offloading and resource allocation in case of SMART FOG environment it can be interpreted that a higher Kappa statistics value of 0.82 in case of MLP and 0.64 in case of Logistic Regression suggests that they are the better classifiers as compared to other classification techniques.

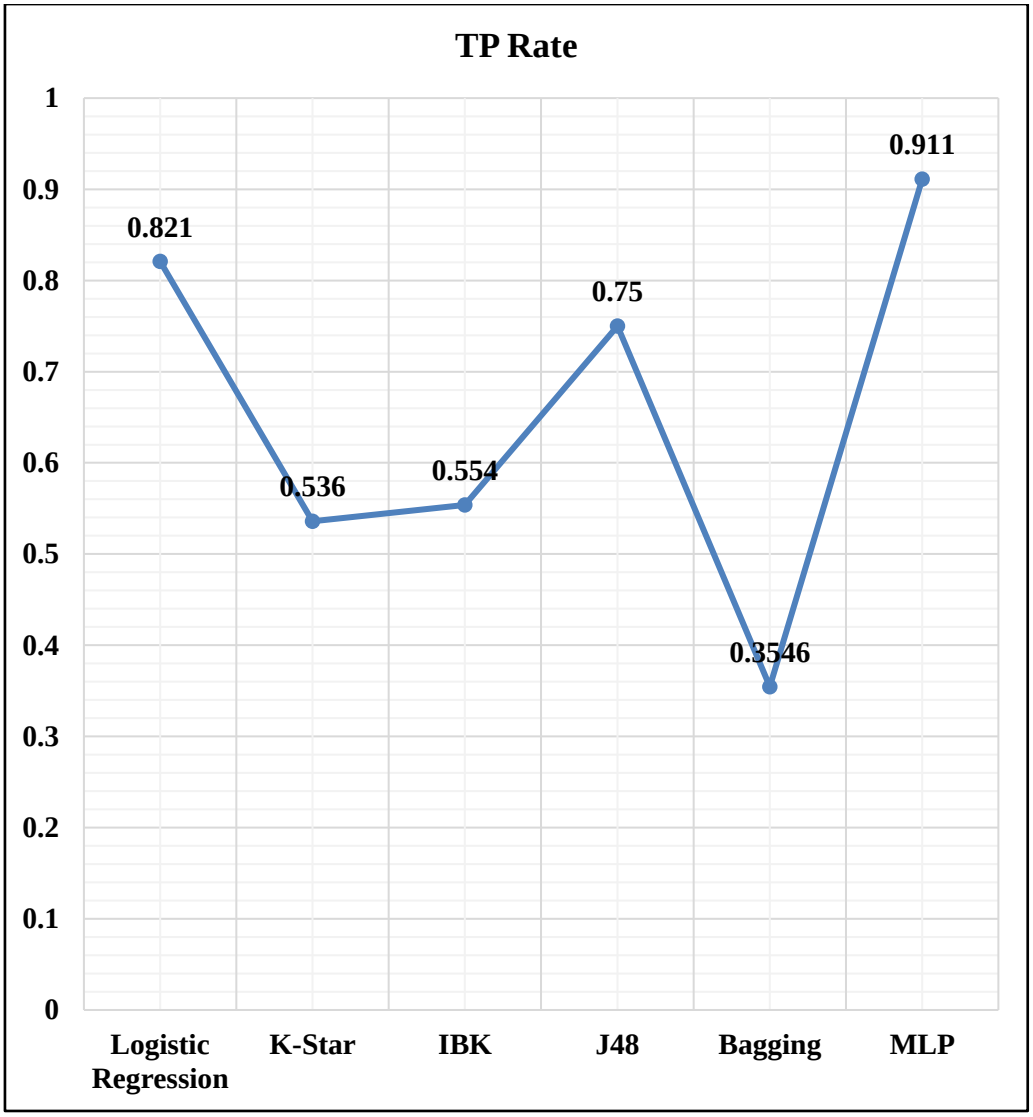

Figure 5.4: TP Rate Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

According to figure 5.4, it can be concluded that at the configuration setting of cross-validation, 10-fold the TP Rate of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with a value of 0.82. The other classification algorithms had to have TP Rate of about 0.75 in case of J48 classifier, 0.35, 0.55, and 0.54 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure TP rate were found to be MLP and LR.

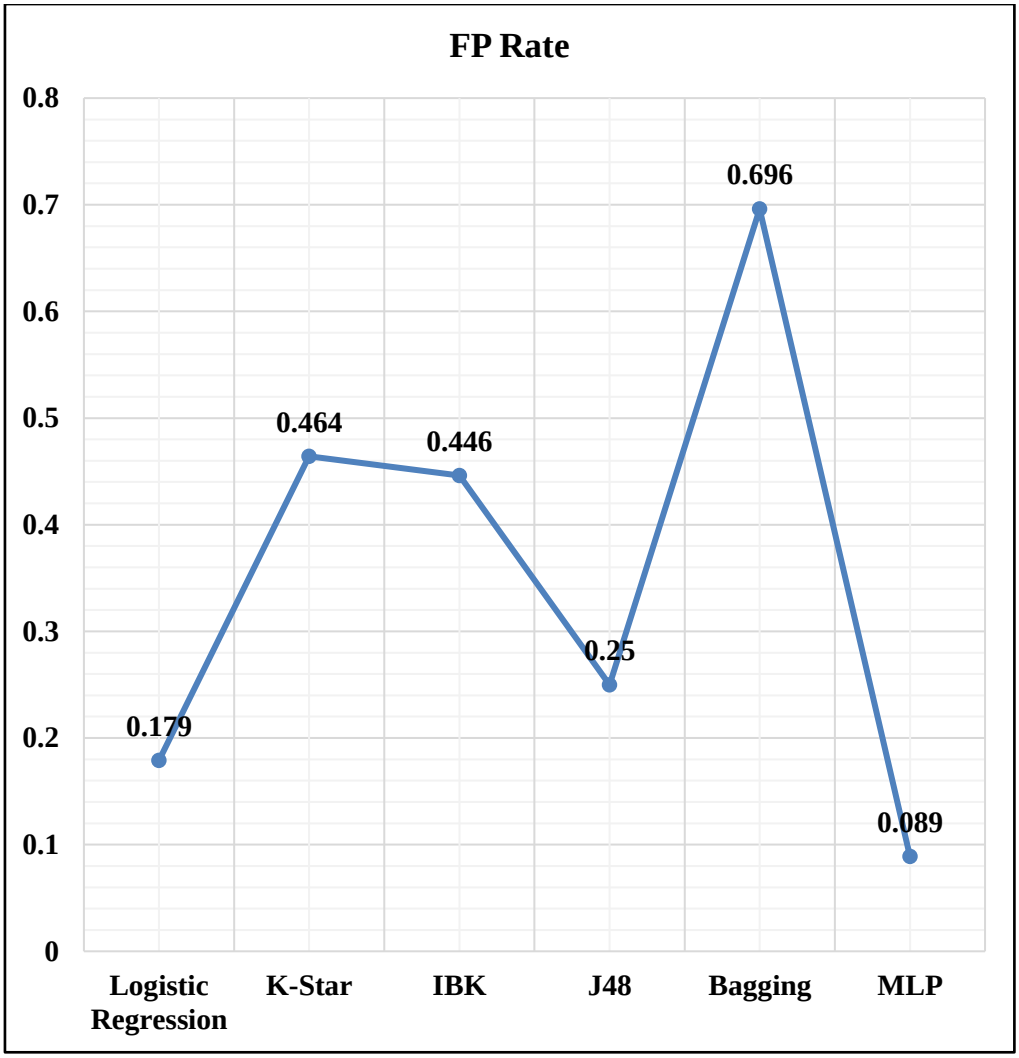

Figure 5.5: FP Rate Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.5, it can be concluded that at the configuration setting of cross-validation 10 folds the FP Rate of MLP classifier with value 0.09 is found to be lowest followed by Logistic Regression with value 0.18. The other classification algorithms had to have an FP Rate of about 0.25 in case of J48 classifier, 0.69, 0.45, and 0.46 in case of Bagging, IBK, and K-Star which were found to be quite higher. The most appropriate classifiers based on performance measure FP rate were found to be MLP and Logistic Regression having lesser FP rate values as compared to others.

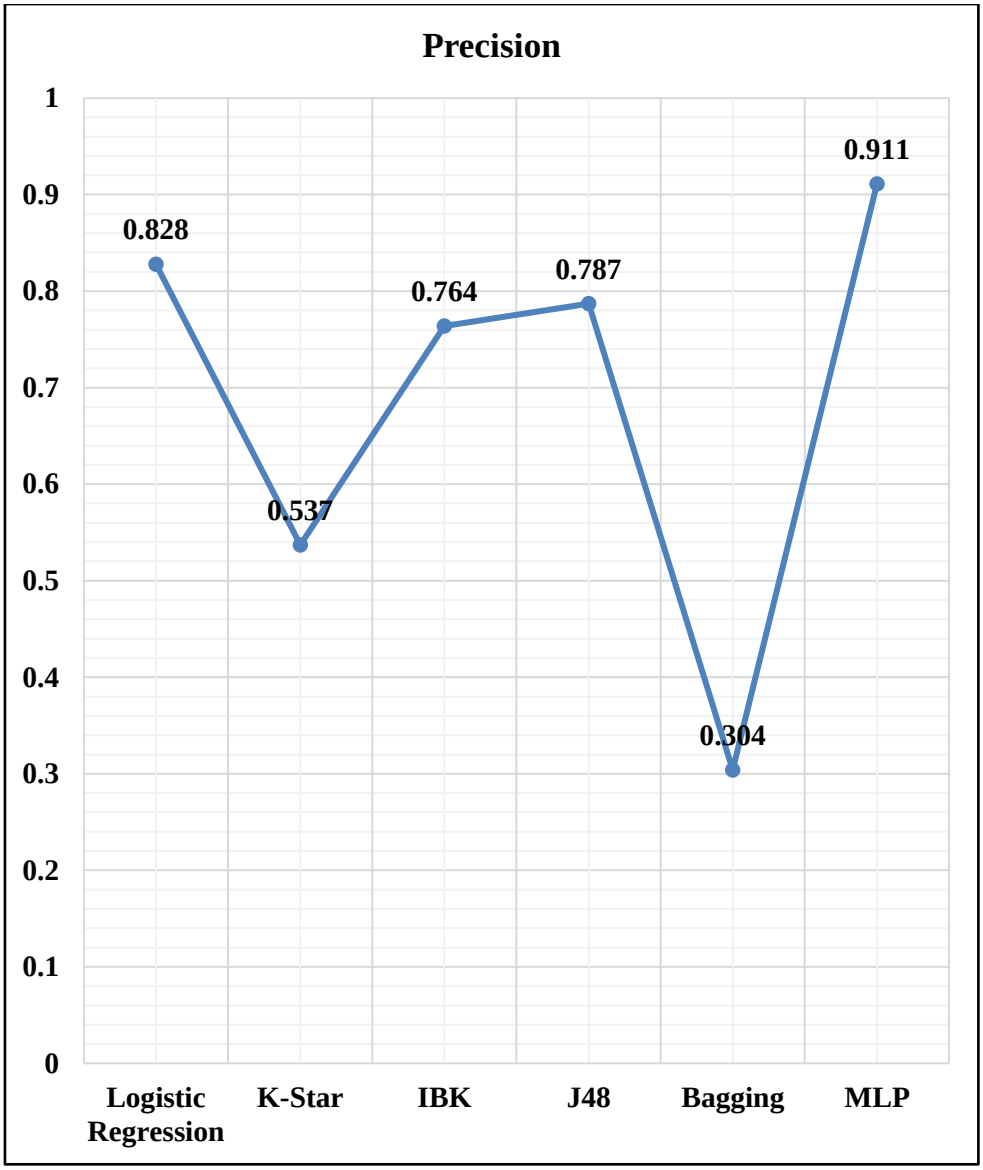

Figure 5.6: Precision Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.6, it can be concluded that at the configuration setting of cross-validation 10 folds the Precision of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.83. The other classification algorithms had to have a Precision of about 0.79 in case of J48 classifier, 0.30, 0.76, and 0.53 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure Precision were found to be MLP and LR.

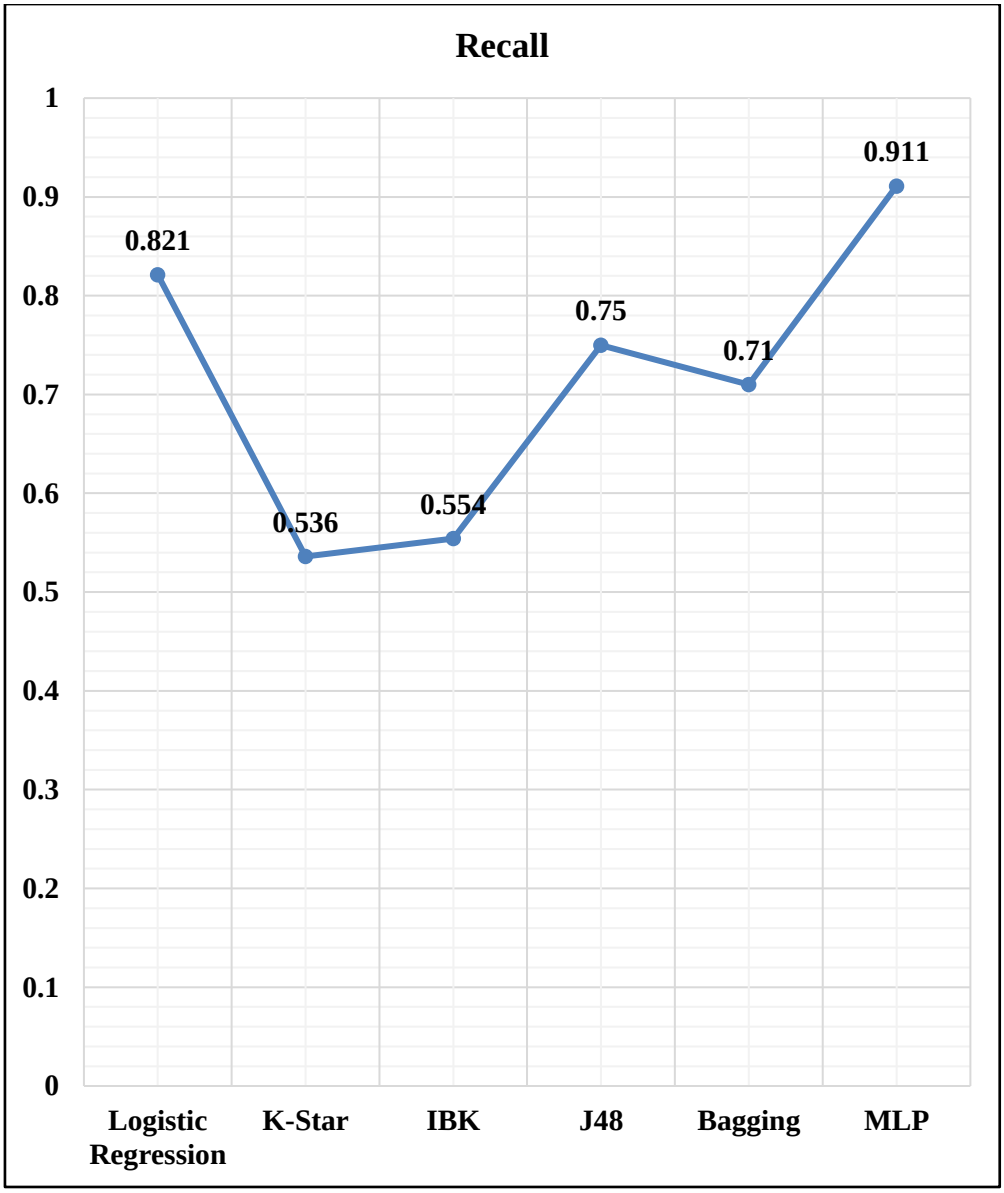

Figure 5.7: Recall Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Results as shown in figure 5.7, confirm that at configuration setting of cross-validation 10 folds the Recall of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.82. The other classification algorithms had to have a Recall of about 0.75 in case of J48 classifier, 0.71, 0.55, and 0.53 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure Recall were found to be MLP and LR.

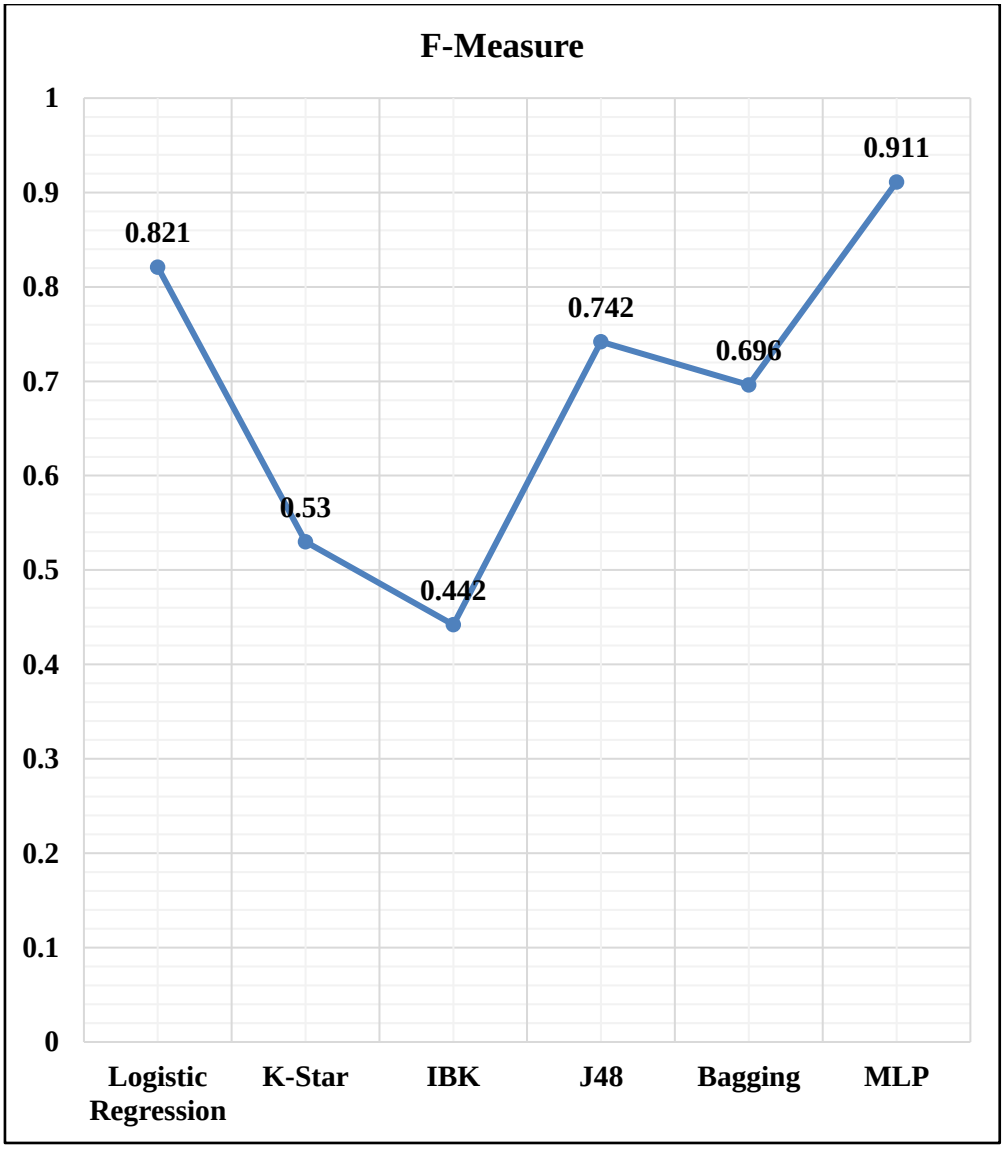

Figure 5.8: F-Measure Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

According the figure 5.8, it can be concluded that at the configuration setting of cross-validation, 10 folds the F-Measure of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.82. The other classification algorithms had to have an F-Measure of about 0.74 in case of J48 classifier, 0.69, 0.44, and 0.53 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure F-Measure were found to be MLP and LR.

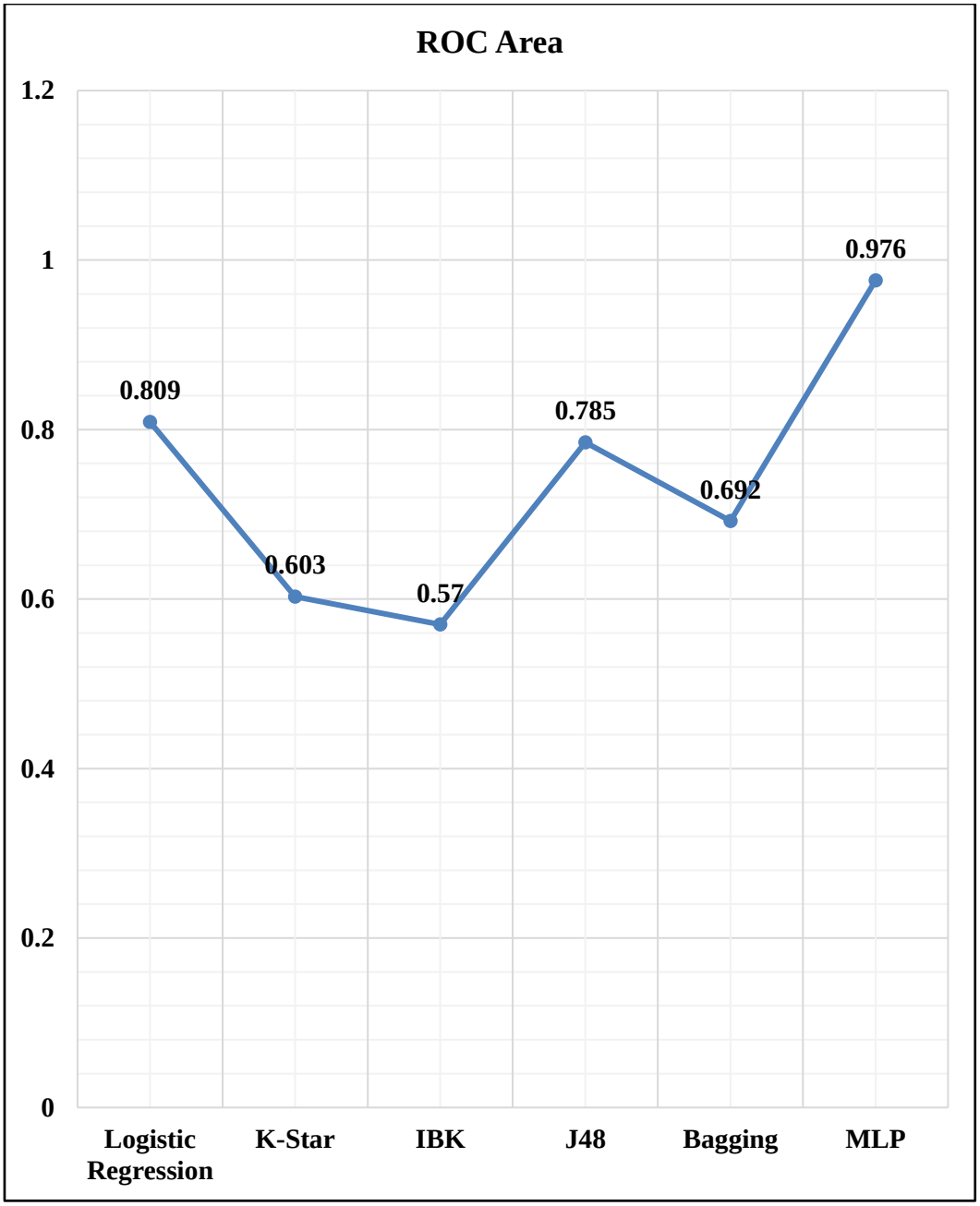

Figure 5.9: ROC Area Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.9, it can be concluded that at configuration setting of cross-validation 10 folds, the ROC Area of MLP classifier with value 0.97 is found to be highest followed by Logistic Regression with value 0.80. The other classification algorithms had to have a ROC Area of about 0.78 in case of J48 classifier, 0.69, 0.57, and 0.60 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure ROC Area were found to be MLP and LR.

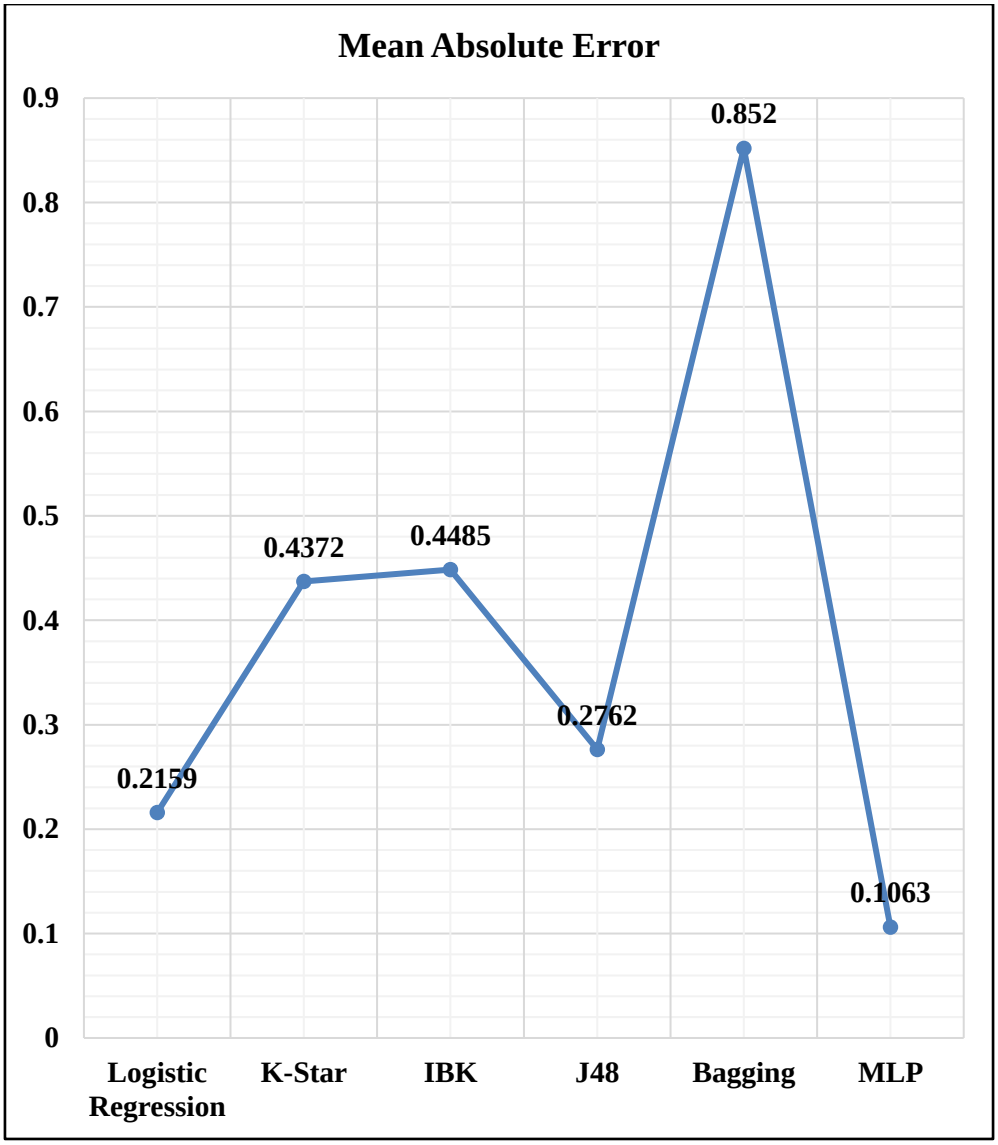

Figure 5.10: MAE Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.10, it can be concluded that at the configuration setting of cross-validation, 10 folds the mean absolute error value of MLP classifier with 0.10 is found to be lowest followed by Logistic Regression with value 0.21. The other classification algorithms had mean absolute error values of about 0.28 in case of J48 classifier, 0.85, 0.45, and 0.43 in case of Bagging, IBK, and K-Star were found to be quite high. The most appropriate classifiers based on performance measure mean absolute error value were found to be MLP and LR having lesser mean absolute error values as compared to others.

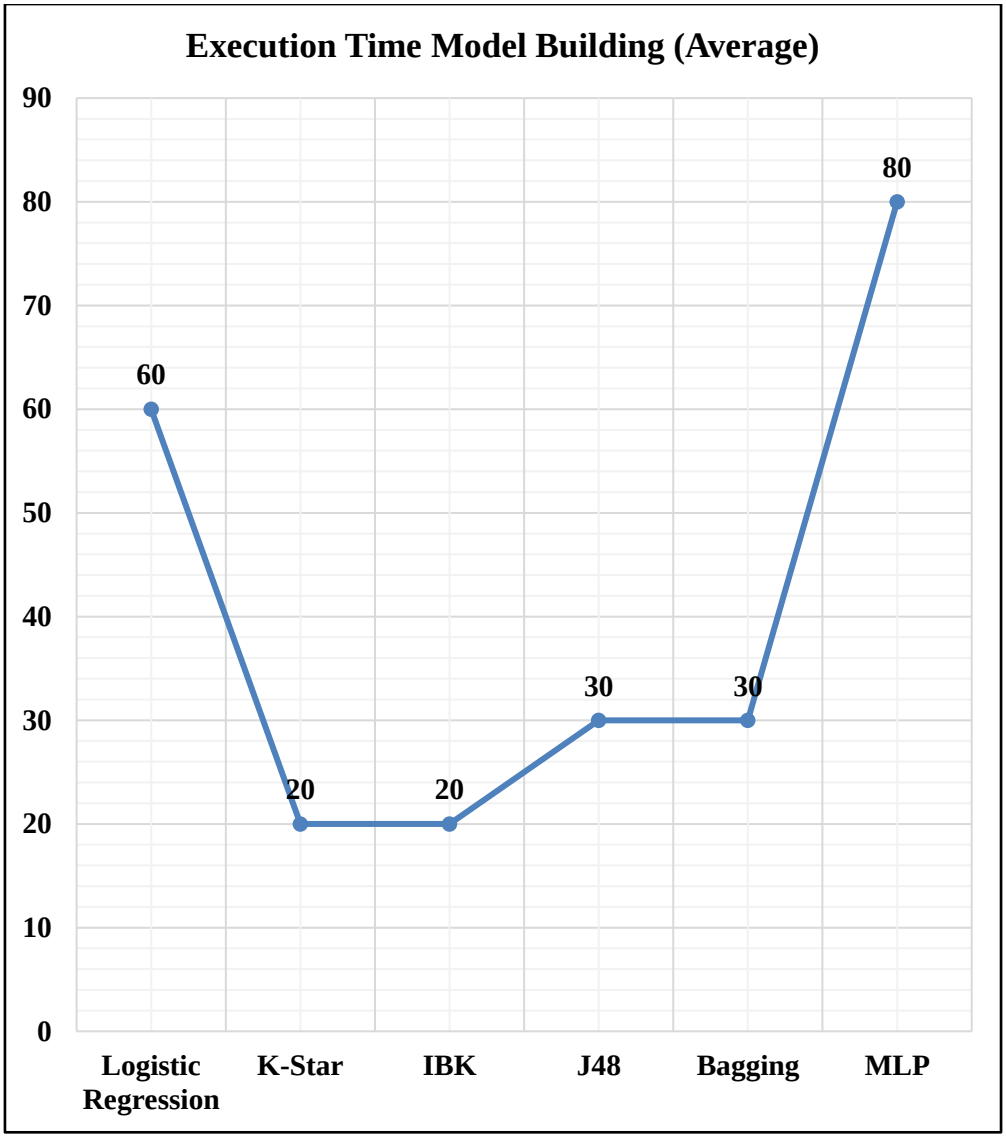

Figure 5.11: Average Execution Time for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-10 folds)

Figure 5.11, it can be concluded that at the configuration setting of cross-validation 10 folds the average execution time of model building of K-Star and IBK classifier is found to be 20 milliseconds which is quite less as compared with other classifiers. The other classification algorithms had to have an average execution time of model building of about 30ms in case of J48 classifier, 30, 60, and 80ms in case of Bagging, LR, and MLP. The most appropriate classifiers based on performance measure average execution time of model building were found to be K-Star and IBK.

5.4 Comparative Analysis Based on Cross-Validation 20 Folds

In the performance analysis of classification algorithms used for task offloading based on 20-fold cross-validation, the evaluation provides a comprehensive understanding of each algorithm's effectiveness in handling the task offloading problem. Cross-validation is a re-sampling technique that partitions the dataset into 20 subsets (folds), where each fold serves as both a training set and a testing set.

Table 5.2: Performance Analysis of Classification Algorithms Used for Task Offloading: 20fold Cross-validation

Performance Measure	Logistic Regression	K-Star	IBK	J48	Bagging	MLP
Accuracy	0.82	0.48	0.55	0.76	0.64	0.91
Kappa Statistic	0.64	-0.03	0.10	0.53	0.28	0.82
TP Rate	0.82	0.48	0.55	0.76	0.64	0.91
FP Rate	0.17	0.51	0.44	0.23	0.35	0.08
Precision	0.82	0.48	0.76	0.78	0.65	0.91
Recall	0.82	0.48	0.55	0.76	0.64	0.91
F-Measure	0.82	0.47	0.44	0.76	0.63	0.91
ROC Area	0.79	0.56	0.56	0.80	0.73	0.97
Mean Absolute Error	0.24	0.47	0.44	0.25	0.36	0.09
Execution Time Model Building	70ms	20ms	25ms	35ms	40ms	90ms

Table 5.2, shows that evaluation metrics, such as accuracy, precision, recall, F1 score, and area under the receiver operating characteristic ROC curve, are computed for each fold to assess the algorithm's performance consistently across different subsets of the data. The average performance metrics across all 20 folds provide a robust estimate of how well each algorithm generalizes to unseen data as shown in the figure below.

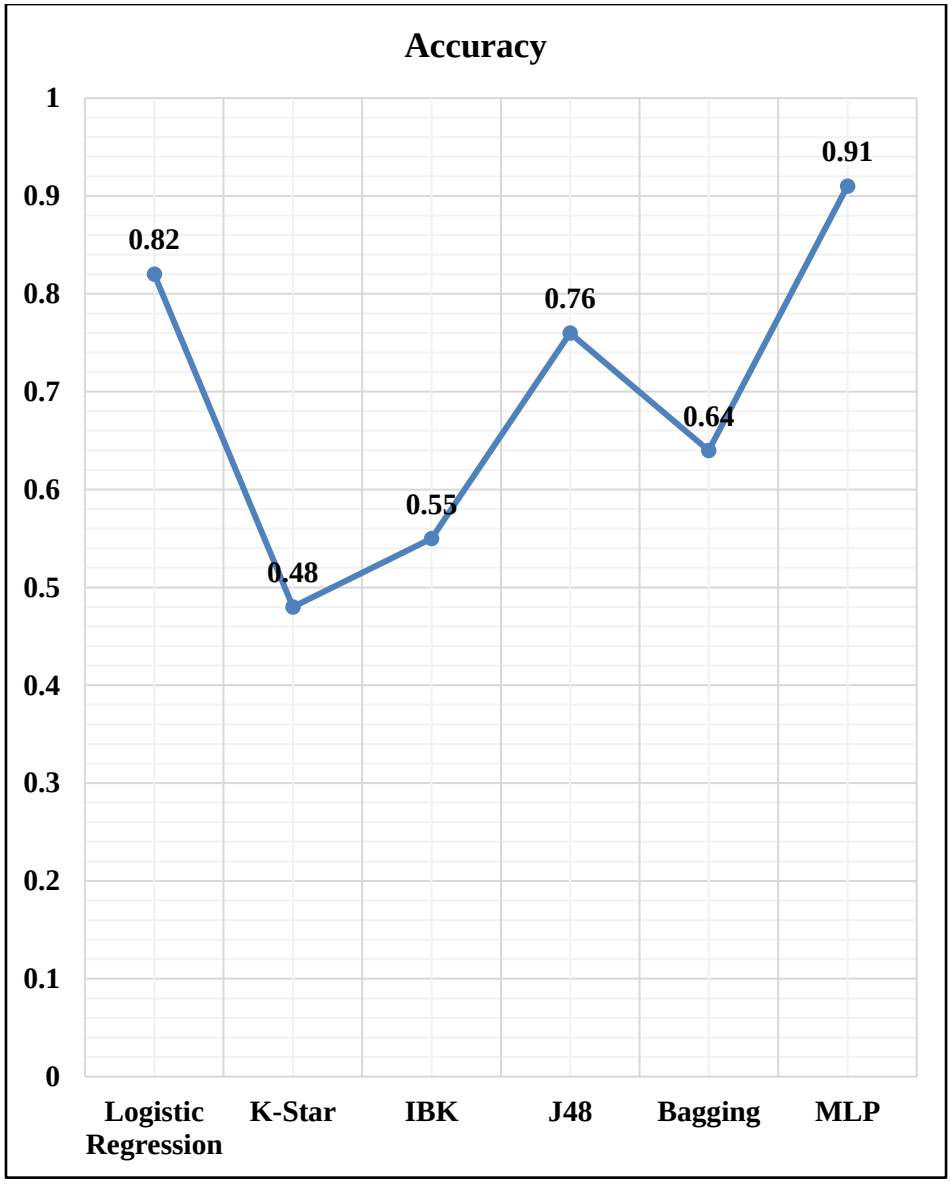

Figure 5.12: Accuracy Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.12, confirms that at the configuration setting of cross-validation, 20 folds the accuracy of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.82. The other classification algorithms had to have an accuracy of about 0.76 in case of J48 classifier, 0.64, 0.55, and 0.48 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure accuracy were found to be MLP and LR.

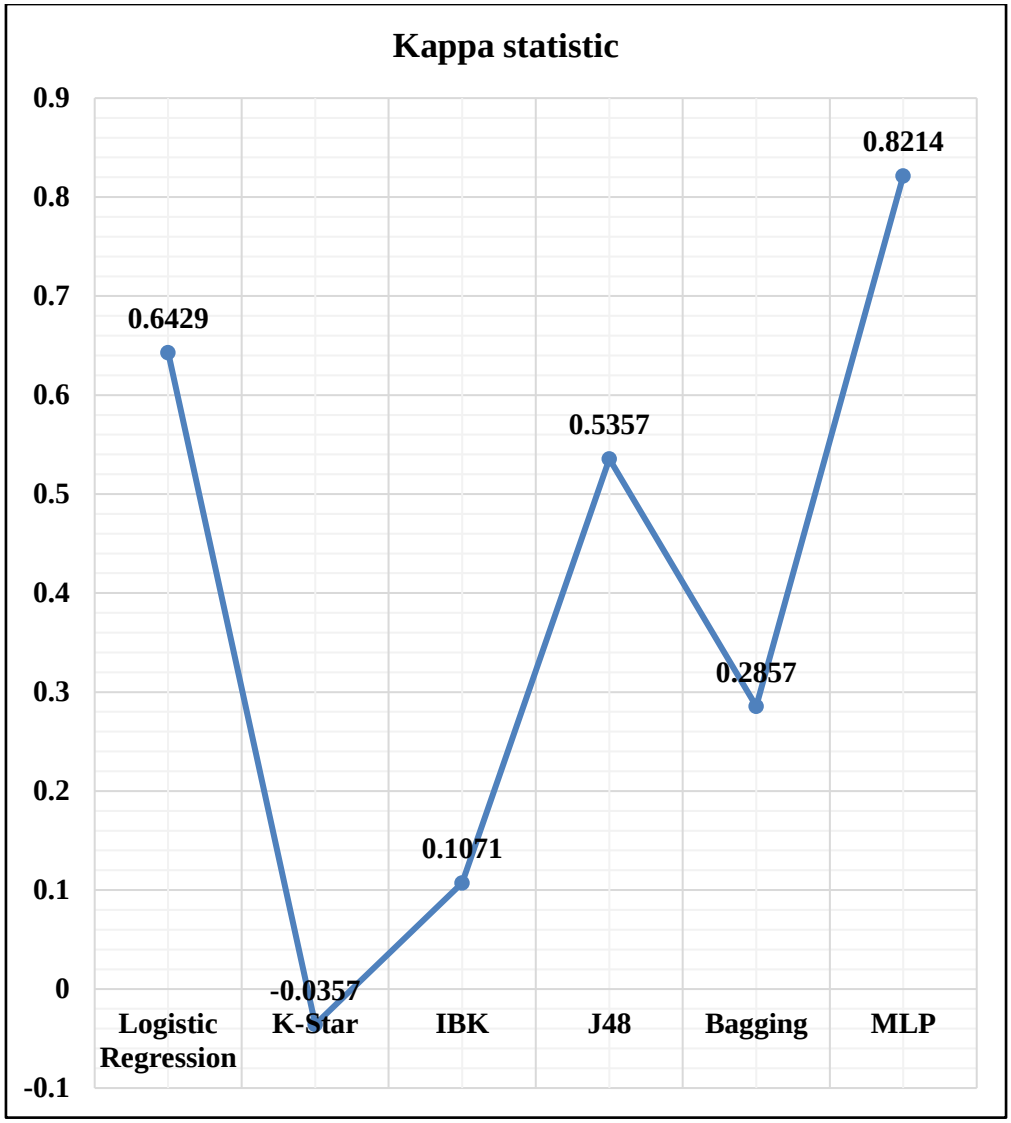

Figure 5.13: Kappa Statistics Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.13, it shows comparing the performance of classifiers based on Kappa statistics used for task offloading and resource allocation in case of SMART FOG environment it can be interpreted that a higher Kappa statistics value of 0.82 in case of MLP and 0.64 in case of Logistic Regression suggests that they are the better classifiers as compared to other classification techniques.

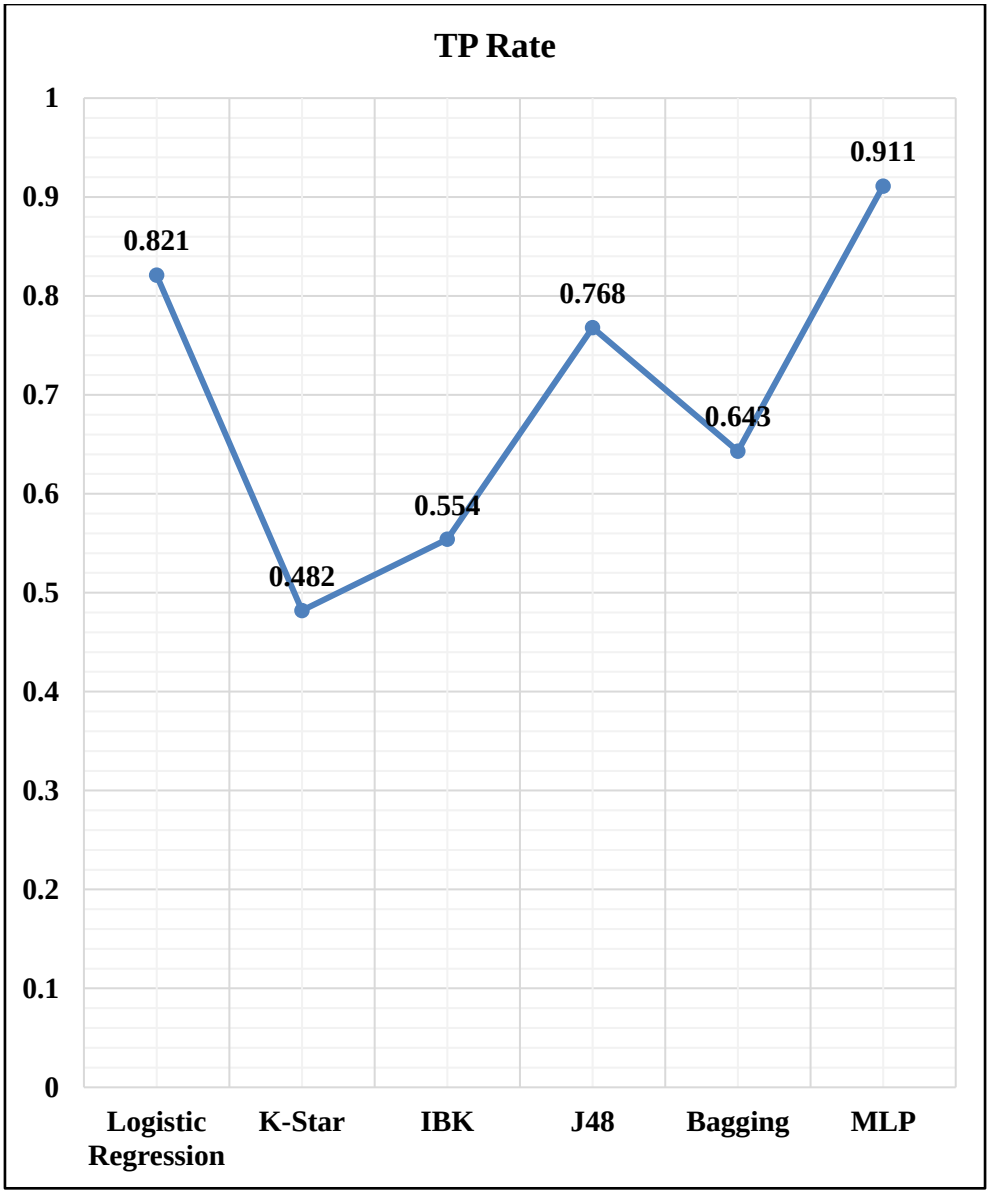

Figure 5.14: TP Rate Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

According to figure 5.14, it can be concluded that at configuration setting of cross-validation 20 folds the TP Rate of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.82. The other classification algorithms had to have a TP Rate of about 0.76 in the J48 classifier, 0.64, 0.55, and 0.48 in Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure TP rate were found to be MLP and LR.

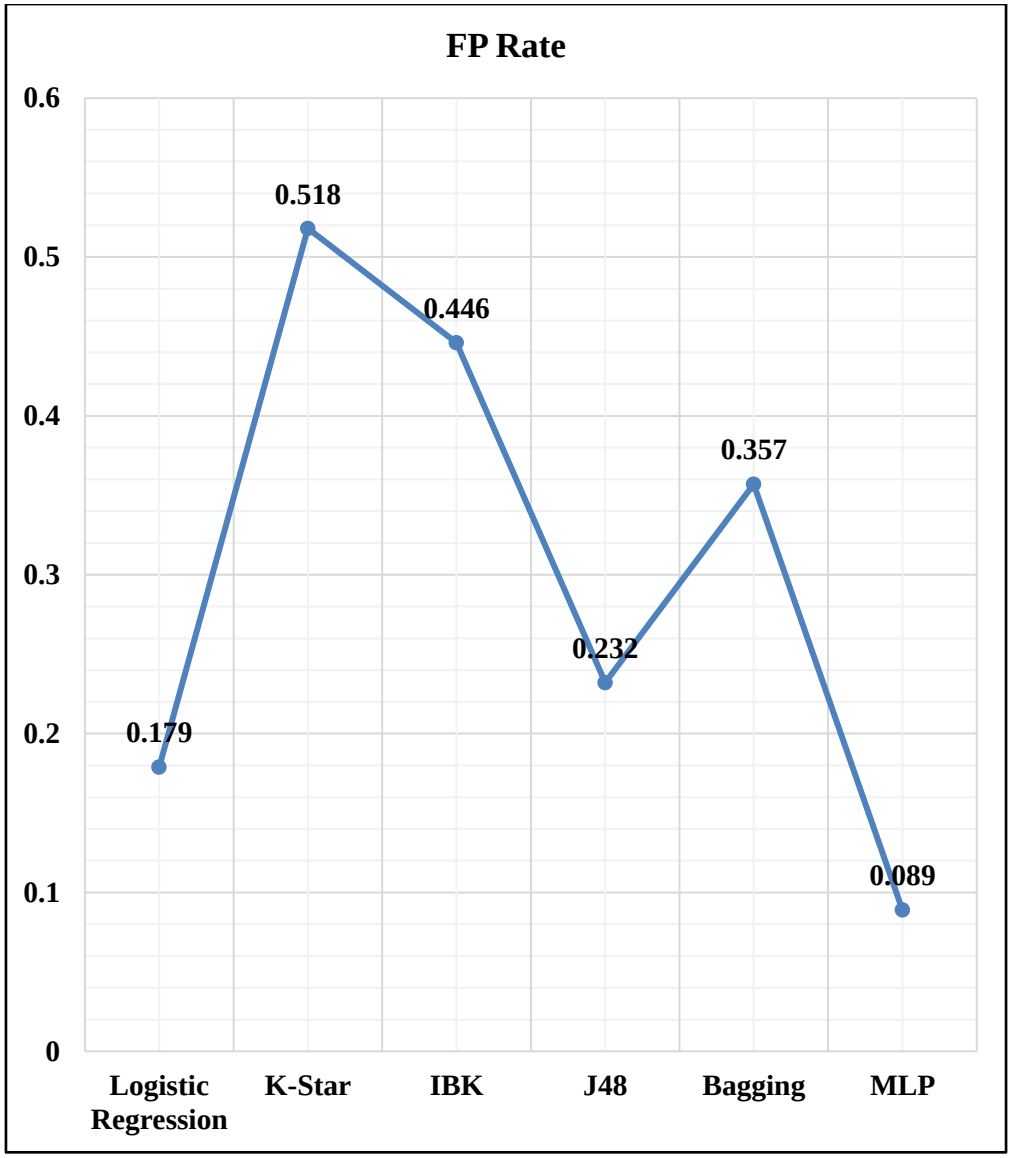

Figure 5.15: TP Rate Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.15, it can be concluded that at the configuration setting of cross-validation, 20 folds the FP Rate of MLP classifier with value 0.08 is found to be lowest followed by Logistic Regression with value 0.17. The other classification algorithms had to have an FP Rate of about 0.23 in case of J48 classifier, 0.35, 0.44, and 0.51 in case of Bagging, IBK, and K-Star which were found to be quite higher. The most appropriate classifiers based on performance measure FP rate were found to be MLP and LR having lesser FP rate values as compared to others.

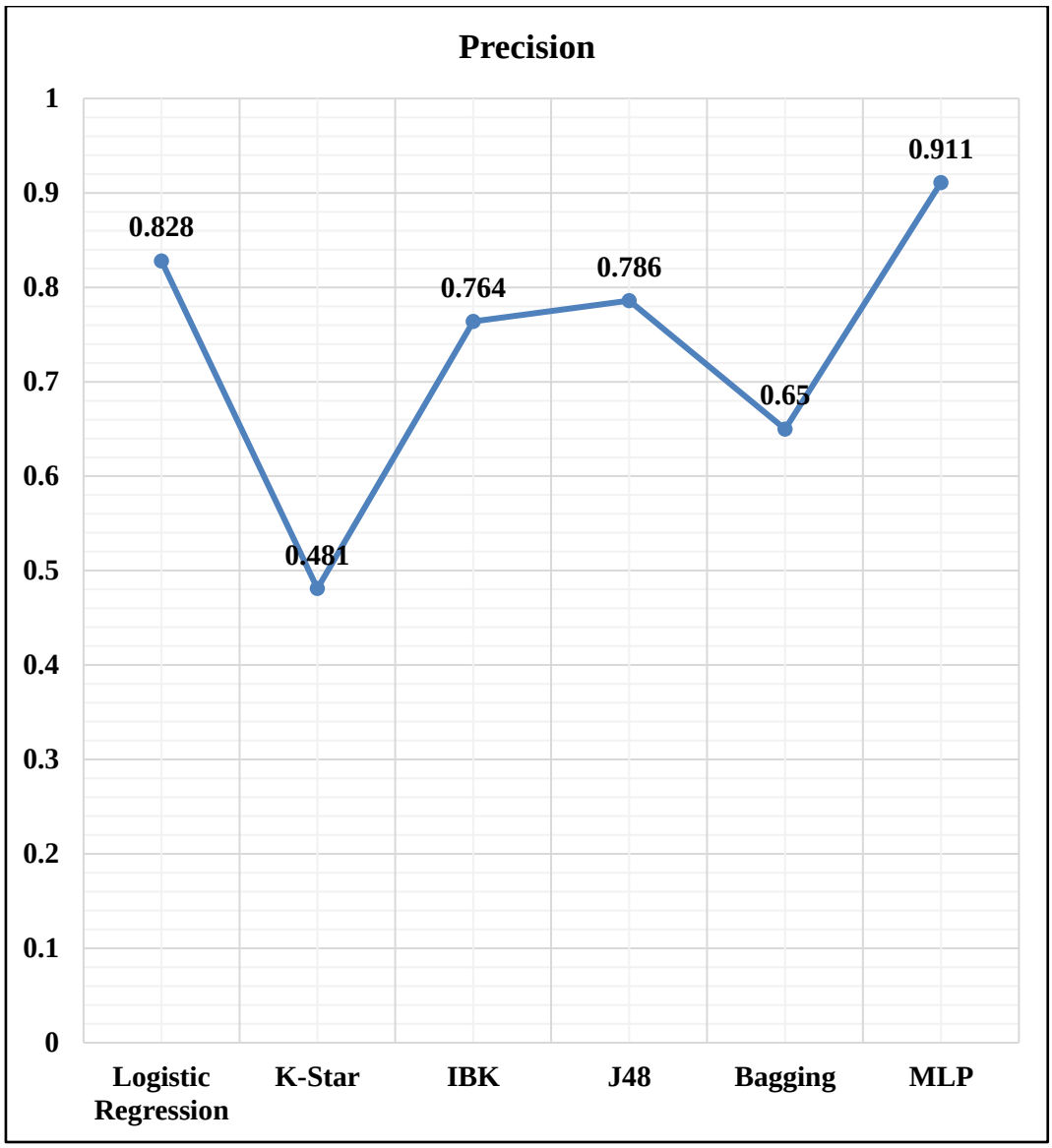

Figure 5.16: Precision Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.16, it can be concluded that at the configuration setting of cross-validation 20 folds the Precision of MLP classifier with value 0.91 is found to be the highest followed by Logistic Regression with value 0.82. The other classification algorithms had to have a Precision of about 0.78 in case of J48 classifier, 0.65, 0.76, and 0.48 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure Precision were found to be MLP and LR.

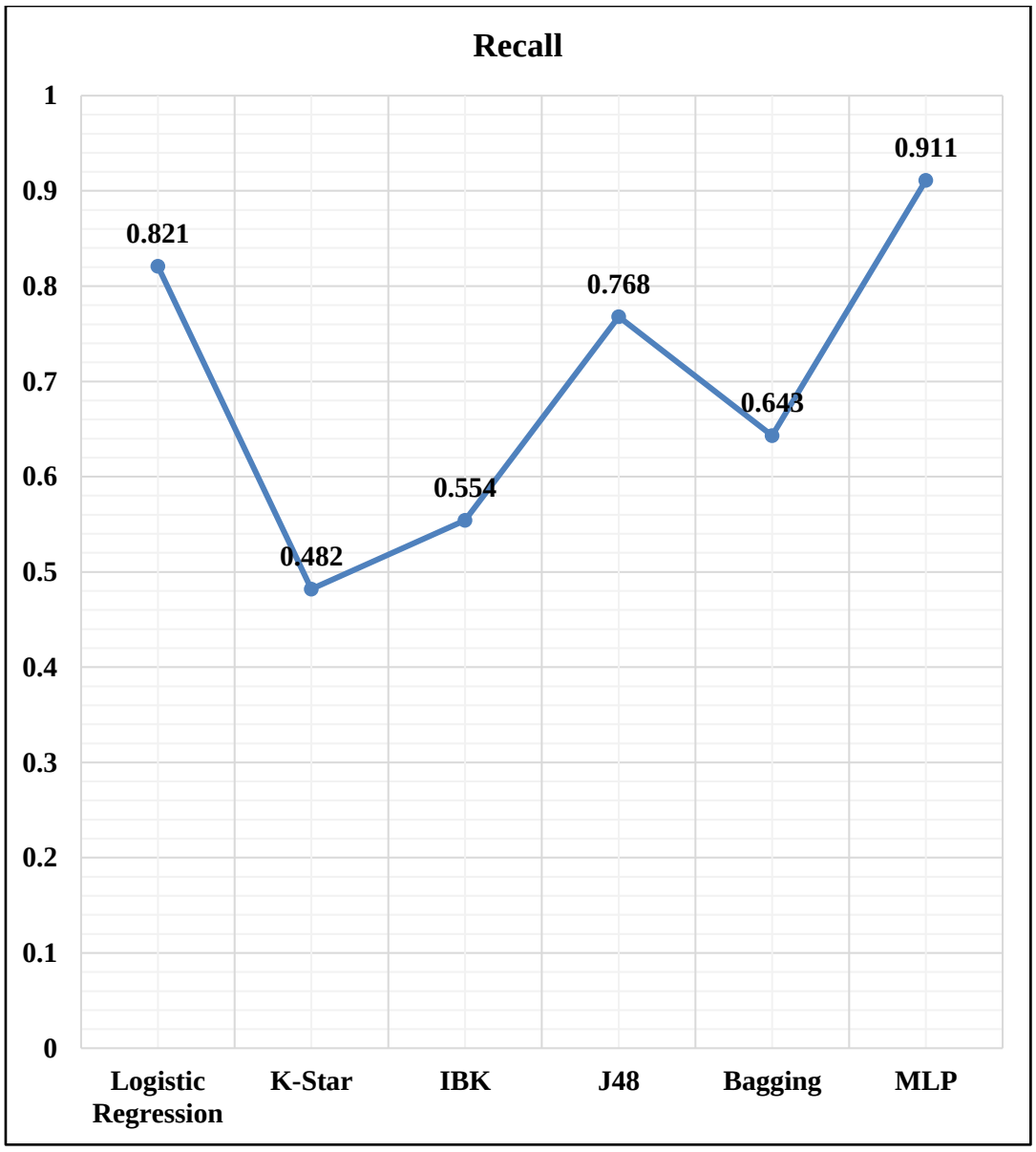

Figure 5.17: Recall Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Results as shown in figure 5.17 confirm that at configuration setting of cross-validation 20 folds the Recall of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.821. The other classification algorithms had to have a Recall of about 0.76 in case of J48 classifier, 0.64, 0.55, and 0.48 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure Recall were found to be MLP and LR.

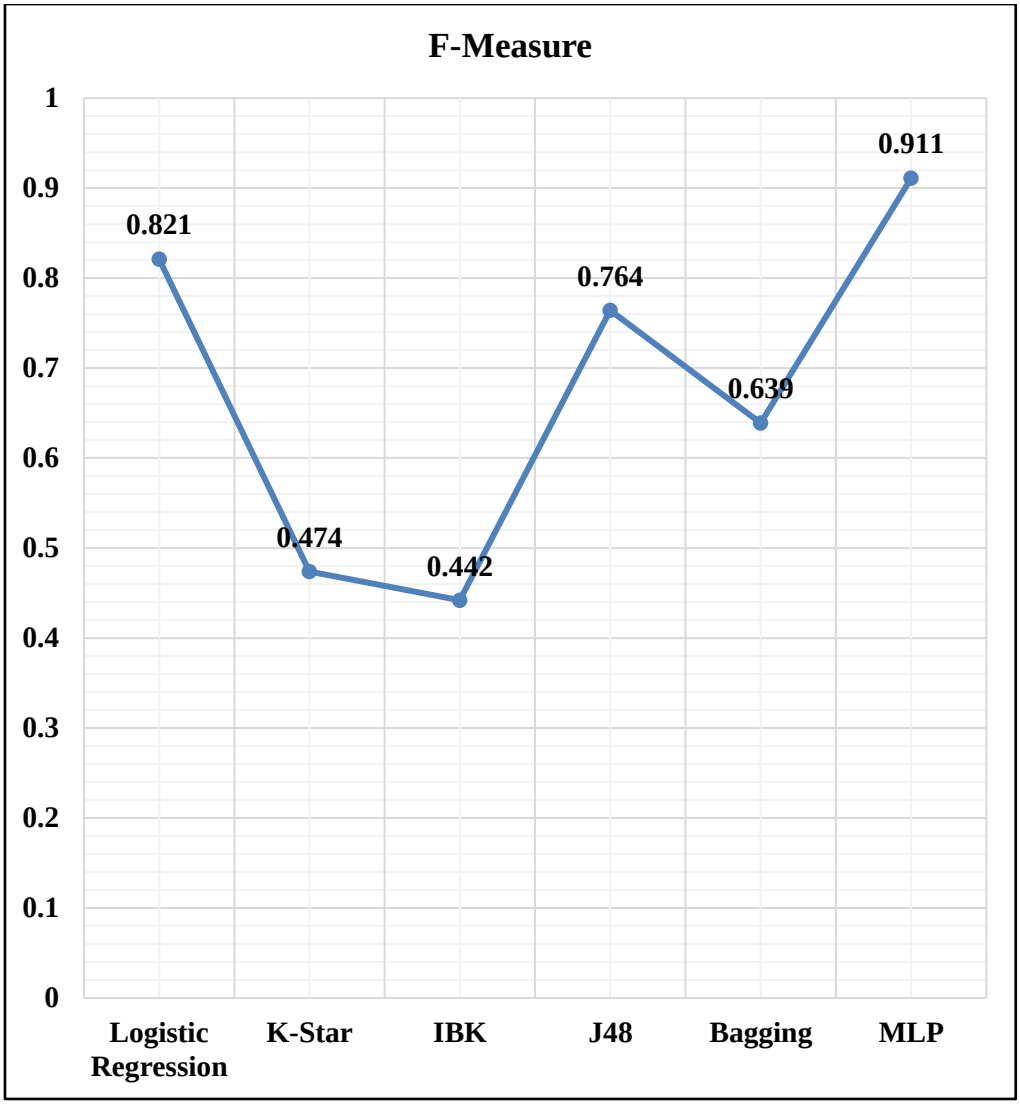

Figure 5.18: F-Measure Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

According to figure 5.18, it can be concluded that at the configuration setting of cross-validation 20 folds the F-Measure of MLP classifier with value 0.91 is found to be highest followed by Logistic Regression with value 0.82. The other classification algorithms had to have an F-Measure of about 0.76 in case of J48 classifier, 0.63, 0.44, and 0.47 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure F-Measure were found to be MLP and LR.

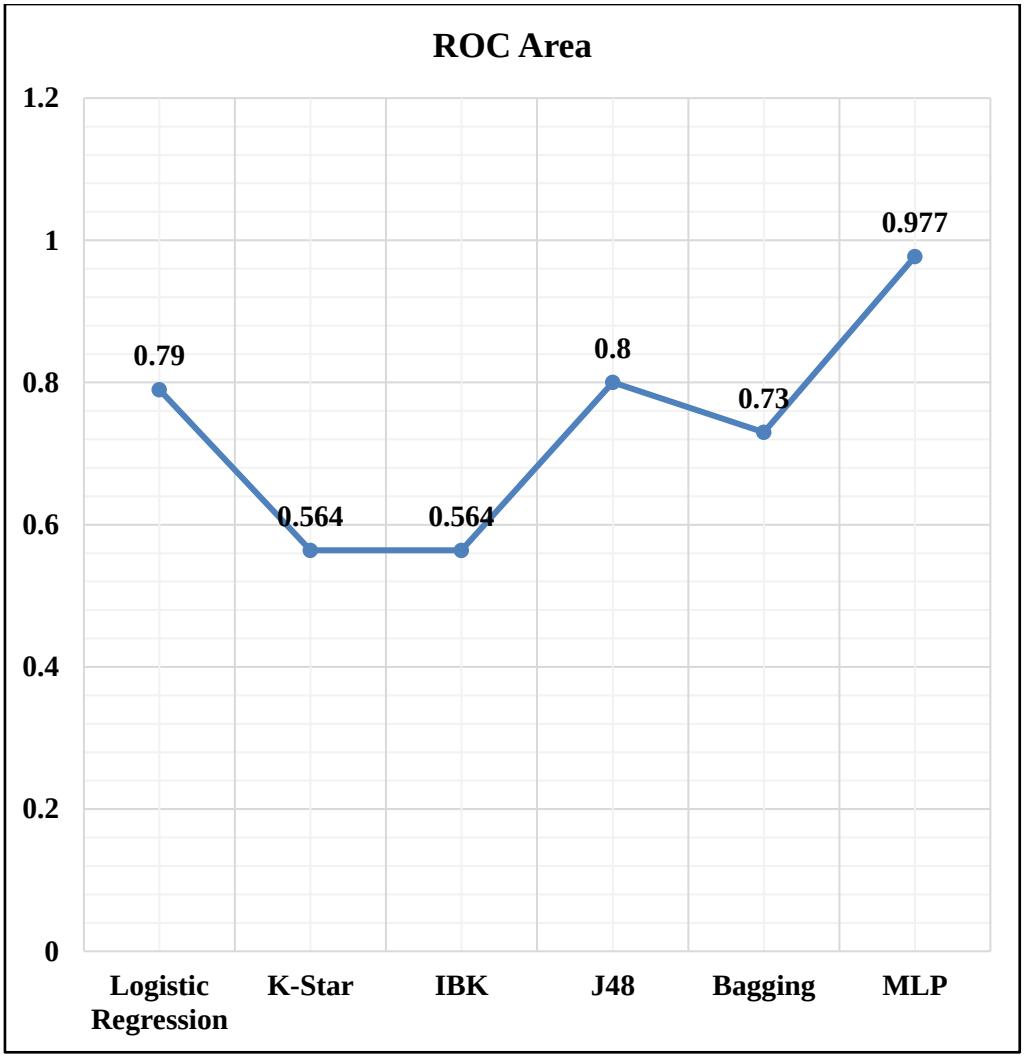

Figure 5.19: ROC Area Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.19, it can be concluded that at the configuration setting of cross-validation 20 folds, the ROC Area of MLP classifier with value 0.97 is found to be highest followed by Logistic Regression with value 0.79. The other classification algorithms have ROC Area of about 0.8 in case of J48 classifier, 0.73, 0.56, and 0.56 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure ROC Area were found to be MLP and LR.

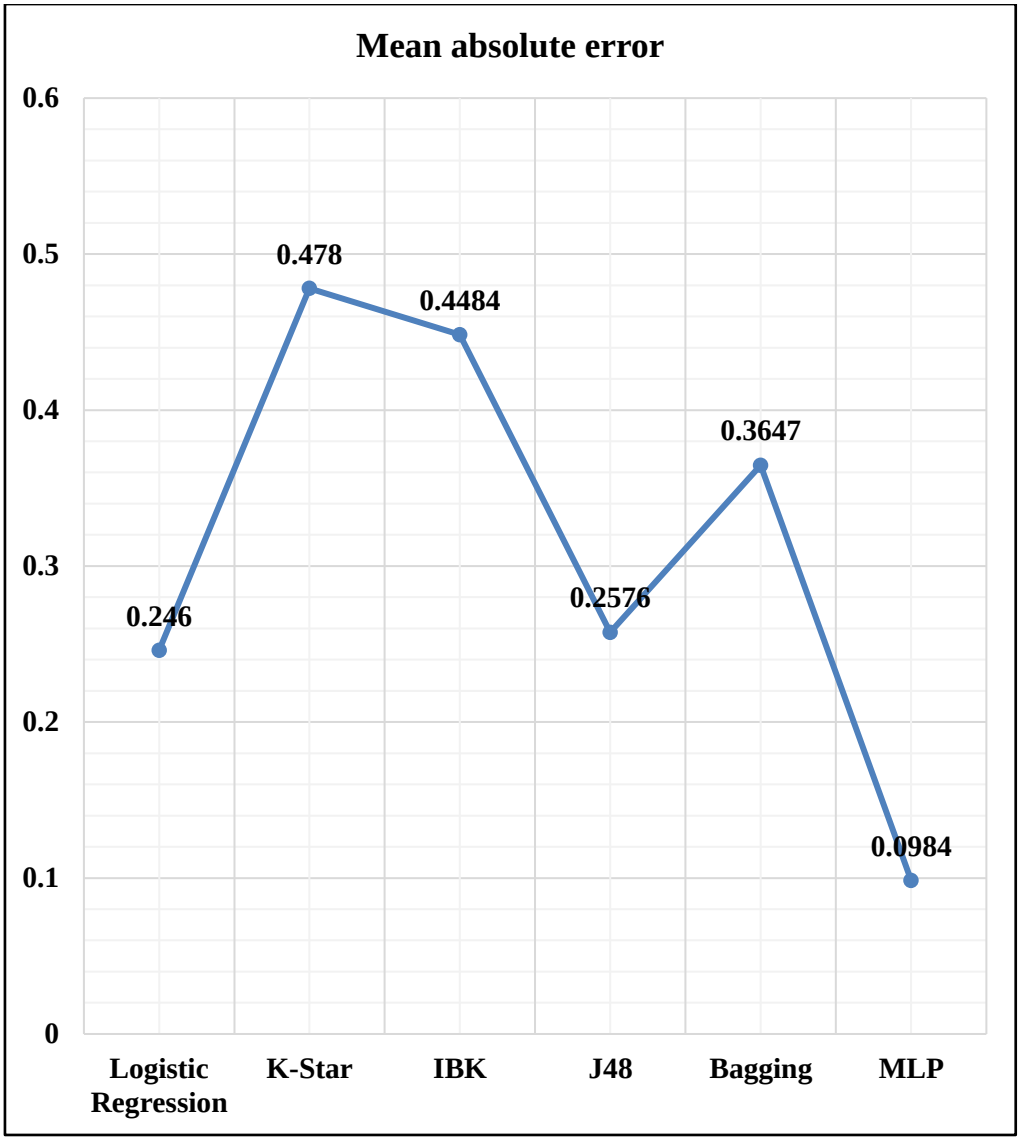

Figure 5.20: MAE Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.20, it can be concluded that at the configuration setting of cross-validation, 20 folds the mean absolute error value of MLP classifier with 0.09 is found to be the lowest followed by Logistic Regression with value 0.24. The other classification algorithms were having mean absolute error value of about 0.25 in case of J48 classifier, 0.36, 0.44, and 0.47 in case of Bagging, IBK, and K-Star were found to be quite high. The most appropriate classifiers based on performance measure mean absolute error value were found to be MLP and LR having lesser mean absolute error values as compared to others.

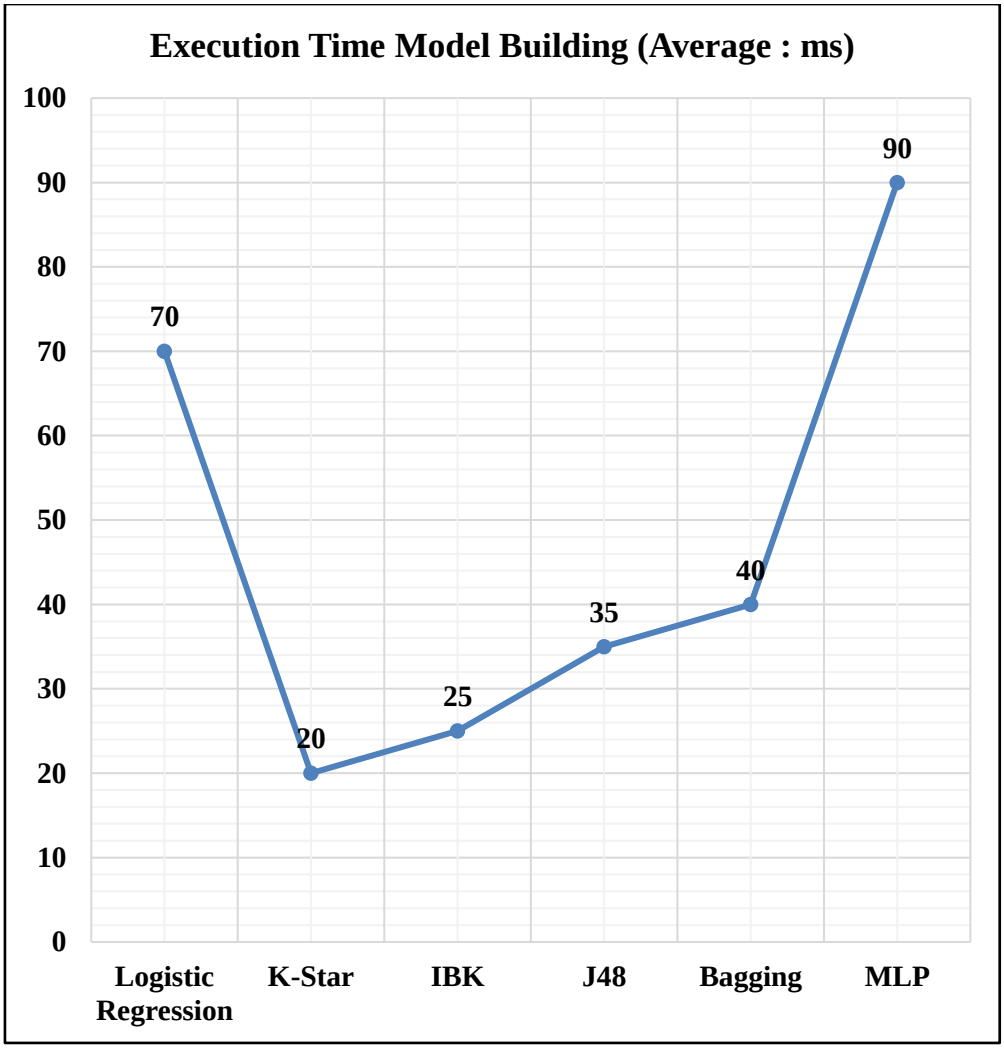

Figure 5.21: Average Execution Time for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Cross Validation-20 folds)

Figure 5.21, it can be concluded that at the configuration setting of cross-validation 20 folds, the average execution time of model building of K-Star and IBK classifier is found to be 20 and 25 milliseconds respectively which is quite less as compared with other classifiers. The other classification algorithms had to have an average execution time of model building of about 35ms in case of J48 classifier, 40, 70, and 90 ms in case of Bagging, Logistic Regression, and MLP. The most appropriate classifiers based on performance measure average execution time of model building were found to be K-Star and IBK.

5.5 Comparative Analysis Based on Split 33%

Classifier mode – Percentage Split Method – 33%: In the performance analysis of classification algorithms used for task offloading with the Percentage Split Method (also known as the Holdout Method) using a split ratio of 33%, the dataset is divided into a training set comprising 67% of the data and a testing set comprising 33% of the data. The training set is used to train the classification algorithm, and the testing set is used to evaluate its performance.

Table 5.3: Performance Analysis of Classification Algorithms Used for Task Offloading: Percentage Split Method – 33%

Performance Measure	Logistic Regression	K-Star	IBK	J48	Bagging	MLP
Accuracy	0.76	0.44	0.73	0.73	0.47	0.68
Kappa Statistic	0.52	-0.10	0.47	0.47	-0.05	0.36
TP Rate	0.76	0.44	0.73	0.73	0.47	0.68
FP Rate	0.23	0.55	0.26	0.26	0.52	0.31
Precision	0.83	0.44	0.74	0.78	0.47	0.72
Recall	0.76	0.44	0.73	0.73	0.47	0.68
F-Measure	0.74	0.43	0.73	0.72	0.47	0.67
ROC Area	0.86	0.57	0.73	0.73	0.45	0.77
Mean absolute error	0.23	0.44	0.28	0.26	0.51	0.30
Execution Time Model Building	30ms	35ms	30ms	35ms	30ms	60ms

Table 5.3, shows that by using a 33% split, a larger portion of the data is allocated to training, which allows the algorithm to learn patterns and relationships within the data. However, the testing set is still substantial enough to provide a good assessment of the algorithm's generalization and performance on unseen data. The results of the evaluation are shown below in the figure.

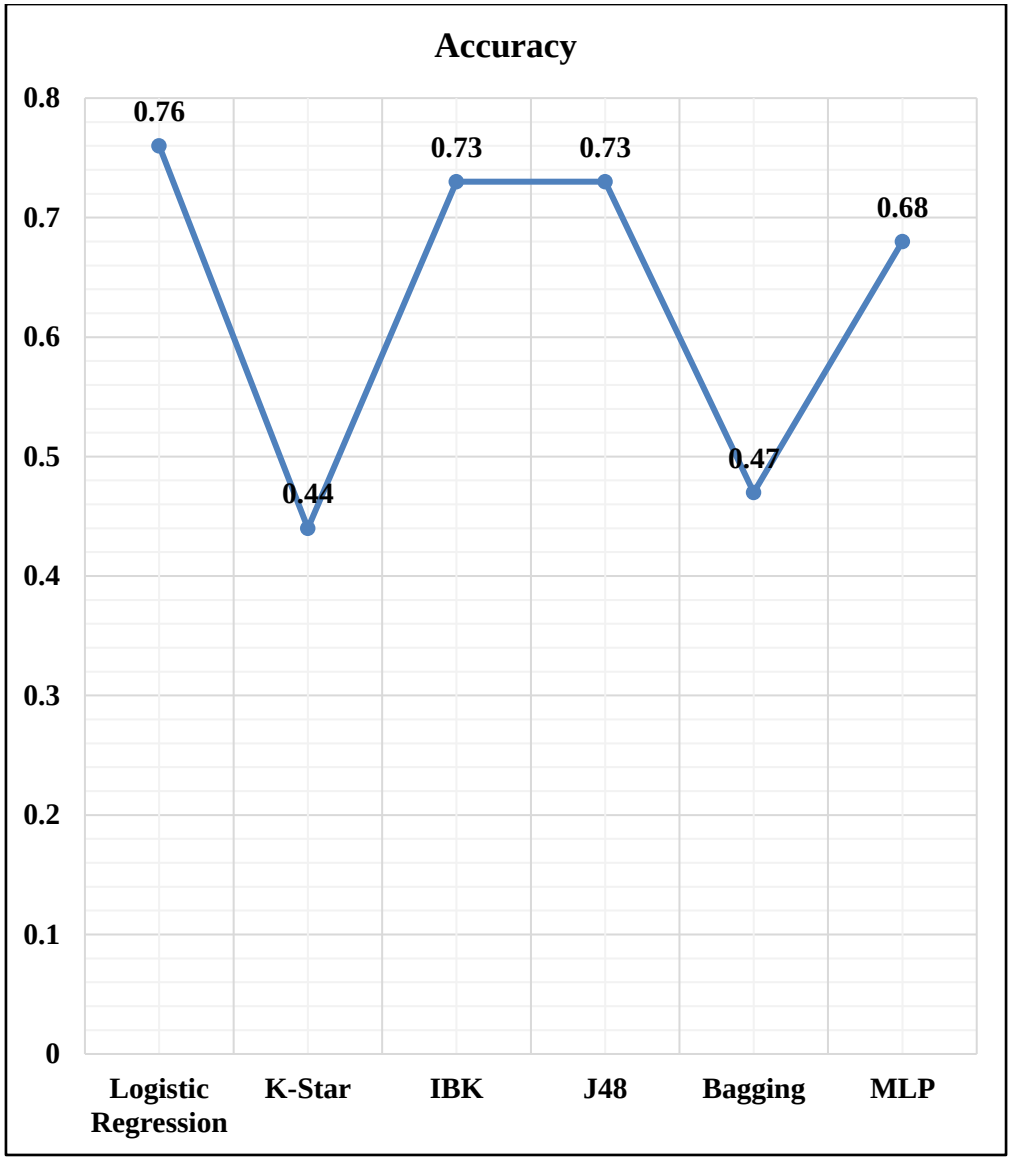

Figure 5.22: Accuracy Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.22, confirms that at the configuration setting of split 33% the accuracy of the Logistic Regression classifier with value 0.76 is found to be highest followed by IBK and J48 with values 0.73 respectively. The other classification algorithms had to have an accuracy of about 0.47 in case of Bagging classifier, 0.44 in case of K-Star. The most appropriate classifier based on performance measure accuracy was found to be Logistic Regression is 0.76 as compared with others.

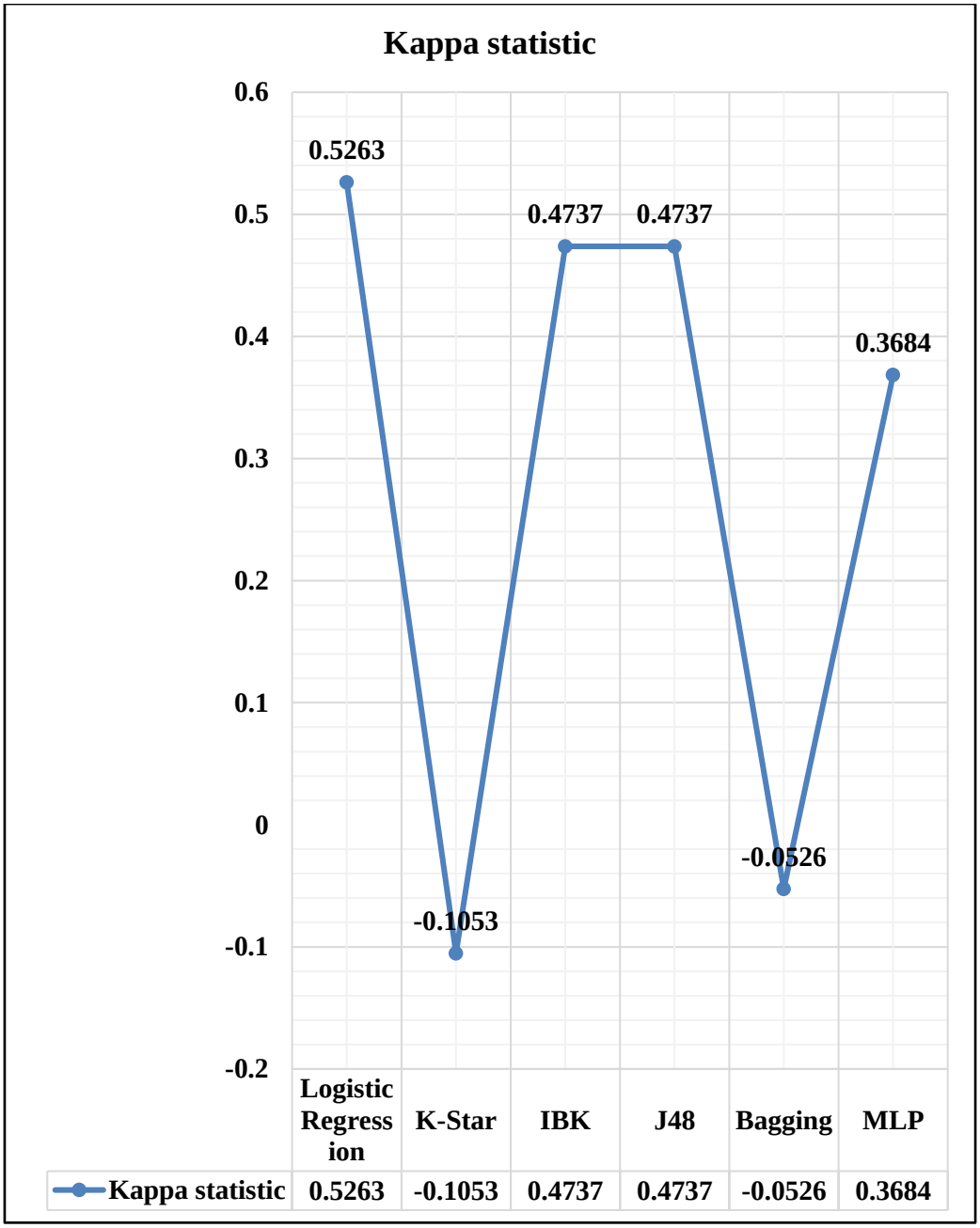

Figure 5.23: Kappa Statistics Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.23, shows that comparing the performance of classifiers used for task offloading and resource allocation in SMART FOG environment based on Kappa statistics it was found that Logistic Regression with value 0.52 was better as compared to other classifiers with Kappa statistics values 0.47, 0.47, 0.36, -0.05, and -0.10 for IBK, J48, MLP, Bagging, and K-Star respectively.

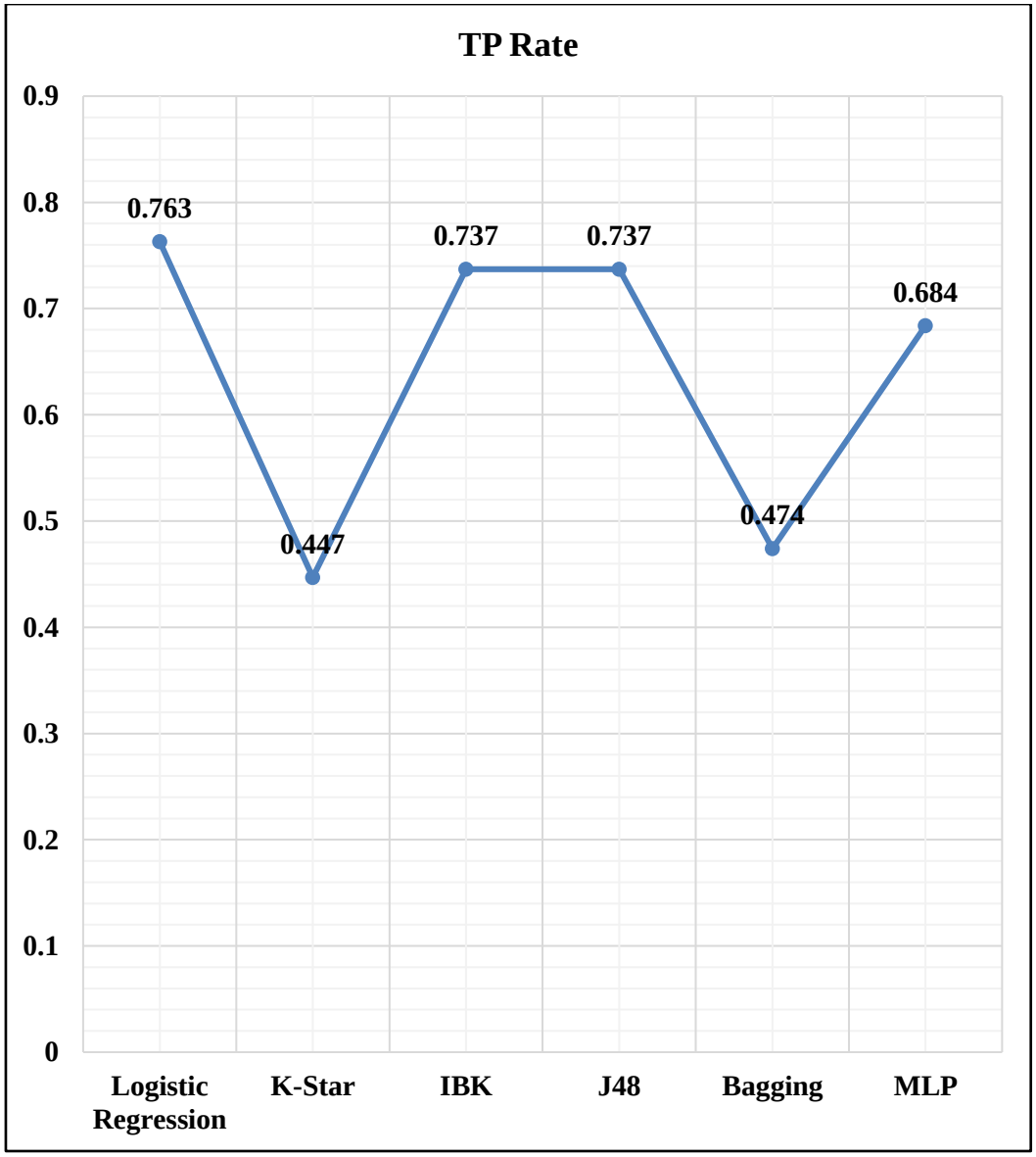

Figure 5.24: TP Rate Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

According to figure 5.24, it can be concluded that at configuration setting spilt 33%, the TP Rate of Logistic Regression classifier with value 0.76 is found to be highest followed by IBK and J48 with value 0.73. The other classification algorithms had to have a TP Rate of about 0.68 in case of MLP classifier, 0.47 and 0.44 for Bagging, and K-Star respectively. The most appropriate classifier based on performance measure TP rate was found to be Logistic Regression is 0.76.

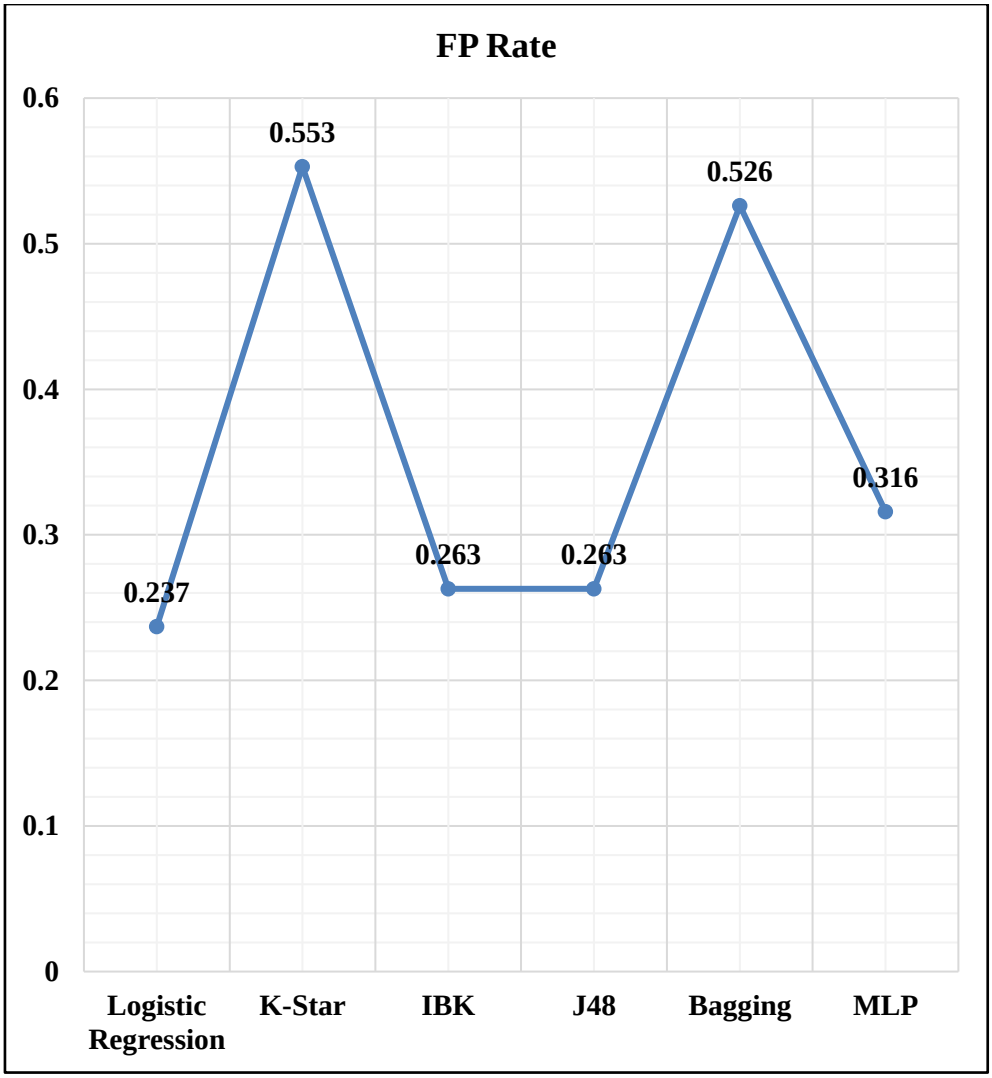

Figure 5.25: FP Rate Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.25, it can be concluded that at a configuration setting split 33%, the FP Rate of the Logistic Regression classifier with a value 0.23 is found to be the lowest followed by IBK and J48 with a value 0.26. The other classification algorithms had to have an FP Rate of about 0.31 in case of MLP classifier, 0.52, and 0.55 in case of Bagging and K-Star had to be quite high. The most appropriate classifiers based on performance measure FP rate were found to be Logistic Regression, IBK and J48 having lesser FP rate is 0.55 values as compared to others.

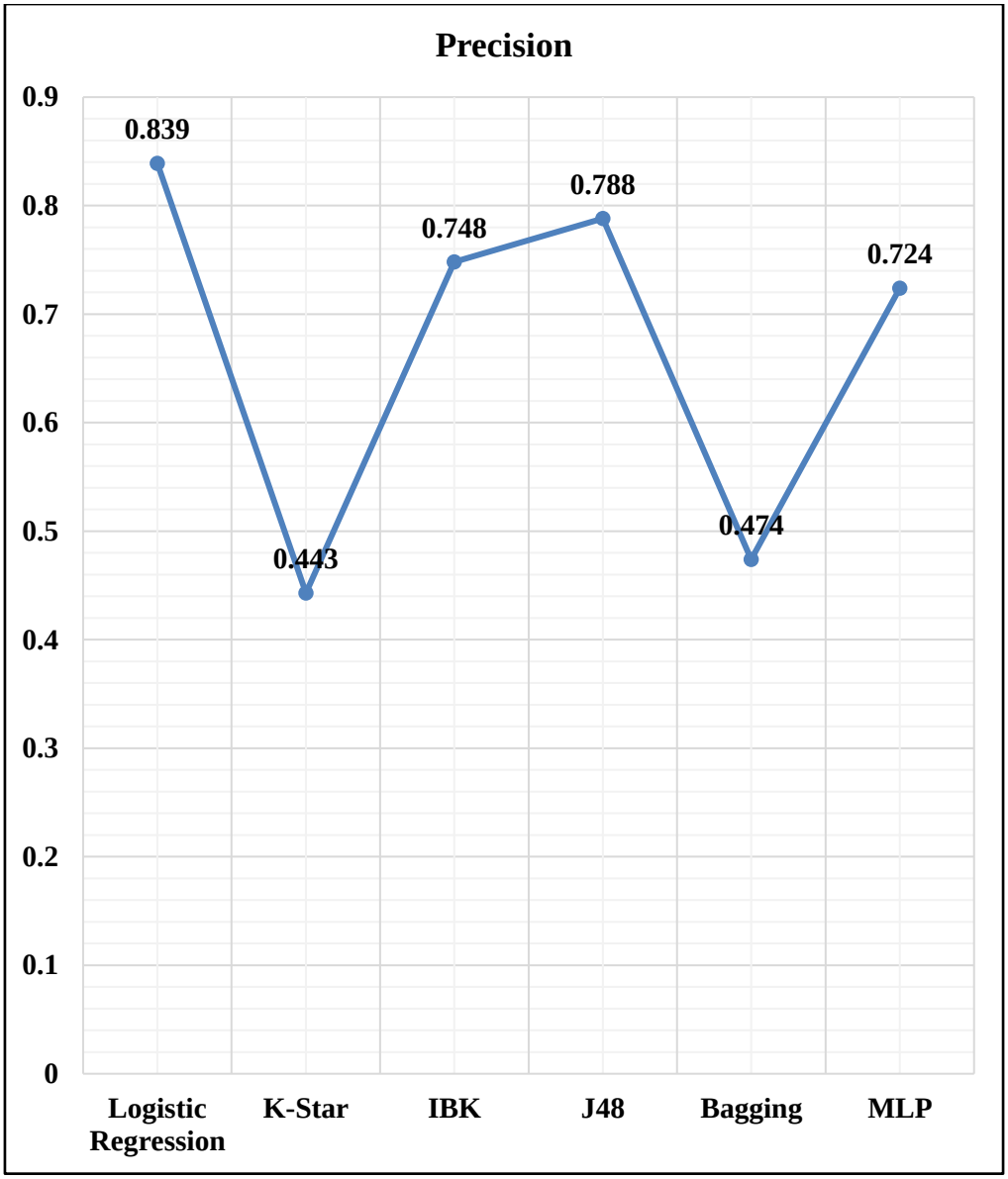

Figure 5.26: Precision Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

According to figure 5.26, it can be concluded that at the configuration setting spilt 33%, the Precision score of the Logistic Regression classifier with value 0.83 is found to be highest followed by IBK and J48 with value 0.74 and 0.78 respectively. The other classification algorithms had to have a Precision of about 0.72 in case of MLP classifier, 0.47 and 0.44 for Bagging, and K-Star respectively. The most appropriate classifier based on performance measure Precision was found to be Logistic Regression is 0.83.

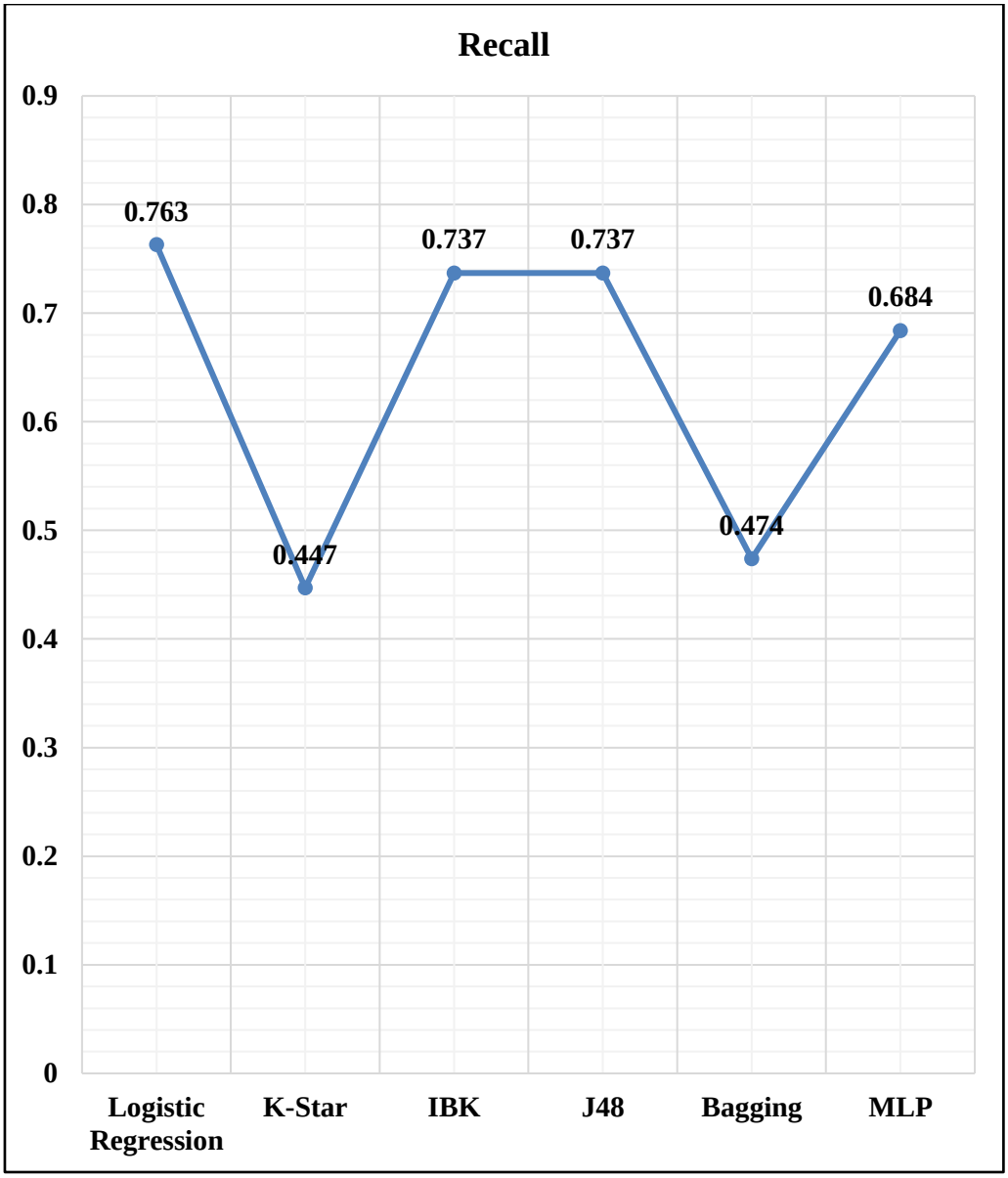

Figure 5.27: Recall Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.27, it can be concluded that at configuration setting spilt 33%, the Recall score of the Logistic Regression classifier with value 0.76 is found to be highest followed by IBK and J48 with value 0.73. The other classification algorithms had to have a Recall of about 0.68 in case of MLP classifier, 0.47 and 0.44 for Bagging, and K-Star respectively. The most appropriate classifier based on performance measure Recall was found to be LR is 0.76.

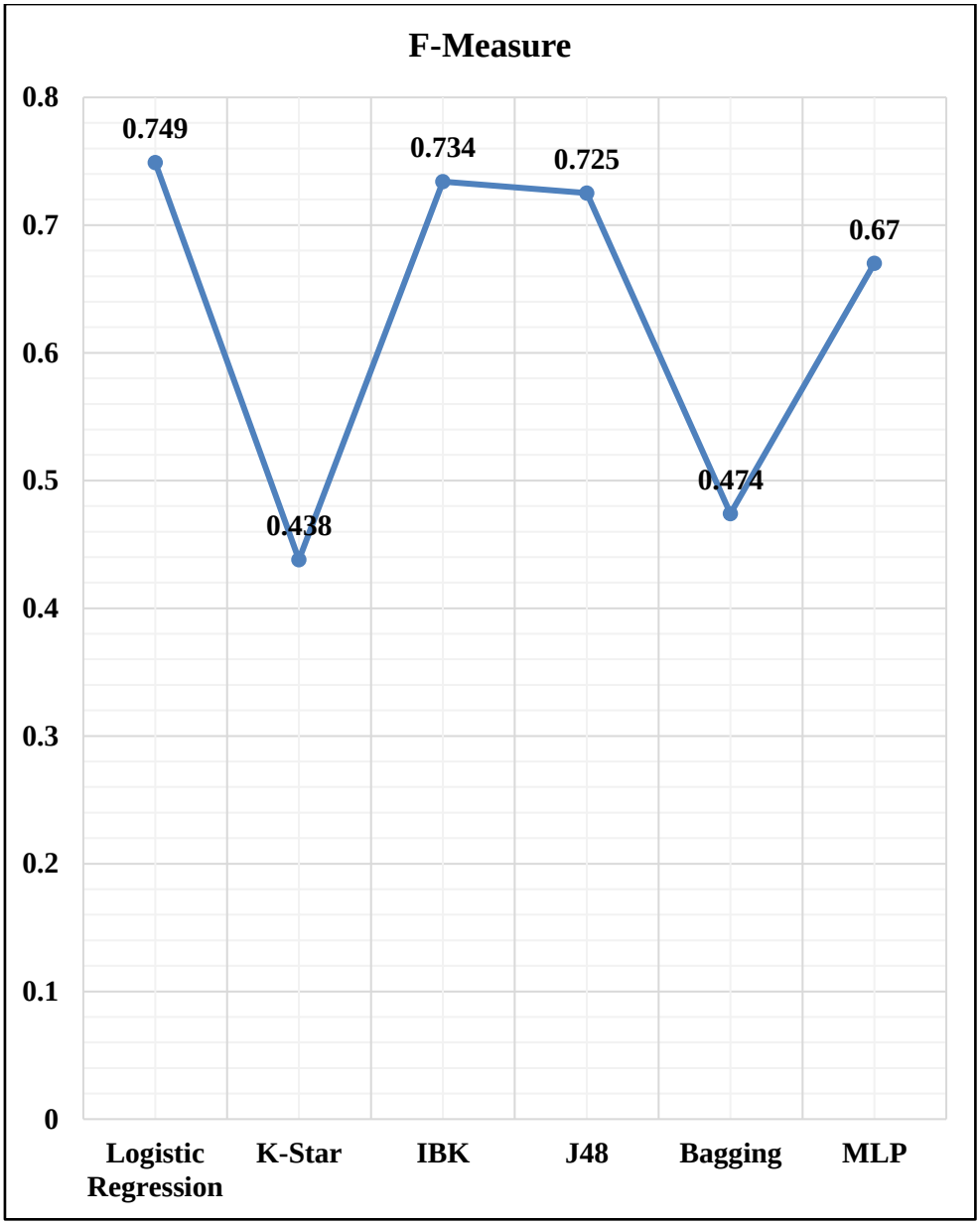

Figure 5.28: F-Measure Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

According to figure 5.28, it can be concluded that at configuration setting spilt 33%, the F-Measure score of the Logistic Regression classifier with value 0.74 is found to be highest followed by IBK and J48 with value 0.73 and 0.72 respectively. The other classification algorithms had to have an F-Measure of about 0.67 in case of MLP classifier, 0.47 and 0.43 for Bagging, and K-Star respectively. The most appropriate classifier based on performance measure F-Measure was found to be LR is 0.74.

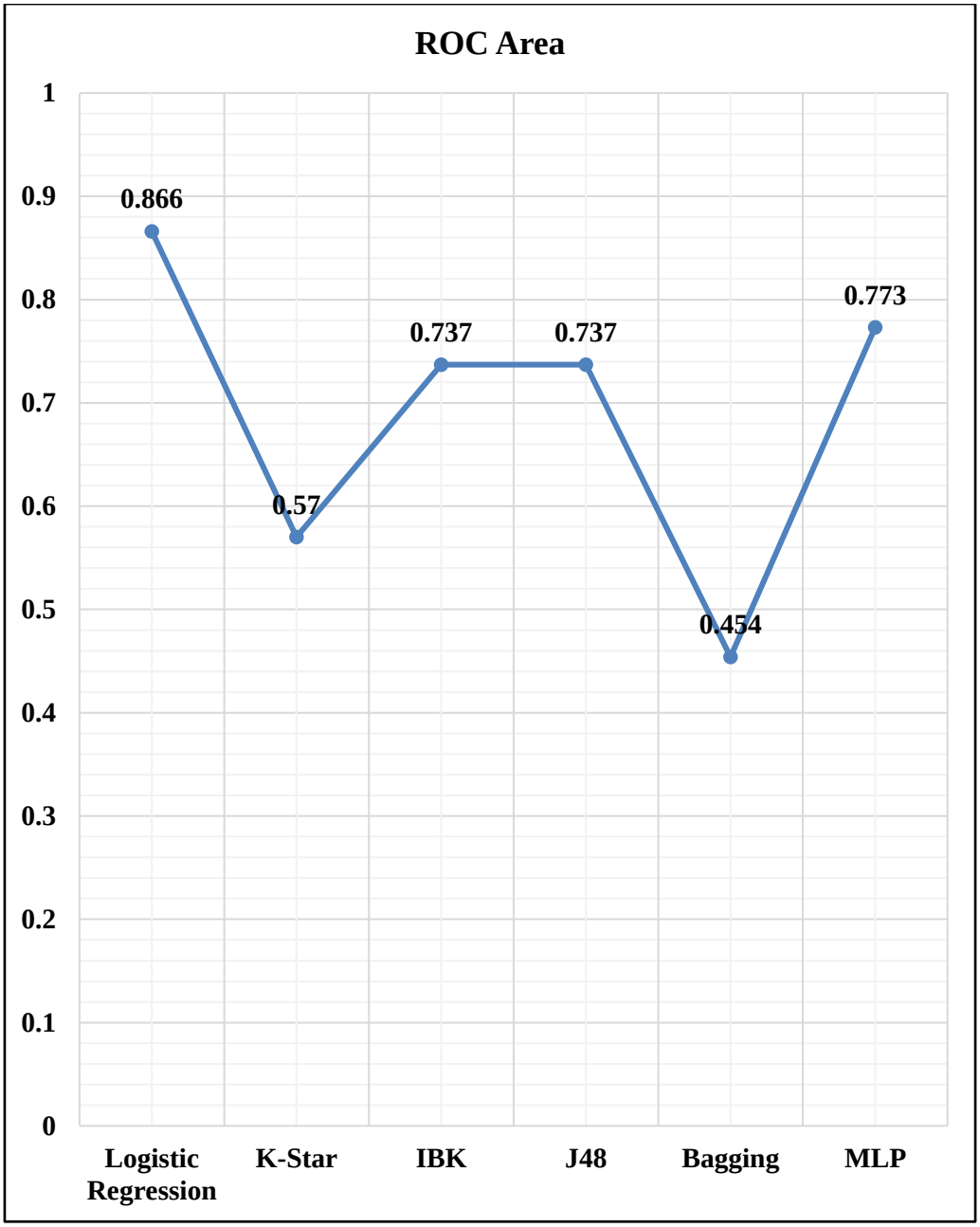

Figure 5.29: ROC Area Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.29, it can be concluded that at configuration setting spilt 33%, the ROC Area score of the Logistic Regression classifier with value 0.86 is found to be highest followed by MLP, IBK and J48 with value 0.77, 0.73 and 0.73 respectively. The other classification algorithms had ROC Area scores of about 0.57, 0.45 for K-Star and Bagging respectively. The most appropriate classifier based on performance measure ROC Area score was found to be LR is 0.86.

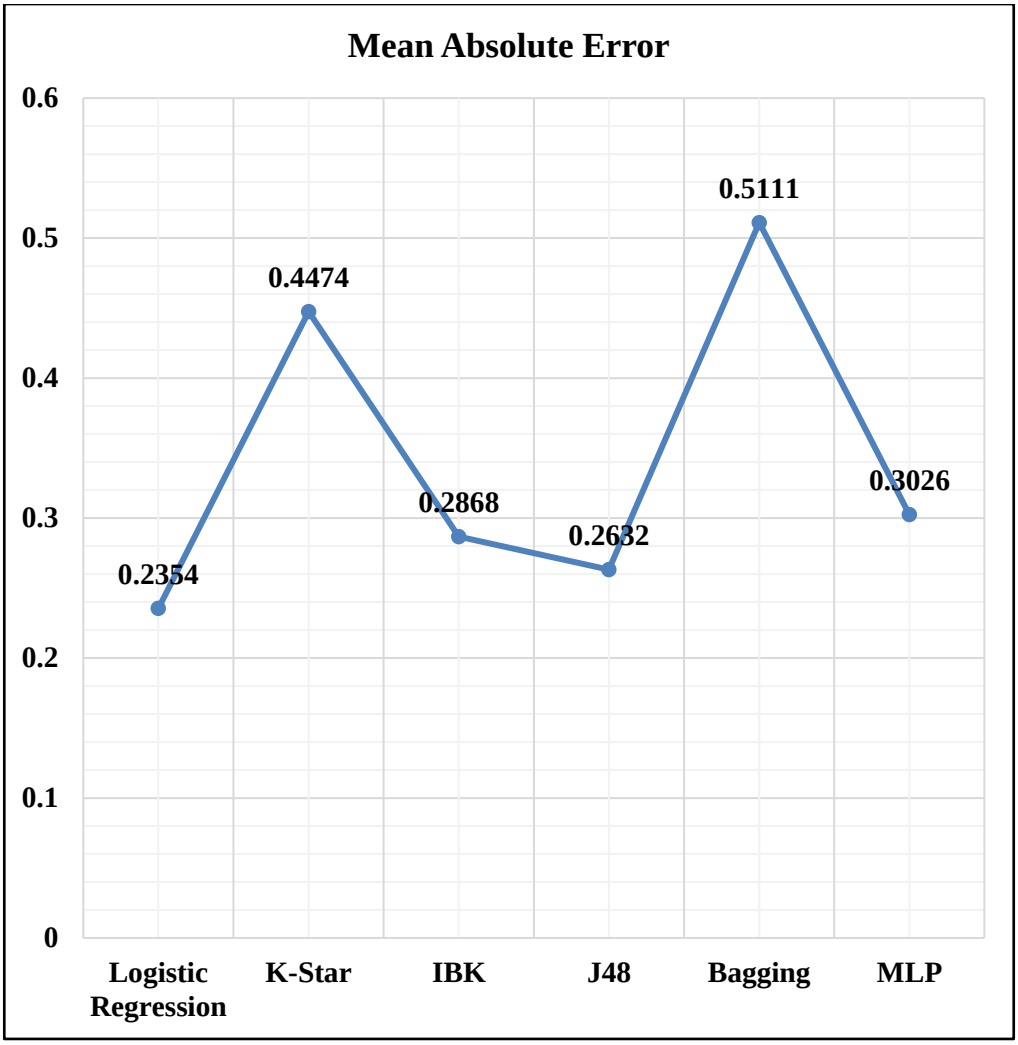

Figure 5.30: MAE Value for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.30, it can be concluded that at the configuration setting split 33%, the Mean Absolute Error of the Logistic Regression classifier with value 0.23 is found to be the lowest followed by IBK and J48 with value 0.28 and 0.26 respectively. The other classification algorithms had MAE value of about 0.30 in case of MLP classifier, 0.44 and 0.51 in case of Bagging and K-Star had to be quite high. The most appropriate classifiers based on performance measure MAE value were found to be LR, IBK and J48 having lesser MAE values as compared to others.

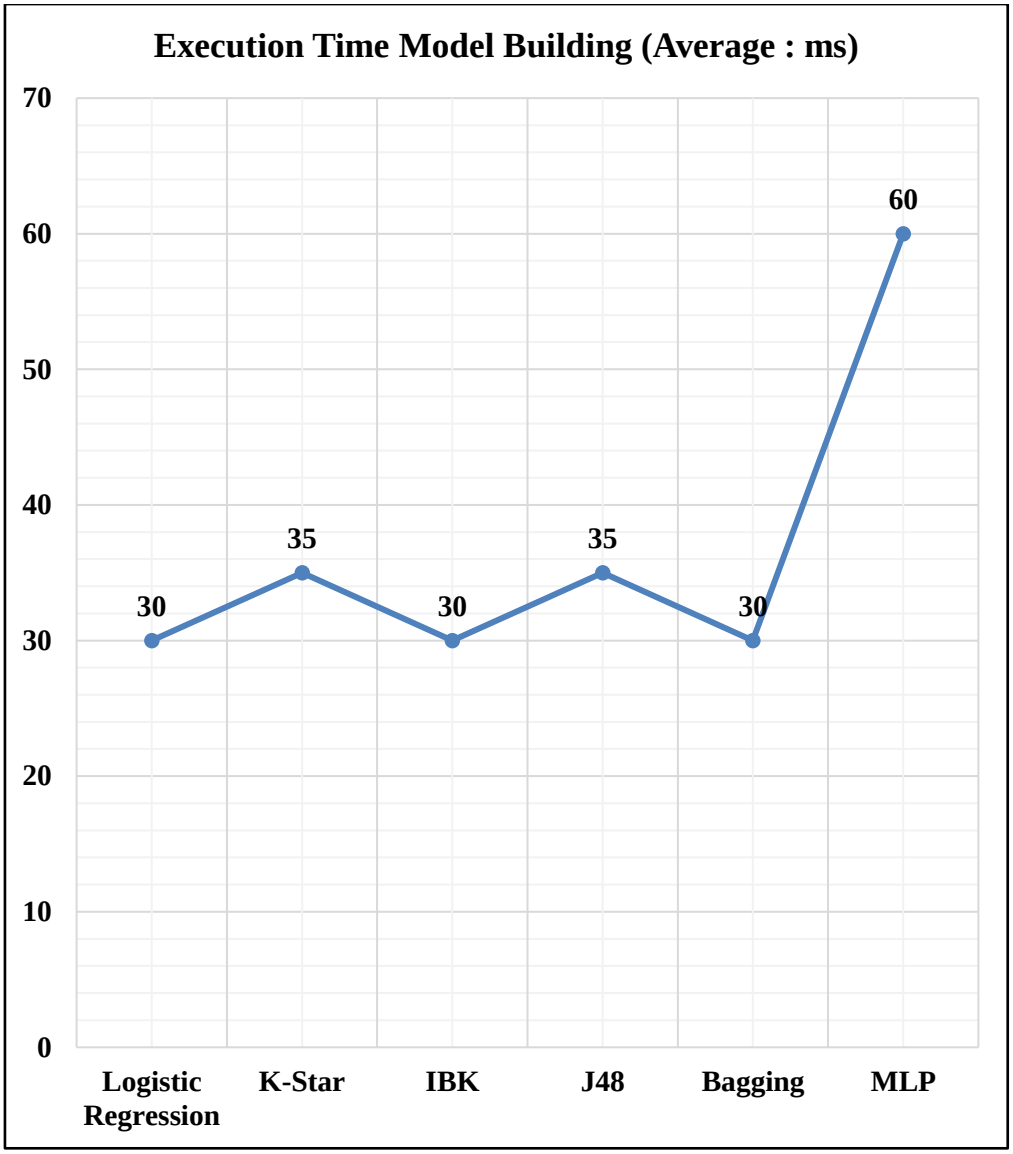

Figure 5.31: Average Execution Time for Classifiers Used in Task Offloading and Resource Management (Configuration Setting: Split-33%)

Figure 5.31, it can be concluded that at configuration setting spilt 33% the average execution time of model building of Logistic Regression, IBK and Bagging classifier were found to be 30 milliseconds for each which is quite less as compared with other classifiers. The other classification algorithms had to have an average execution time of model building of about 35ms in case of J48 and K-Star classifier, and 60ms in case of MLP. The most appropriate classifiers based on performance measure average execution time of model building were found to be Logistic Regression, IBK and Bagging.

5.6 Overall Performance of Classification Algorithms

The overall performance of classification algorithms in task offloading and resource allocation for IoT and fog computing is an active area of research and development. Various algorithms, including decision trees, random forest, support vector machines, K-nearest neighbors, neural networks, naive Bayes, and logistic regression, have been explored for these tasks, each with its strengths and weaknesses.

Table 5.4: Overall Performance of Classification Algorithms Used for Task Offloading

Type of Performance Measure	Logistic Regression	K-Star	IBK	J48	Bagging	MLP
Accuracy	0.80	0.48	0.61	0.75	0.60	0.83
Kappa Statistic	0.60	-0.02	0.23	0.50	0.21	0.67
TP Rate	0.80	0.49	0.62	0.75	0.49	0.84
FP Rate	0.20	0.51	0.39	0.25	0.53	0.16
Precision	0.83	0.49	0.76	0.79	0.48	0.85
Recall	0.80	0.49	0.62	0.75	0.61	0.84
F-Measure	0.80	0.48	0.54	0.74	0.60	0.83
ROC Area	0.82	0.58	0.62	0.77	0.63	0.91
Mean Absolute Error	0.23	0.45	0.39	0.27	0.58	0.17
Execution Time Model Building	53.33	25.00	25.00	33.33	33.33	76.67

Table 5.4, shows that in terms of the Kappa statistic, MLP had the highest value of 0.67, indicating good agreement between predicted and actual classes. J48 and Logistic Regression also showed substantial agreement with Kappa values of 0.50 and 0.60, respectively. However, IBK 0.23 and Bagging 0.21 had a moderate agreement, and K-Star had a negative Kappa statistic -0.02, suggesting lower agreement and potential issues with its performance. The evaluation demonstrates the varying success and limitations of each algorithm, with Logistic Regression and MLP performing relatively well in both accuracy and agreement metrics.

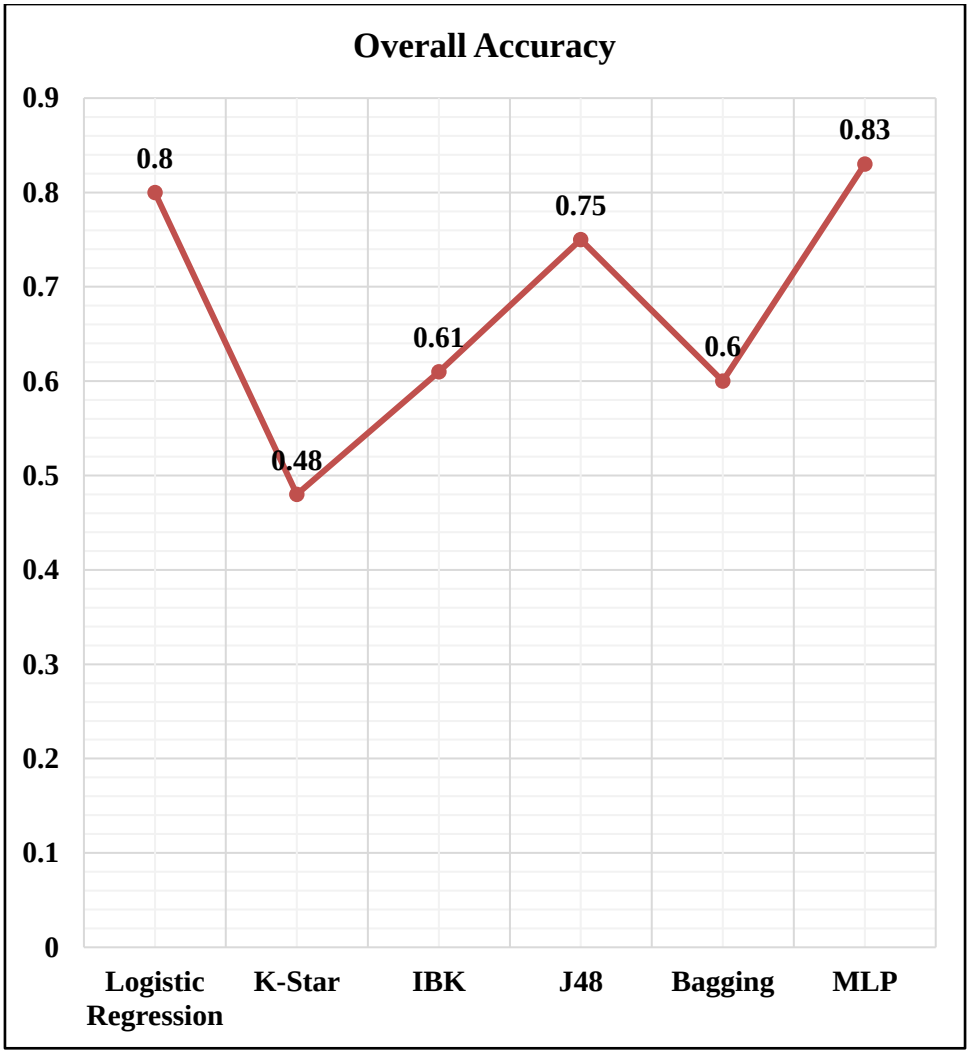

Figure 5.32: Overall Accuracy for Classifiers Used in Task Offloading and Resource Management

Figure 5.32, confirms that the overall accuracy of MLP classifier with value 0.83 is found to be the highest followed by Logistic Regression with value 0.80. The other classification algorithms had to have an overall accuracy of about 0.75 in case of J48 classifier, 0.60, 0.61, and 0.48 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure overall accuracy were found to be MLP and LR.

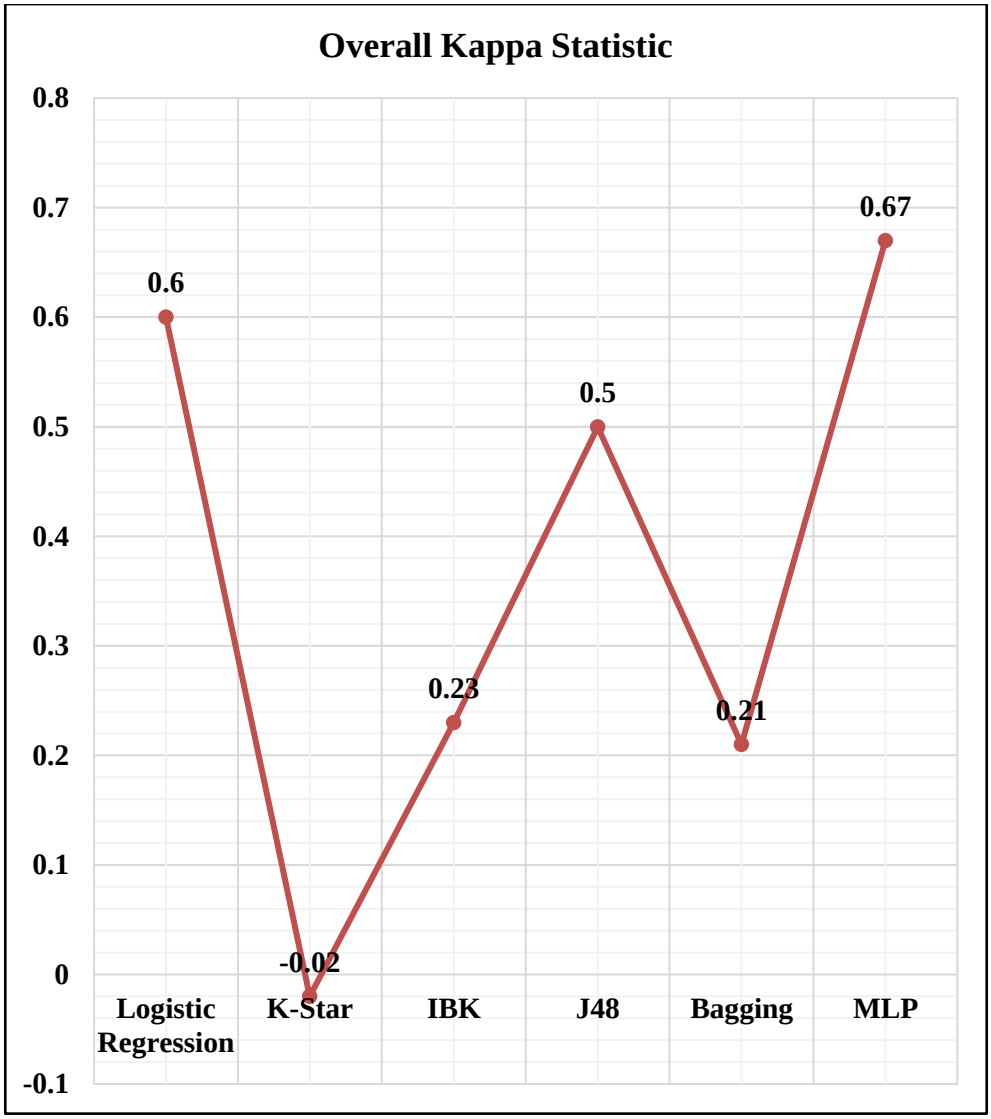

Figure 5.33: Overall, Kappa Statistics for Classifiers Used in Task Offloading and Resource Management

Figure 5.33, shows a comparison of the performance of classifiers based on overall Kappa statistics used for task offloading and resource allocation in case of SMART FOG environment it can be interpreted that higher overall Kappa statistics value of 0.67 in case of MLP and 0.6 in case of MLP and LR suggests that they are the better classifiers as compared to other techniques.

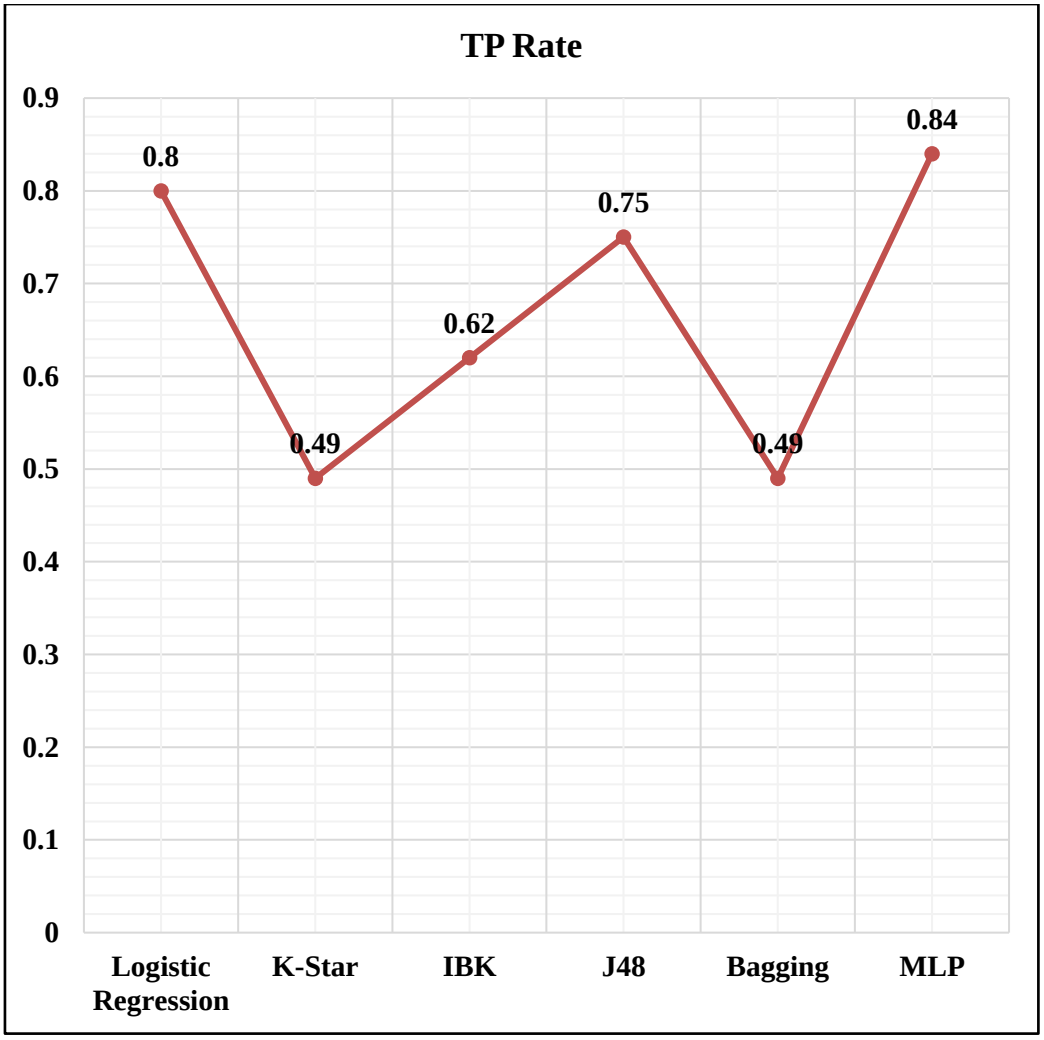

Figure 5.34: Overall TP Rate for Classifiers Used in Task Offloading and Resource Management

According to figure 5.34, it can be concluded that the overall TP Rate of MLP classifier with value 0.84 is found to be the highest followed by Logistic Regression with value 0.80. The other classification algorithms had to have an overall TP Rate of about 0.75 in case of J48 classifier, 0.49, 0.62, and 0.49 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure overall TP rate were found to be MLP and LR.

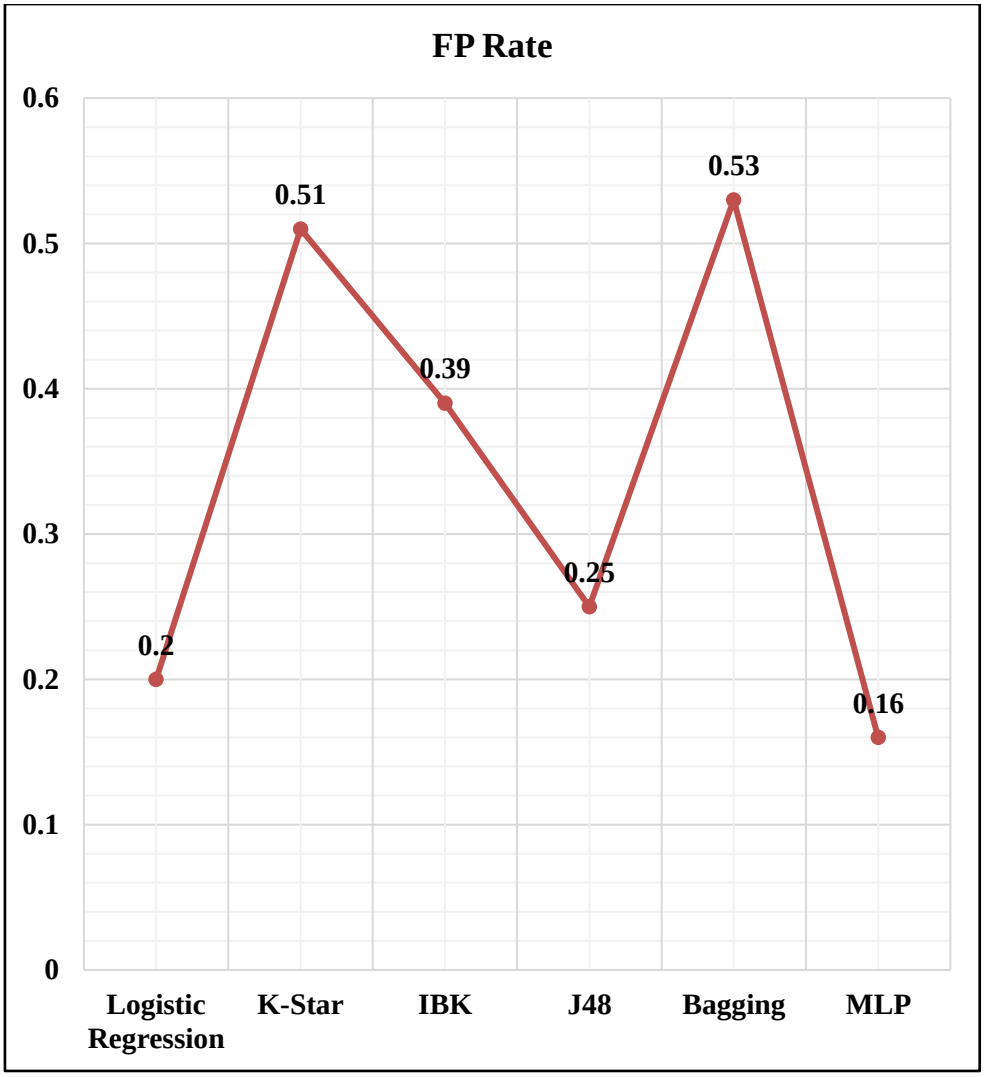

Figure 5.35: Overall FP Rate for Classifiers Used in Task Offloading and Resource Management

Figure 5.35, it can be concluded that the overall FP Rate of MLP classifier with value 0.16 is found to be lowest followed by Logistic Regression with value 0.2. The other classification algorithms had to have an overall FP Rate of about 0.25 in case of J48 classifier, 0,53, 0.39, and 0.51 in case of Bagging, IBK, and K-Star which were found to be quite higher. The most appropriate classifiers based on performance measure FP rate were found to be MLP and LR having lesser overall FP rate values as compared to others.

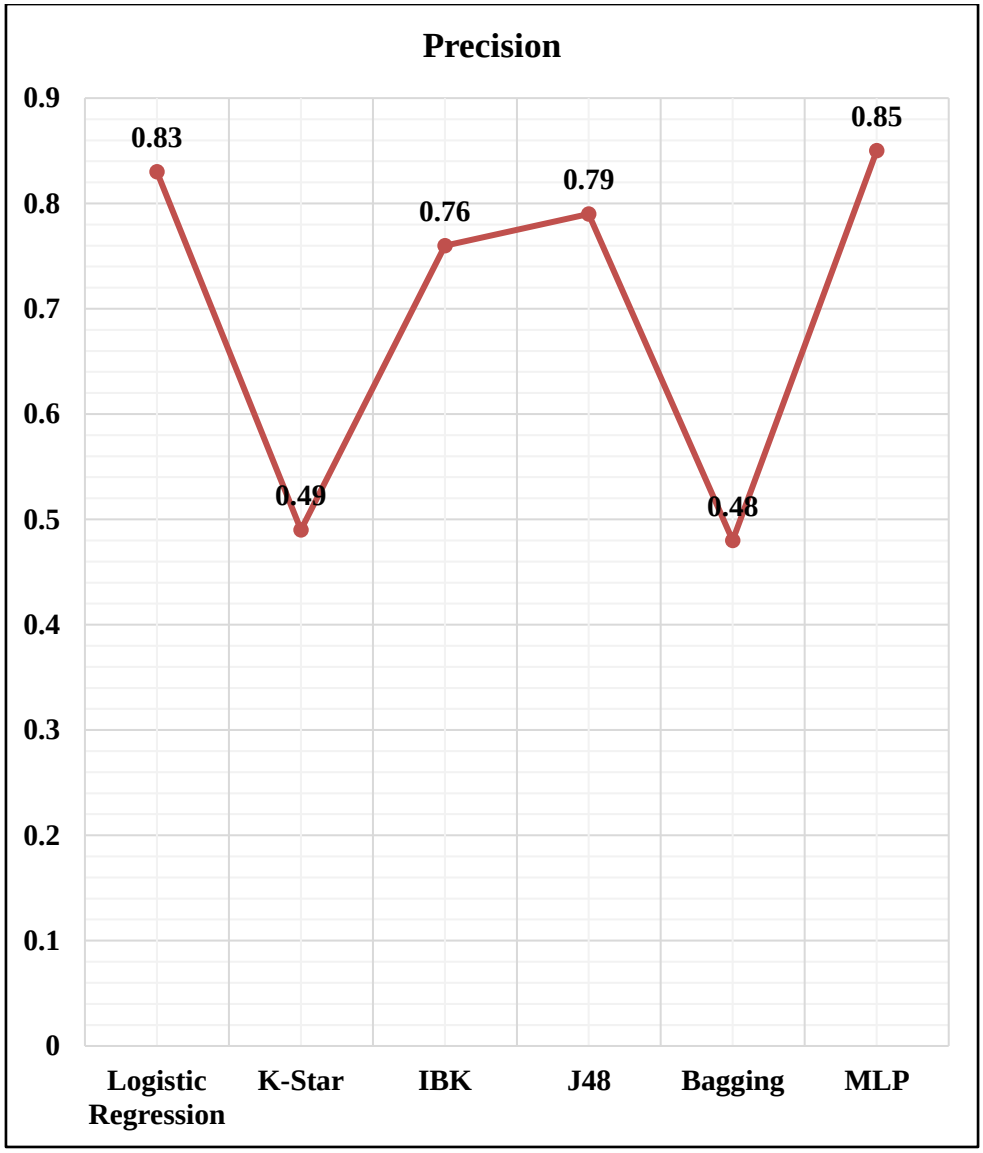

Figure 5.36: Overall Precision for Classifiers Used in Task Offloading and Resource Management

Figure 5.36, it can be concluded that the overall Precision of MLP classifier with value 0.85 is found to be highest followed by Logistic Regression with value 0.83. The other classification algorithms had to have an overall Precision of about 0.79 in case of J48 classifier, 0.48, 0.76, and 0.49 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure overall Precision were found to be MLP and LR.

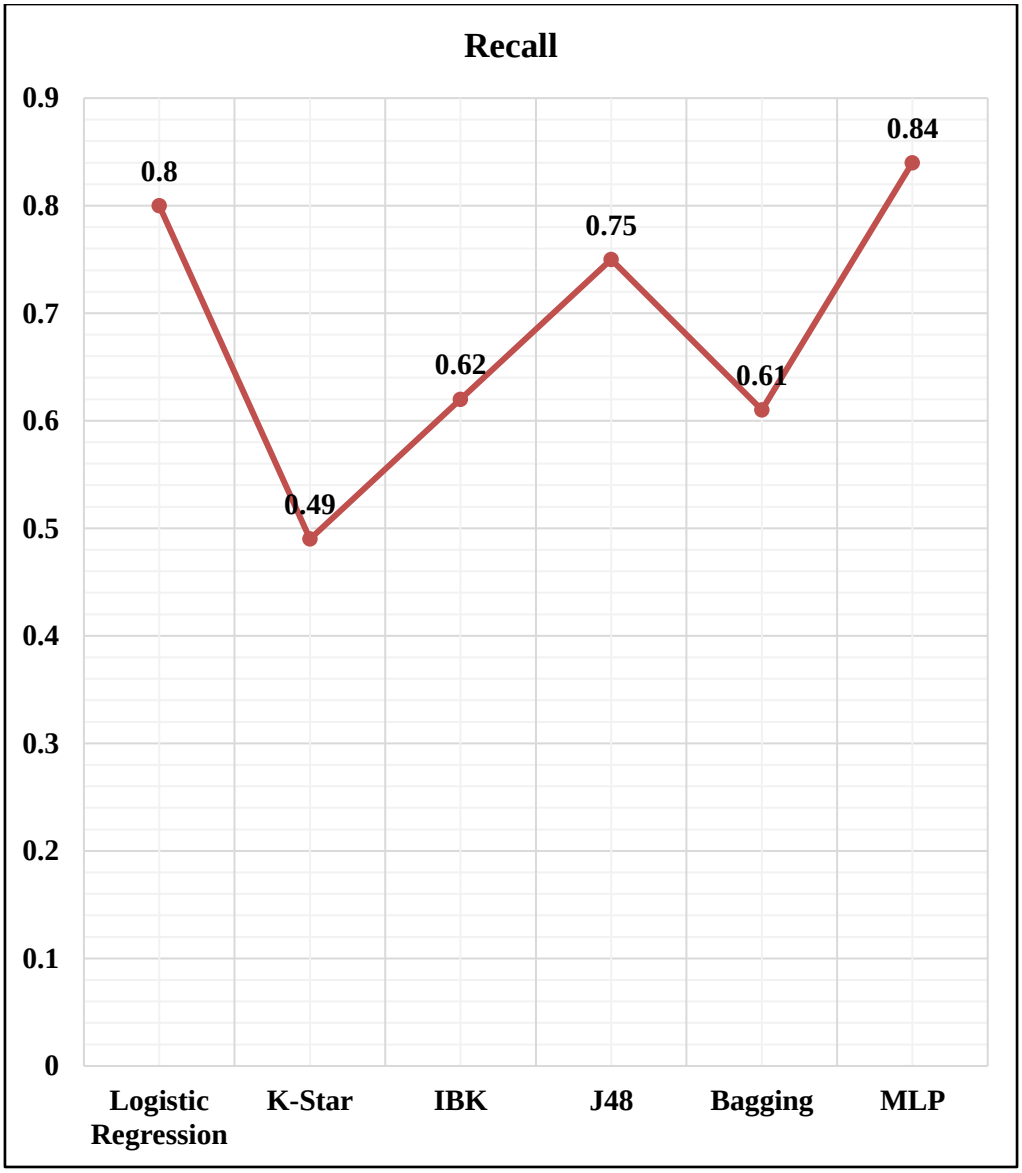

Figure 5.37: Overall Recall for Classifiers Used in Task Offloading and Resource Management

Results as shown in figure 5.37 confirm that the overall Recall of MLP classifier with value 0.84 is found to be the highest followed by Logistic Regression with value 0.80. The other classification algorithms had to have an overall Recall of about 0.75 in case of J48 classifier, 0.61, 0.62, and 0.49 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure overall Recall were found to be MLP and LR.

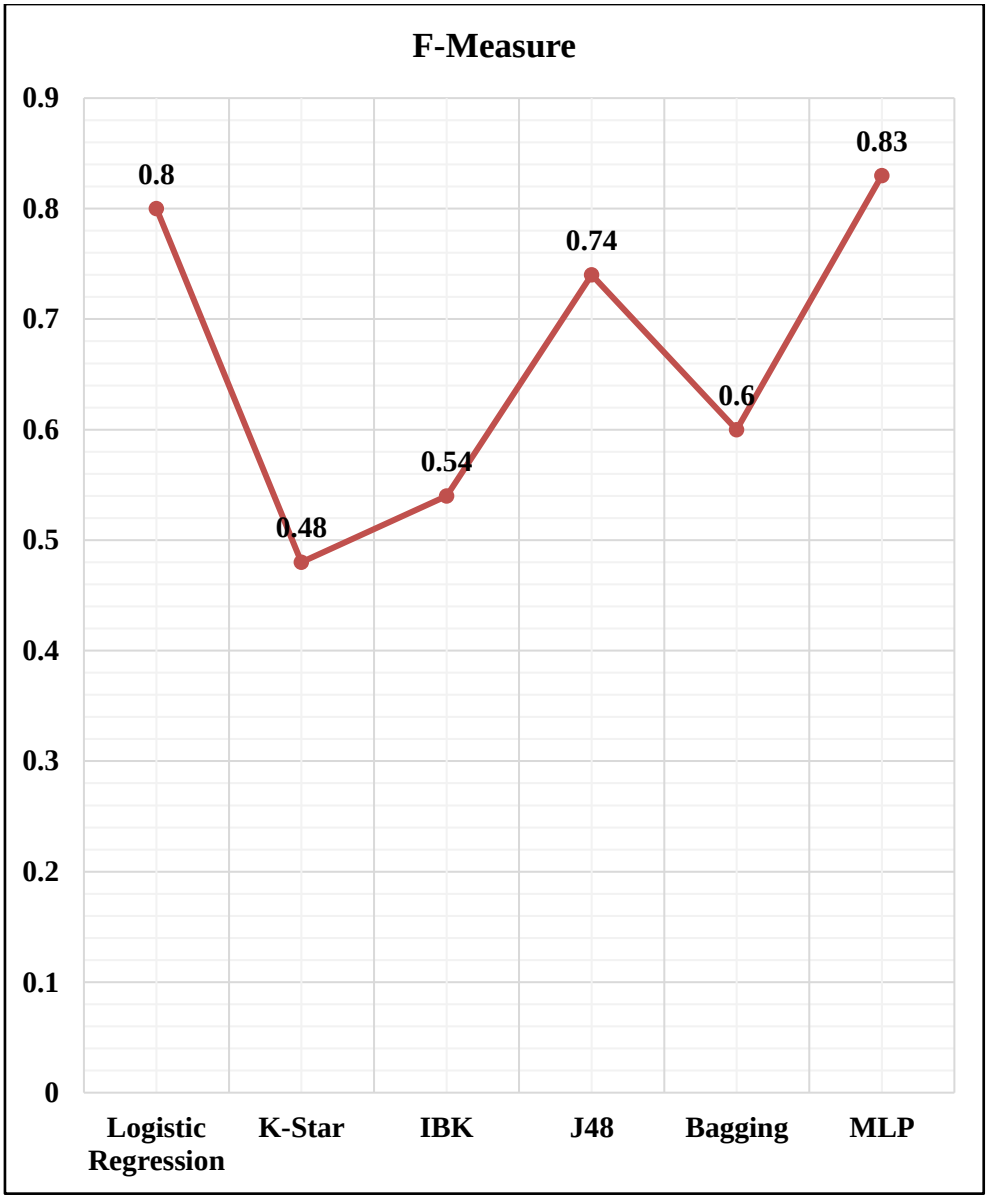

Figure 5.38: Overall F-Measure for Classifiers Used in Task Offloading and Resource Management

According to figure 5.38, it can be concluded that the overall F-Measure score of MLP classifier with value 0.83 is found to be the highest followed by Logistic Regression with value 0.8. The other classification algorithms had to have an overall F-Measure score of about 0.74 in case of J48 classifier, 0.6, 0.54, and 0.48 in case of Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure overall F-Measure score were found to be MLP and LR.

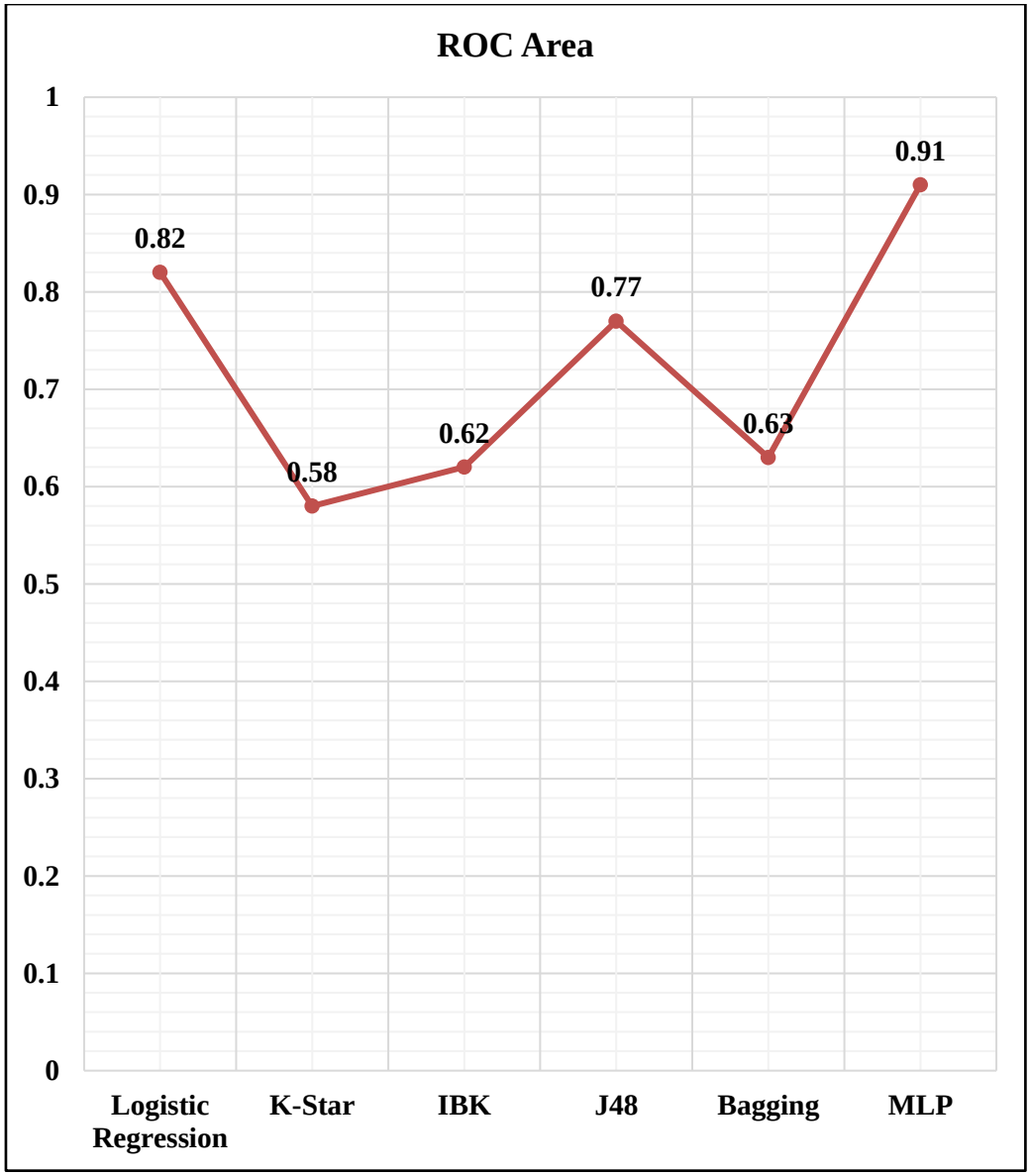

Figure 5.39: Overall, ROC Area for Classifiers Used in Task Offloading and Resource Management

Figure 5.39, It can be concluded that the overall ROC Area of MLP classifier with value 0.91 is found to be the highest followed by Logistic Regression with value of 0.82. The other classification algorithms had to have an overall ROC Area of about 0.77 in the J48 classifier, 0.63, 0.62, and 0.58 in Bagging, IBK, and K-Star. The most appropriate classifiers based on performance measure overall ROC Area were found to be MLP and LR.

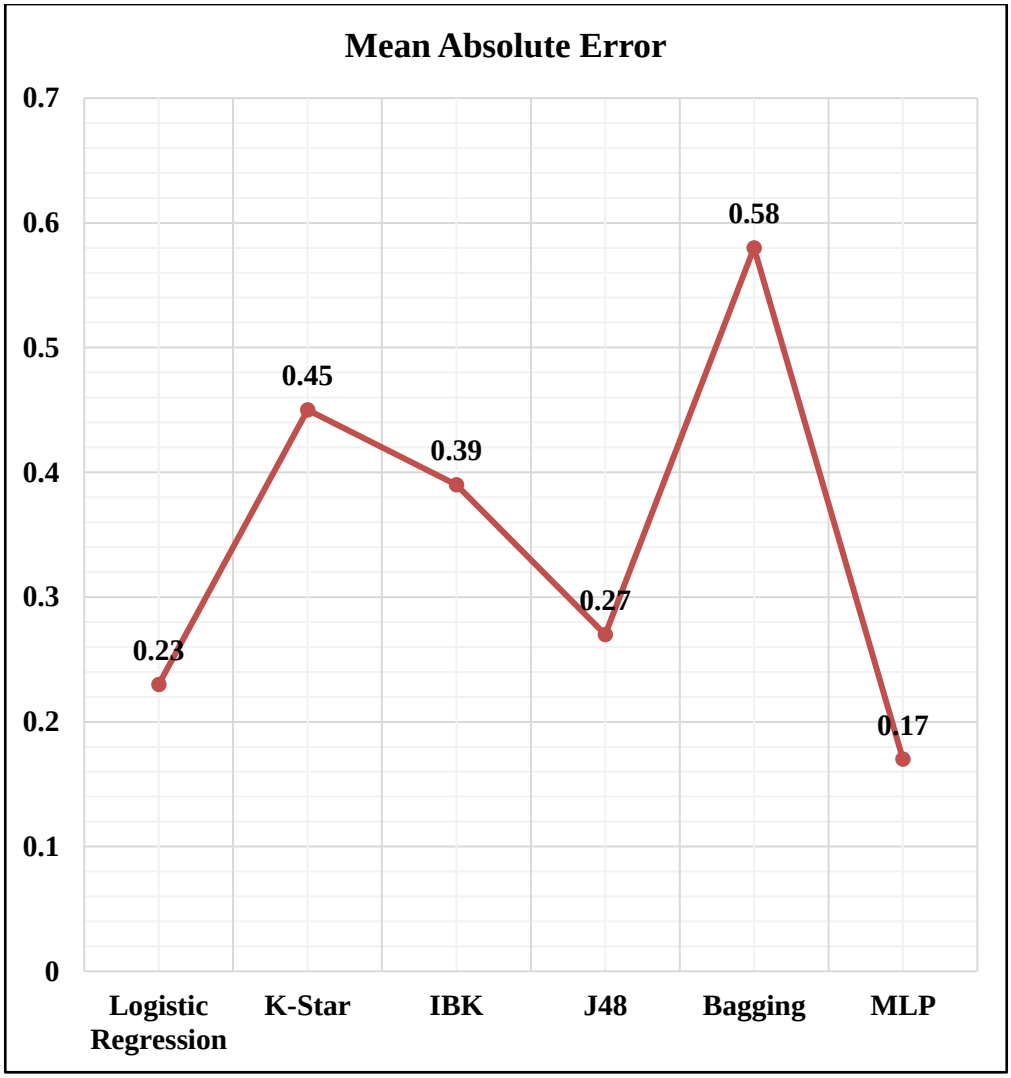

Figure 5.40: Overall, MAE for Classifiers Used in Task Offloading and Resource Management

Figure 5.40, It can be concluded that overall, the mean absolute error value of MLP classifier with 0.17 is found to be the lowest followed by Logistic Regression with value 0.23. The other classification algorithms had to have mean absolute error values of about 0.27 in case of J48 classifier, 0.58, 0.39, and 0.45 in case of Bagging, IBK, and K-Star were found to be quite high. The most appropriate classifiers based on performance measure mean absolute error value were found to be MLP and LR having lesser mean absolute error values as compared to others.

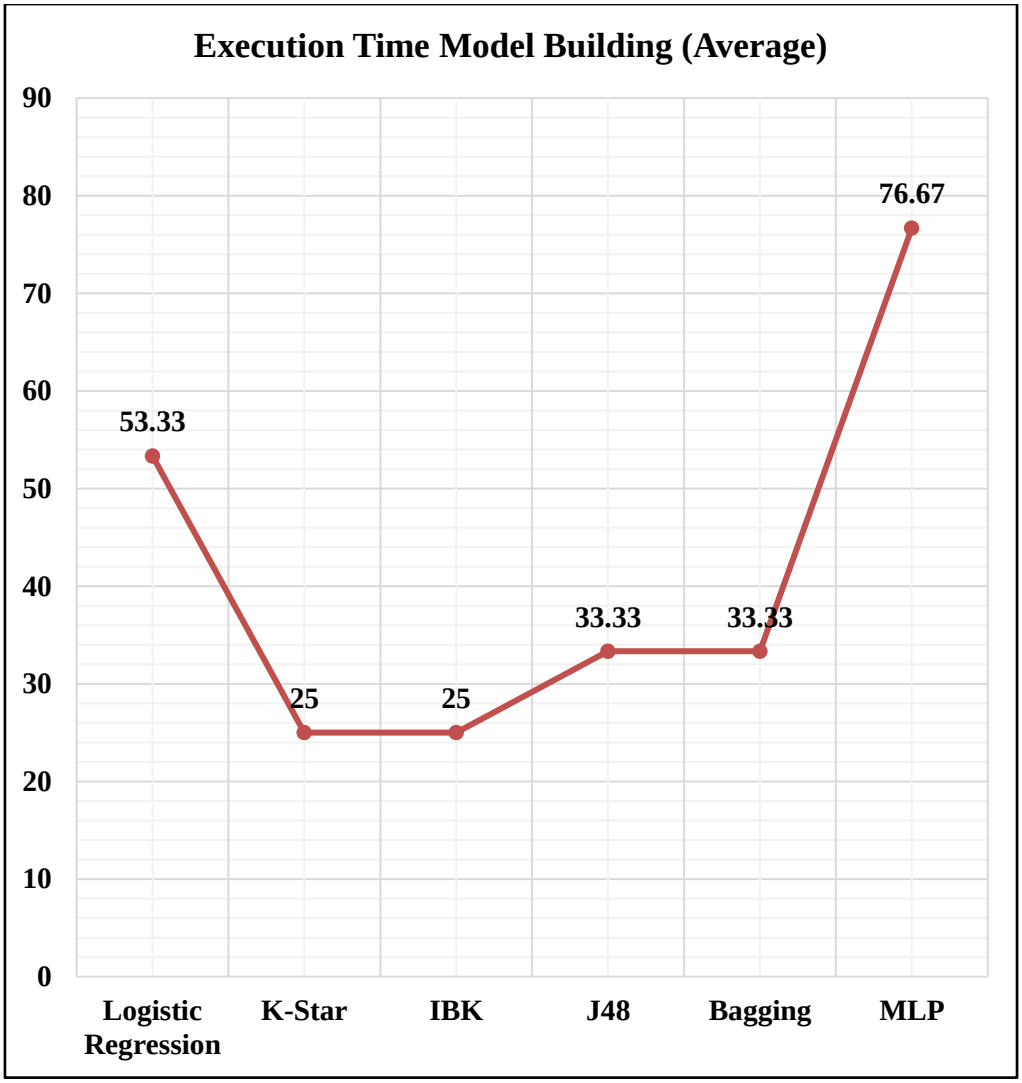

Figure 5.41: Overall Average Execution Time for Classifiers Used in Task Offloading and Resource Management

Figure 5.41, it can be concluded that at configuration setting split 33% the average execution time of model building of K-Star and IBK classifier is found to be 25 milliseconds which is quite less as compared with other classifiers. The other classification algorithms had to have an average execution time of model building of about 33.33 ms in case of J48 classifier, 33.33, 53.33, and 76.67 ms in case of Bagging, LR, and MLP. The most appropriate classifiers based on performance measure average execution time of model building were found to be K-Star and IBK.

In this study, a classification-based intelligent job offloading model is developed in the fog-cloud collaboration network. Initially, an optimization issue involving offloading is solved by considering the threshold values of the relevant cloud data center-related factors. Several application kinds, such as delay-sensitive and computation-intensive ones, must precisely complete their intended duties by the computing resources they demand, which must be provided accordingly. Second, the suggested model uses an intelligent task offloading management system that anticipates the incoming tasks produced by various IoT and mobile devices that are scattered over several remote sites. Simulation findings show that the suggested model can correctly forecast the task delegated to either a fog network or a cloud network with the greatest overall accuracy of 83% and 80% in case of MLP and LR construct. Finally, comparing all the classification algorithms based on various accuracy parameters it can be concluded that MLP and LR are the most appropriate classification algorithms for resource allocation and task offloading although the execution time is higher in both the cases.

Chapter - 6
Conclusion and Future Work

6.1 Findings and Conclusions

6.2 Summarization of Hypotheses Testing Results

6.3 Use of Machine Learning Techniques for Task Scheduling

6.4 Classification Algorithms in Task Offloading and Resource Allocation

6.5 Future Scope

6.6 Limitations

This chapter describes the research activity and its outcomes versus the predicted results as thought throughout the design phase. A complete analysis is being carried out to estimate future possibilities and enhancement to the system gained as a consequence of the suggested study. The study also discusses the important challenges/issues that could be investigated further to move it ahead.

6.1 Findings and Conclusions

By comparing the results of using the FCFS task scheduling algorithm in a Fog and cloud context, it appears that FCFS in the Fog environment better optimizes latency, total network utilization, and energy consumption. In contrast to cloud environments, latency, quality of service, and cost are all improved by using the fuzzy series parallel preprocessing resource scheduling algorithm in a Fog setting.

Latency and power consumption can be minimized by using the Shortest Job First Heuristic approach to schedule work. Much like the preemptive task priority network, the resource allocation technique greatly improves both QoS and efficiency.

Rule-based fuzzy network often known as fuzzy logic, is a resource scheduling technique that optimizes both latency and energy usage. In a similar vein, the QoS may be significantly optimized with the Fault, Configuration, Accounting, Performance, and Security methods.

6.2 Summarization of Hypotheses Testing Results

The comparison between fog-based and cloud-based systems based on execution time ($H_0$1) demonstrates that using smart fog-based systems results in a significant decrease in execution time when compared to cloud-based systems. With values of 9872, 3008, 7866, 5417, 4533, 4024, and 8703, respectively, there is a significant reduction in execution time in the Fog systems 8:10, 9:9, 7:10, 6:10, 6:6, 4:10, and 2:6. It is therefore abundantly evident that the Fog layer is crucial to cutting down on execution time.

The comparison of Fog-based and Cloud-based systems based on latency ($H_0$2) reveals that there is a significant reduction in latency with the usage of Smart Fog-based systems as opposed to Cloud-based systems. There is a significant reduction in latency value in the Fog system 10:5, 4:4, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4, 1:3, 1:2, and 1:1,

such as 453.52, 198.92, 190.69, 198.13, 199.71, 201.36, 191.91, 197.73, 199.41, 201.16, and 194.08. As a result, the fog layer plays a crucial role in latency reduction.

The evaluation between Fog-based and Cloud-based systems based on energy consumption reveals a significant reduction in energy consumption when using Smart Fog-based systems against Cloud-based systems ($H_0$3). There is a significant reduction in energy consumption in the fog systems 10:5, 5:5, 4:5, 4:4, 3:5, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4,1:3, 1:2 and 1:1. Hence, based on the performance measure energy used, it is apparent that there is a considerable difference between the SMART FOG protocol-based system and the cloud-based system.

The analysis of Fog fog-based systems and Cloud cloud-based systems based on cost of execution reveals that there is a significant cost of execution decrease with the usage of Smart Fog-based systems as compared to Cloud-based systems ($H_0$4). There is a significant cost reduction in the Fog system 10:5, 6:10, 5:5, 4:5, 4:4, 3:10, 3:5, 2:9, 2:8, 2:6, 2:5, 2:4, 2:3, 2:2, 1:5, 1:4, 1:3, 1:2, and 1:1. Hence, based on the performance measure cost of execution, it is evident that there is a considerable difference between the SMART FOG protocol-based system and the cloud-based system.

The comparison between Fog-based system and Cloud cloud-based system based on total network usage reveals that there is a significant decrease in total network usage when using a Smart Fog-based system against Cloud-based systems ($H_0$5).In the fog system, there is a significant reduction in overall network utilization such as 813124, 100000, 100000, 889585, 100000, 717690, 600582.6, 100000, 560311.2, 200000, 100000, 376389.8, 300487.8, 226130, 151466.2, 187988.4, 150136.4, 112806.6, 75270.6, and 38142.7. Hence, based on the performance measure of total network use, it is apparent that there is a considerable difference between the SMART FOG protocol-based system and the cloud-based system.

The analysis of Fog-based and Cloud-based systems based on computational power consumed reveals a significant reduction in computational power consumed when using Smart Fog-based systems against Cloud-based systems ($H_0$6). There is a significant reduction in computational power consumed by Fog systems in all cases when compared to cloud-based systems, implying that there is a significant difference between SMART FOG protocol-based systems and cloud-based systems based on the

performance measure computational power consumed by Fog devices in comparison to Cloud devices. For statistical validation of our findings, various null hypotheses were tested and the outcomes of these tests are as follows:

Table 6.1: Chi-Square (χ^2) Test for Awareness Level

Sr. No.	Hypothesis	Result @ 5 % Level
$H_0$1	There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure execution time.	Rejected
H_a2	There is a significant difference between SMART FOG protocol-based System and cloud-based systems based on the performance measure latency.	Accepted
$H_0$3	There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure energy consumed.	Rejected
H_a4	There is significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure cost of execution.	Accepted
$H_0$5	There is no significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure of total network usage.	Rejected
H_a6	There is a significant difference between SMART FOG protocol-based system and cloud-based system based on the performance measure computational power consumed.	Accepted

Table 6.1, can be concluded that the hypothesis “SMART FOG protocol-based technique to create Fog Computing environment will share computational power to IoT devices with low computational power and other aspects” is being accepted which suggests that SMART FOG protocol-based technique reduces computational power consumption for the Fog devices and share computational power with IoT devices by lower the total consumption.

Finally, the hypothesis "H_a1: SMART FOG protocol-based technique to create Fog Computing environment will share computational power to IoT devices with low computational power and other aspects" is accepted, implying that the SMART FOG protocol-based technique reduces computational power consumption for Fog devices and shares computational power with IoT devices by lowering total consumption.

6.3 Use of Machine Learning Techniques for Task Scheduling

It was discovered that when the K-Star classifier was employed for task scheduling, it properly identified around 91% of the cases, which was much higher than the other classification approaches tested, such as IBK, Logistic Regression, and AdaBoostM1. Similarly, the accuracy, recall, and F-measure values of 0.92, 0.91, and 0.90 were greater in comparison to IBK, Logistic Regression, and AdaBoostM1; also, the mean absolute error value was 0.05, and the FP rate value was 0.04.

In logistic regression, the correctly categorized examples were about 88%, which was much higher than the other classification approaches investigated, such as IBK and AdaBoostM1. Similarly, the accuracy, recall, and F-measure values of 0.88, 0.88, and 0.87 were greater in comparison to IBK and AdaBoostM1, as was the mean absolute error value of 0.05 and the FP rate value of 0.04.

Overall K-star is the best classification algorithm that can be used for task scheduling followed by Logistic Regression as in the majority of observations at different configuration settings the Accuracy, Precision, Recall, F-Measure, etc. are higher in case of algorithms mentioned above.

6.4 Classification Algorithms in Task Offloading and Resource Allocation

The results confirm that the MLP classifier has the best overall accuracy value 0.83, followed by the Logistic Regression value 0.80. The other classification methods had an overall accuracy of roughly 0.75 in the case of the J48 classifier, 0.60, 0.61, and 0.48 in the case of Bagging, IBK, and K-Star, respectively. MLP and Logistic Regression were discovered to be the best acceptable classifiers based on performance measure total accuracy. Comparing classifiers based on overall Kappa statistics used for task offloading and resource allocation in SMART FOG environment, MLP and Logistic Regression have higher overall Kappa statistics values of 0.67 and 0.6, respectively, indicating that they are superior classifiers.

MLP classifier has the best precision at 0.85, followed by Logistic Regression at 0.83. J48 had 0.79 Precision, Bagging 0.48, IBK 0.76, and K-Star 0.49. MLP and Logistic Regression were the most precise classifiers.

MLP classifier has the greatest recall overall with 0.84, followed by Logistic Regression with 0.80. Bagging, IBK, and K-Star had Recalls of 0.61, 0.62, and 0.49, respectively, while J48 had 0.75. MLP and Logistic Regression were the best classifiers for total Recall.

MLP classifier has the lowest mean absolute error value of 0.17, followed by Logistic Regression with 0.23. J48, Bagging, IBK, and K-Star had mean absolute error values of 0.27, 0.58, 0.39, and 0.45, respectively. MLP and Logistic Regression were the best classifiers based on performance metric mean absolute error value.

In conclusion, after examining each classification algorithm based on a variety of accuracy parameters, one can conclude that MLP and Logistic Regression are the classification algorithms that are best suited for resource allocation and task offloading.

6.5 Future Scope

In this section, the key issues, future difficulties, and future research prospects for task scheduling in fog computing are discussed.

Resource Utilization of Fog Node

The fog devices have limited storage, processing, and energy capabilities due to their lack of resources. They receive dynamic workloads from applications that are sensitive to latency as well as apps that are tolerant of delay. As a result, the difficult aspect is to schedule the unpredictability of the arrival of activities on these fog nodes to make the best possible use of the available resources.

Optimal Resource Allocation

IoT devices produce a large number of tasks, which have to be appropriately distributed between fog nodes to achieve a quicker reaction time. This is especially important for applications that are sensitive to latency. Since fog computing makes it possible for fog nodes and Internet of Things devices to move about freely, the resources that are reachable at any given time may be inaccessible at other times. Because of this, the process of allocating resources is a difficult endeavor. The problems that need to be addressed are long latency for real-time applications, a lack of generalization, and rapid adaptation of the algorithms that are currently available.

Parallel Scheduling

In the method known as parallel processing, one operation is broken down into several smaller tasks, all of which are then carried out at the same time. Another unresolved problem that requires attention is the division of activities into subtasks that can decrease delays through the use of distributed computing.

Privacy

Several different fog applications, such as smart healthcare, send a significant amount of personally identifiable information to fog nodes. As a result, protecting the confidentiality of such data is of the utmost importance to users. Even while some researchers use methods that protect users' privacy on fog nodes, there is yet no authentication solution that can be considered satisfactory. Because the fog nodes are more susceptible to possible dangers, authenticating users can be a difficult and time-consuming process.

Security

Fog nodes are vulnerable to attacks. As a result, developing a safety algorithm that is not only lightweight but also has a fast speed and is trustworthy is still a tough issue. At the moment, only a small number of academics are focusing their attention on the security concerns associated with fog computing; nonetheless, there are still several outstanding challenges, such as dynamic authentication, access controls, external threats, and intrusion detection.

Context-aware Service Provisioning

The context is made up of the many runtime elements that have the potential to influence the applications. The currently available approaches to context-aware service provisioning are less flexible and scalable, and they are unable to manage a significant number of Internet of Things applications. Because of this, more approaches to context-aware service delivery should be researched so that the aforementioned restrictions may be solved.

Energy Consumption

Energy-aware computing in fog is still an open question that has to be answered since fog devices are limited in their ability to use energy due to their usage of low-power batteries. Several academics are concentrating their efforts on energy optimization,

but several problems still need to be addressed, including improper utilization of bandwidth during data transfer, energy waste, and battery-draining concerns.

6.6 Limitations

Fog computing faces several limitations, including high latency compared to edge computing, increased complexity in network management, potential security vulnerabilities, and limited scalability. It can also suffer from resource constraints due to dependency on intermediate devices and challenges in data processing efficiency. Additionally, ensuring consistent connectivity and handling diverse data types can pose significant difficulties in fog computing environments.

1. Scope and Generalizability: The study may have focused on specific IoT architectures, protocols, and technologies, which might limit its generalizability to other IoT scenarios or environments.

2. Real-world Implementation Challenges: The study might not have addressed the practical challenges associated with implementing the SMART FOG protocol-based technique, such as hardware compatibility, software integration, security considerations, and deployment complexities.

3. Benchmarking and Comparison: The study might lack comprehensive benchmarking or comparison with existing IoT architectures, protocols, or alternative solutions. Comparative analysis would provide a better understanding of the advantages and limitations of the proposed SMART FOG approach.

4. Limited Testing Scenarios: The evaluation of the SMART FOG technique might have been conducted under specific testing scenarios or simulated environments, which may not fully capture the complexities and dynamics of real-world IoT deployments.

5. Time Constraints: The study might have faced time limitations, which could impact the depth of analysis, experimentation, and validation of the proposed techniques.

6. Lack of Real-world Deployment Validation: The proposed SMART FOG technique might not have been validated in real-world IoT deployments or scenarios, which may limit the assessment of its practical applicability and performance.

In conclusion, fog computing's limitations include potential latency issues, increased network complexity, and security vulnerabilities. It also faces scalability challenges, resource constraints from intermediary devices, and inefficiencies in data processing.

REFERENCES

A. Journal and Books

Aalsadie D. (2022), “Task Scheduling in Fog Computing – Classification, Review, Challenges and Future Directions” IJCSNS International Journal of Computer Science and Network Security, VOL.22 No.4, pp. 89-96.

Aazam M., & Huh E.N. (2015), “Dynamic resource provisioning through Fog micro datacenter. Pervasive Computing and Communication Workshops (PerCom Workshops)”, IEEE, 2015, pp. 105-110.

Aazam M., & Huh E.N. (2015), “Fog computing micro datacentre based dynamic resource estimation and pricing model for IoT. In Advanced Information Networking and Applications (AINA)”, IEEE 29th International Conference, 2015, pp. 687-694.

Abdul-Qawy A., Magesh E., &Tadisetty S. (2015), “The IoT: An Overview”, A S Abdul-Qawy et al. Int. Journal of Engineering Research and Applications ISSN: 2248-9622, Vol. 5, Issue 12, (Part - 2) December 2015, pp.71-82.

Abohamama A.S., El-Ghamry A., & Hamouda E. (2016), “Real-Time Task Scheduling Algorithm for IoT-Based Applications in the Cloud–Fog Environment”, J NetwSyst Manage, pp. 30-54.

Abomhara M., Koien G.M. (2014), “Security and privacy in the IoT: Current status and open issues. In Privacy and Security in Mobile Systems (PRISMS)”, International Conference on. IEEE, pp. 1–8.

Adel A. (2020), “Utilizing technologies of fog computing in educational IoT systems: privacy, security, and agility perspective”, J Big Data, 7, 99. doi:10.1186/s40537-020-00372., 2020, pp. 37-72.

Ahmed, A., Arkian, H., Battulga, D., Fahs, A., Farhadi, M., Giouroukis, D., & Wu, L. (2019), “Fog Computing Applications: Taxonomy and Requirements”, pp.1-4.

Aimal Khan, Assad Abbas, Hasan Ali Khattak, Faisal Rehman, Ikram Ud Din, Sikandar Ali. (2022), "Effective Task Scheduling in Critical Fog Applications", Scientific Programming, vol. 2022, Article ID 9208066, pp. 1-15.

Alavi, A., Jiao, P., Buttlar, W., & Lajnef, N. (2018), “IoT-enabled smart cities: State-of-the-art and future trends”. Measurement, pp.129.

Alizadeh M., Khajehvand V., Rahmani A., & Akbari E. (2020), “Task scheduling approaches in fog computing: A systematic review”, International Journal of Communication Systems 33, pp. 45-83.

Aljumah, A., &Ahanger, T. A. (2018), “Fog computing and security issues: A review. In Proceedings of the 2018 7th International Conference on Computers Communications and Control (ICCCC)”, Oradea, Romania, 8–12 May 2018, pp. 237–239.

Alrawais, A., Alhothaily, A., Hu, C., & Cheng, X. (2017), “Fog computing for the IoT: Security and privacy issues”. IEEE Internet Comput., 21(6), pp. 34–42.

Alsmadi A.M., Aloglah R.M.A., Abu-darwish N.J.S., Al Smadi A., Alshabanah M., Alkhaldi H., Alsmadi M.K. (2021), “International Journal of Electrical and Computer Engineering (IJECE)”, Vol. 11, No. 3, June 2021, pp. 2219~2228. ISSN: 2088-8708, DOI: 10.11591/ijece.v11i3, pp. 2219-2228.

Alturki, B., Reif-Marganiec, S., Perera, C., & De, S. (2019), “Exploring the Effectiveness of Service Decomposition in Fog Computing Architecture for the IoT”.1904.00381, pp.10-12.

Ansari D.B. (2018), Atteeq-Ur-Rehman, and R. A. Mughal, “Internet of Things (IoT) protocols: A brief exploration of MQTT and CoAP”, International Journal of Computer Applications, vol. 179, no. 27, pp. 9–14.

Atlam, H., Walters, R., & Wills, G. (2018), “Fog Computing and the IoT: A Review”. Big Data and Cognitive Computing, 2(2), pp.1-10.

Attar, A. H., & Sutagundar, A. (2018), “A survey on resource management for fog-enhanced services and applications”. Int. J. Sci. Res., 17(2), p.138.

Badidi, E., &Ragmani, A. (2020), “An Architecture for QoS-Aware Fog Service Provisioning”. Procedia Comput. Sci., 170, pp.411–418.

Bandyopadhyay, D., & Sen, J. (2011), “IoT: Applications and challenges in technology and standardization”. Wireless Personal Communications, 58, pp.49–69.

Baniata, H., & Kertesz, A. (2020), “A survey on blockchain-fog integration approaches”. IEEE Access, 8, 102657–102668, pp.25-27.

Baouya, A., Chehida, S., Bensalem, S., &Bozga, M. (2020), “Fog Computing and Blockchain for Massive IoT Deployment”. In 2020 9th Mediterranean Conf. on Embedded Computing (MECO), pp.1-2.

Bellavista, P., Berrocal, J., Corradi, A., Das, S., Foschini, L., & Zanni, A. (2019), “A survey on fog computing for the IoT”. Pervasive Mob. Comput., 52, pp. 71–99.

Berlin, (2018), “A Research Perspective on Fog Computing”, Springer International Publishing AG, part of Springer Nature 2018L. Braubach et al. (Eds.): ICSOC Workshops 2017, LNCS 10797, pp. 198–210.

Berman, F., Cabrera, E., Jebari, A., & Marrakchi, W. (2022), “The impact universe – a framework for prioritizing the public interest in the IoT”. Patterns, 3(1), 100398. pp. 1–8.

Bitam S., Zeadally S., Mellouk A. (2018), “Fog computing job scheduling optimization based on bees swarm”, Enterpr. Inform. Syst., https://www.tandfonline.com/doi/full/10.1080/17517575.2017.1304579, Vol. 12, pp. 373-397.

Bonomi, F., Milito, R., Zhu, J., Addepalli, S. “Fog computing and its role in the internet of things”, In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM (2012), pp. 13–16.

Borodin, V. A. (2014), “IoT– the next stage of the digital revolution”. Educational Resources and Technologies, 2(5), pp.4-5.

Bosman, R., Lukkien, J., & Verhoeven, R. (2011), “Gateway architectures for service-oriented application-level gateways”. IEEE Trans. on Consumer Electronics, 57(2), pp. 453–461.

Bourque, P., & Fairley, R. (Eds.). (2014), SWEBOK: “Guide to the Software Engineering Body of Knowledge (3.0 ed.)”. IEEE Computer Society, Los Alamitos, pp. 1-2.

Bubnova, M. Yu., & Kryukova, A. A. (2014), “Social client-oriented technologies in the activities of modern companies”. Economics and Society, 3(4), pp. 65–67.

Butun, I., Sari, A., & ÃŰsterberg, P. (2019), “Security Implications of Fog Computing on the IoT”. In 2019 IEEE International Conference on Consumer Electronics (ICCE), pp. 1-10.

Chen S., Xu H., Liu D., Hu B., Wang H. (2014), “A vision of IOT: Applications, challenges, and opportunities with China perspective”. IEEE IoTjournal, vol. 1, no. 4, pp. 349–359.

Chiang, M., & Zhang, T. (2016), “Fog and IoT: An overview of research opportunities”. IEEE Internet Things J., 3, pp. 854–864.

CIW Team. (2023), “China’s IoT spending to reach US$298 billion by 2026”. Retrieved from https://www.chinainternetwatch.com/31628/iot-market-trends, pp. 2–8.

DeMedeiros, K., Hendawi, A., & Alvarez, M. (2023), “A survey of AI-based anomaly detection in IoT and sensor networks. Sensors”, 23, pp. 1352.

Din, I. U., Guizani, M., Kim, B. S., Hassan, S., & Khan, M. K. (2018), “Trust management techniques for the IoT: A survey”. IEEE Access, 7, pp. 29763–29787.

Dubravac, S., & Ratti, C. (2015), IoT: Evolution or revolution? Part 1 of the IoT report series. pp. 8–9.

Edemacu, K., &Bulega, T. (2014), “Resource sharing between M2M and H2H traffic under time-controlled scheduling scheme in LTE networks”. In: 2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA), pp. 1–6.

El Alami, Hassan & Sidna, Jeddou & Baina, Amine & Najid, Abdellah. (2020), "Analysis and evaluation of communication Protocols for IoT Applications". IEEE Transactions on Industrial Informatics, 7(4), pp. 630–640.

Ema, R. R., Islam, T., & Ahmed, M. H. (2019), "Suitability of Using Fog Computing Alongside Cloud Computing". In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019, pp. 1–4.

Gandotra, P., & Lall, B. (2020), "Evolving Air Pollution Monitoring Systems for Green 5G: From Cloud to Edge. In Proceedings of the 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 4–5 June 2020, pp. 1231–1235.

Ghobaei-Arani M., Souri A., Safara F., Norouzi M. (2020), "An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing", Trans. Emerg. Telecommunication. Technol., Vol. 31, pp. 37-70.

Giordano, A., Spezzano, G., Vinci, A. (2016), "Smart Agents and Fog Computing for Smart City Applications." In: Alba, E., Chicano, F., Luque, G. (eds) Smart Cities. Smart-CT 2016. Lecture Notes in Computer Science (), vol 9704. Springer, Cham, pp. 1-14.

González-Martínez, J. A., Bote-Lorenzo, M. L., Gómez-Sánchez, E., & Cano-Parra, R. (2015), "Cloud computing and education: A state-of-the-art survey". Comput. Educ., 80, pp. 132–151.

Gu, Lin, Z. Deze, G. Song, B. Ahmed, and X. Yong. (2015), "Cost-efficient resource management in fog computing supported medical cps", IEEE Transactions on Emerging Topics in Computing, 2015, pp. 1-12.

Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., & Savio, D. (2010), "Interacting with the SOA-based IoT: Discovery, query, selection, and on-demand provisioning of web services". IEEE Transactions on Services Computing, 3(3), pp. 223–235.

Gupta H., Dastjerdi A. V., Ghosh S. K., & Buyya R. (2016), "Ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge, and fog computing environments", CoRR, abs/1606.02007, pp.23-45.

Guzuyeva, E. R. (2018), "Application of information technology in large and small businesses. In Proceedings of the IV International Correspondence Scientific and Practical Conference." AIP Publishing, pp. 226–230.

Hakan (2023), "Bibliometric analysis and scientific mapping of IoT", https://www.researchgate.net/publication/367286890_Bibliometric_Analysis_and_Scientific_Mapping_of_IoT,Journal of Computer Information Systems, pp. 1–8.

Hamdoun, S., Rachedi, A., & Ghamri-Doudane, Y. (2015), “Radio resource sharing for MTC in LTE-A: An interference-aware bipartite graph approach”, In: 2015 IEEE Global Communications Conference (GLOBECOM) IEEE., pp. 1–7.

Hassan Z., Ali H., Badawy M. (2015), “IoT: Definitions, Challenges, and Recent Research Directions”, International Journal of Computer Applications, Vol. 128, pp. 975-987.

Heck, M., Edinger, J., Schaefer, D., & Becker, C. (2018), “IoT Applications in Fog and Edge Computing: Where Are We and Where Are We Going?”. In Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 30 July–2 August 2018, pp. 1–6.

Heer T., Garcia-Morchon O., Hummen R., Keoh SL., Kumar S.S., Wehrle K. (2011), “Security challenges in the IP based IoT”, Wirel Pers Commun, 61(3), pp. 527–542.

Henze, M., Matzutt, R., Hiller, J., Erik, M., Ziegeldorf, J. H., van der Giet, J., & Wehrle, K. (2020), “Complying with Data Handling Requirements in Cloud Storage Systems”, IEEE Trans. Cloud Computing., pp. 1-10.

Hoang, D., & Dang, T. D. (2017), “FBRC: Optimization of task scheduling in fog-based region and cloud”. In: 2017 IEEE Trustcom/ BigDataSE /ICESS, pp. 1109–1114. IEEE.

Huang, Q., Yang, Y., & Wang, L. (2017), “Secure data access control with ciphertext update and computation outsourcing in fog computing for the IoT”. IEEE Access, 5, pp. 12941–12950.

Huttunen, J., Jauhiainen, J., Lehti, L., Nylund, A., Martikainen, M., & Lehner, O. (2019), “Big data, cloud computing and data science applications in finance and accounting”. ACRN Oxf. J. Financ. Risk Perspect., 8, pp. 16–30.

Jamil B., Ijaz H., Shojafar M., Munir K., &Buyya R. (2022), “Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions. ACM Computing Surveys”, pp. 1-35.

Jia, B., Hu, H., Zeng, Y., Xu, T., & Yang, Y. (2018), “Double-matching resource allocation strategy in fog computing networks based on cost efficiency”. J. Commun. Netw., 20(3), pp. 237–246.

Katal A, Sethi V, Lamba S, and Choudhury T (2016), “Fog computing: Issues, challenges, and tools Advances in Intelligent Systems and Computing”, pp. 971–982.

Kaur, J., Agrawal, A., & Khan, R. A. (2020), “Security Issues in Fog Environment: A Systematic Literature Review”. Int. J. Wirel. Inf. Netw., 27, pp. 467–483.

Khan, S., Parkinson, S., & Qin, Y. (2017), “Fog computing security: A review of current applications and security solutions”. J. Cloud Computing., 6, pp. 1–22.

Kimovski, D., Ijaz, H., Saurabh, N., & Prodan, R. (2018), “Adaptive nature-inspired fog architecture”. In: 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), pp. 1–8. IEEE.

Kopras, B., Idzikowski, F., Bossy, B., Kryszkiewicz, P., & Bogucka, H. (2023). “Communication and Computing Task Allocation for Energy-Efficient”. Fog Networks. Sensors, 23, pp. 997.

Kumari A. Dr., Tanwar S., Tyagi S., Kumar N., Rodrigues J. (2019), “Fog Computing for Smart Grid Systems in 5G Environment: Challenges and Solutions. IEEE Wireless Communications”, pp. 1–8.

Lai K.L., Chen J. (2021), “Development of Smart Cities with Fog Computing and IoT”, Journal of Ubiquitous Computing and Communication Technologies, 3, pp. 52-60.

Lata M., Kumar V. (2022), “Fog Computing Infrastructure for Smart City Applications”, Recent Advancements in ICT Infrastructure and Applications, pp.119–133.

Li, H., Shou, G., Hu, Y., & Guo, Z. (2016), “Mobile edge computing: Progress and challenges”. In Proceedings of the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (Mobile Cloud), Oxford, UK, 29 March–1 April 2016, pp. 83–84.

Li, Q., Zhao, J., Gong, Y., & Zhang, Q. (2019), “Energy-efficient computation offloading and resource allocation in fog computing for the internet of everything”. China Common., 16(3), pp. 32–41.

Liu, L., Qi, D., Zhou, N., & Wu, Y. (2018), “A task scheduling algorithm based on classification mining in fog computing environment”. Wirel. Commun. Mobile Compute., 2018, pp. 1-100.

Liu, Z., Yang, X., Yang, Y., Wang, K., & Mao, G. (2018), DATS: “Dispersive stable task scheduling in heterogeneous fog networks”. IEEE Internet Things J., 6(2), pp. 3423–3436.

Macarulla, Marcel & Albano, Michele & Ferreira, Luis & Teixeira, César (2016), “Lessons Learned in Building a Middleware for Smart Grids. Journal of Green Engineering”. 6. 1-26. 10.13052/jge1904-4720.611. IEEE Access, 6, 23626–23638, pp. 4-19.

Mao, Y., You, C., Zhang, J., Huang, K., &Letaief, K. B. (2017), “A survey on mobile edge computing: The communication perspective”. IEEE Common. Surv. Tutor., 19, pp. 2322–2358.

Marbukh, V. (2019), “Towards Fog Network Utility Maximization (FoNUM) for Managing Fog Computing Resources”. In Proceedings of the 2019 IEEE International Conference on Fog Computing (ICFC), Prague, Czech Republic, 24–26 June 2019, pp. 195–200.

Matrouk K., Alatoun K. (2021), “Scheduling Algorithms in Fog Computing: A Survey”. International Journal of Networked and Distributed Computing”, Volume 9, Issue 1, January 2021, pp. 59 – 74.

Mebrek, A., Merghem-Boulahia, L., &Esseghir, M. (2017), “Efficient green solution for a balanced energy consumption and delay in the IoT-Fog-Cloud computing”. In Proceedings of the 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 30 October–1 November 2017, pp. 1–4.

Mohan, P., & Thangavel, R. (2013), “Resource selection in grid environment based on trust evaluation using feedback and performance”. Am. J. Appl. Sci., 10(8), pp. 924.

Naha, R. K., Garg, S., Georgakopoulos, D., Jayaraman, P. P., Gao, L., Xiang, Y., & Ranjan, R. (2018), “Fog computing: Survey of trends, architectures, requirements, and research directions”. IEEE Access, 6, pp. 47980–48009.

Ni, L., Zhang, J., Jiang, C., Yan, C., & Yu, K. (2017), “Resource allocation strategy in fog computing based on priced timed petri nets”. IEEE Internet Things J., 4(5), pp. 1216–1228.

Parikh, S., Dave, D., Patel, R., & Doshi, N. (2019), “Security and privacy issues in cloud, fog and edge computing”. Procedia Comput. Sci., 160, pp.734–739.

Pham, X. Q., Man, N. D., Tri, N. D. T., Thai, N. Q., & Huh, E. N. (2017), “A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing”. Int. J. Distrib. Sens. Netw., 13(11), 1550147717742073, pp. 10-18.

Prakash P., Darshaun K. G., Yaazhlene P., Medidhi V. G., & Vasudha B. (2017), “Fog Computing: Issues, Challenges, and Future Directions”, International Journal of Electrical and Computer Engineering (IJECE), 7(6), pp.3669-3673.

Prakash, M., & Ravichandran, T. (2012), “An efficient resource selection and binding model for job scheduling in grid”. Eur. J. Sci. Res., 81(4), pp. 450–458.

Priyadarshinee P. (2021), “Impact of Fog Computing on Indian Smart-Cities: An Empirical Study”, 10.21203, pp-12-33.

Qasem M., Abu srhan A., Natouryeh H., Alzaghoul E. (2020), “Fog Computing Framework for Smart City Design”, International Journal of Interactive Mobile Technologies (iJIM), 14, pp.109.

Rahmani A.M., Thanigaivelan N.K., Gia T.N., Granados J., Negash B., Liljeberg P., & Tenhunen H. (2020), “Smart e-health gateway: bringing intelligence to internet-of-things based ubiquitous healthcare systems”, In 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Jan 2015, pp. 826–834.

Ravi L. et al. (2016), “A Collaborative Location Based Travel Recommendation System through Enhanced Rating Prediction for the Group of Users”, Hindawi Publishing Corporation Computational Intelligence and Neuroscience, Vol 2016, pp. 1-11.

Ren, Y., Zhu, F., Qi, J., Wang, J., & Sangaiah, A. K. (2019), “Identity management and access control based on blockchain under edge computing for the industrial IoT”. Appl. Sci., 9, pp. 2058.

Sabireen H., Neelanarayanan V. (2021), “A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges” ICT Express, Volume 7, Issue 2, pp. 162-176.

Saini M.K., Saini R.K. (2019), “IoT Applications and Security Challenges: A Review”, International Journal of Engineering Research & Technology (IJERT), Volume 7, Issue 12, pp. 1-5.

Sarkar, S., Chatterjee, S. (2015), “Assessment of the suitability of fog computing in the context of internet of things”, IEEE Transactions on Cloud Computing pp. 99.

Satyakam R., Rajni A. (2021), “Fog Computing Architecture, Application and Resource Allocation: A Review”, WCNC-2021: Workshop on Computer Networks & Communications, May 01, 2021, Chennai, India, pp. 31-36.

Savya S. (2021), “Scheduling in Fog Computing: A Survey. International Journal of Advanced Research in Science”, Communication and Technology (IJARSCT), Volume 1, Issue 2, pp. 154-157.

Sha, K., Yang, T. A., Wei, W., & Davari, S. (2020), “A survey of edge computing-based designs for IoT security”. Digit. Commun. Netw., 6, pp.195–202.

Shalini C., Mohana Y., and Devi S. (2019), “Fog Computing for Smart Cities”, Proceedings of the 2019 14th International Conference on Computer Engineering and Systems (ICCES), pp. 912-916.

Sheikh, M. S., Noor Enam, R., & Qureshi, R. I. (2023), “Machine learning-driven task scheduling with dynamic K-means based clustering algorithm using fuzzy logic in FOG environment”, Frontiers in Computer Science, 5, 1293209, pp. 1-15.

Skarlat O., Schulte S., Borkowski M., Leitner P. (2016), “Resource Provisioning for IoT Services in the Fog. In Service-Oriented Computing and Applications (SOCA)”, IEEE 9th International Conference, pp. 32-39.

Songhorabadi M., Rahimi M., Farid A. M., and Kashani M. H. (2020), "Fog Computing Approaches in Smart Cities: A State-of-the-Art Review" Computer Science Networking and Internet Architecture (cs.NI), Volume 2, pp. 1-19.

Stojmenovic, I., & Wen, S. (2014), "The fog computing paradigm: Scenarios and security issues". In Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, 7–10 Sept. 2014, pp. 1–8.

Sun, Y., & Zhang, N. (2017), "A resource-sharing model based on a repeated game in fog computing". Saudi J. Biol. Sci., 24(3), pp. 687–694.

Syed, M. H., Fernandez, E. B., & Ilyas, M. (2016), "A pattern for fog computing". In Proceedings of the 10th Travelling Conference on Pattern Languages of Programs, Leerdam, The Netherlands, 7–10 April 2016, pp. 1–10.

Tao, Z., Xia, Q., Hao, Z., Li, C., Ma, L., Yi, S., & Li, Q. (2019), "A survey of virtual machine management in edge computing". Proc. IEEE, 107, pp.1482–1499.

Tzavaras, A., Mainas, N., & Petrakis, E. G. M. (2023), "OpenAPI framework for the Web of Things". IoT, 21, 100675, pp. 1–2.

Uckelmann, D., Harrison, M., &Michahelles, F. (2011), "An architectural approach towards the future IoT". In Architecting the IoT. pp. 10–12.

Wadhwa, H., & Aron, R. (2018), "Fog computing with the integration of IoT: Architecture, applications and future directions. Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)", Melbourne, VIC, Australia, 11–13 December 2018, pp. 987–994.

Wagan, S. A., Koo, J., Siddiqui, I. F., Attique, M., Shin, D. R., & Qureshi, N. M. F. (2022), "Internet of medical things and trending converged technologies: A comprehensive review on real-time applications". Journal of King Saud University – Computer and Information Sciences, pp.1-100.

Wang J., Li D. (2019), "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing", Sensors (Basel), Vol. 19, pp. 1023.

Wang, F., Ge, B., Zhang, L., Chen, Y., Xin, Y., & Li, X. (2013), "A system framework of security management in enterprise systems". Systems and Behavioral Research Science, 30(3), pp.287–299.

Wu, Y., Sheng, Q. Z., & Zeadally, S. (2013), "RFID: Opportunities and challenges. In Next-generation wireless technologies", Springer, pp. 105–129.

Xing, Y., Li, L., Bi, Z., Wilamowska-Korsak, M., & Zhang, L. (2013), "Operations research (OR) in service industries: A comprehensive review". Systems and Behavioral Research Science, 30(3), pp. 300–353.

Xu, L. (2011), "Enterprise systems: State-of-the-art and future trends". IEEE Transactions on Industrial Informatics, 7(4), pp. 630–640.

Yang, Y., Wang, K., Zhang, G., Chen, X., Luo, X., & Zhou, M. T. (2018), "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks". IEEE Internet Things J., 5(5), pp. 4076–4087.

Yang, Y., Zhao, S., Zhang, W., Chen, Y., Luo, X., & Wang, J. (2018), "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks". IEEE Internet Things J., 5(3), pp. 2094 –2106.

Yi, S., Qin, Z., & Li, Q. (2015), "Security and privacy issues of fog computing: A survey". In International Conference on Wireless Algorithms, Systems, and Applications, Springer, pp. 685–695.

Yin, L., Luo, J., & Luo, H. (2018), "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing". IEEE Trans. Industr. Inf., 14(10), pp. 4712–4721.

Ystgaard, K. F., Atzori, L., Palma, D. Heegaard, P. E., Bertheussen, L. E., Jensen, M. R., & De Moor, K. (2023), "Review of the theory, principles, and design requirements of human-centric IoT". Journal of Ambient Intelligence and Humanized Computing. pp. 1-10.

Yuan, J., & Li, X. (2018), "A reliable and lightweight trust computing mechanism for IoT edge devices based on multi-source feedback information fusion". IEEE Access, 6, 23626–23638, pp. 4-19.

Yudidharma, A., Nathaniel, N., Gimli, T. N., Achmad, S., & Kurniawan, A. (2023), "A systematic literature review: Messaging protocols and electronic platforms used in the IoT for the purpose of building smart homes". Procedia Computer Science, 216, pp. 194–203.

Zhang C. (2020), "Design and application of fog computing and IoT service platform for smart city". Future Generation Computer Systems, 112, 10.1016/j.future.2020.06.016, pp.2-10.

Zhang, P., Zhou, M., & Fortino, G. (2018), "Security and trust issues in Fog computing: A survey". Future Gener. Compute. Syst., 88, pp. 18–28.

Zhang, Y., Zhou, B., Jiao, L., & Chen, J. (2015), "An innovative low-cost detection system for IoT privacy leaks". Computer Networks, 90, pp. 80–82.

Zhenqi, S., Haifeng, Y., Xuefen, C., &Hongxia, L. (2013), "Research on uplink scheduling algorithm of massive M2M and H2H services in LTE", pp. 1-10.

sB. Websites

[1] https://link.springer.com/chapter/10.1007/978-3-319-57639-8_1 accessed on 18th May 2019.

[2] https://link.springer.com/chapter/10.1007/978-3-319-91764-_16#citeas accessed on 18th October 2019.

[3] https://www.comsoc.org/publications/magazines/ieee-communications-magazine accessed on 27th January 2020.

[4] https://www.researchgate.net/publication/360903277 acscessed on 7th March 2020.

[5] https://doi.org/10.1007/978-3-319-39595-1_14 pdf accessed on 15th August 2020.

[6] https://vitalflux.com/cohen-kappa-score-python-example-machine-learning/ accessed on 26th December 2020.

[7] https://www.sciencedirect.com/science/article/abs/pii/S0167739X17311962?Dihub accessed on 19th January 2021.

[8] https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/ accessed on 30th March 2020.

[9] https://www.google.com/search?q=IoT+protocol+architecture%3A&rlz=1C1RXQR_enIN1102IN1102&oq=IoT+protocol+architecture%3A&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQABiABDIICAIQABgWGB4yCAgDEAAYFhgeMggIBBAAGBYYHjIICAUQABgWGB4yCAgGEAAYFhgeMggIBxAAGBYYHjIICAgQABgWGB4yDQgJEAAYhgMYgAQYigXSAQkxNDMwajBqMTWoAgiwAgE&sourceid=chrome&ie=UTF-8 accessed on 15th April 2020. accessed on 1st May 2020.

[10] https://link.springer.com/chapter/10.1007/978-981-19-2374-6_5 pdf accessed on 31th May 2022.

[11] https://www.scirp.org/journal/paperinformation.aspx?paperid=108574 accessed on 15th October 2022.

[12] http://www.wattics.com. Smart metering, accessed on 19th October 2022.

[13] https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/ # google_vignette, accessed on 29th November 2022.

[14] https://www.geeksforgeeks.org/fuzzy-logic-introduction/, accessed on 1st December 2022.

[15] https://www.hindawi.com accessed on 1st January 2023.

[16] https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/gtd2.12291 accessed on 4th January 2023.

[17] https://www.mqtt.org accessed on 1st February 2023.

[18] https://www.techtarget.com/iotagenda/definition/fog-computing-fogging accessed on 21st February 2023.

[19] https://www.dzone.com accessed on 25th March 2023.

[20] https://www.javatpoint.com/precision-and-recall-in-machine-learning accessed on 30th March 2023.

[21] https://www.paloaltonetworks.com/cyberpedia/what-is-quality-of-service-qos#:~:text=Quality%20of%20service%20(QoS)%20is,specific%20flows%20in%20network%20traffic. accessed on 11th April 2023.

[22] https://http://www.openfogconsortium.org/ra pdf accessed on 23rd May 2023.

[23] https://arxiv.org/abs/2011.14732 accessed on 28th May 2023.

[24] https://www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-steps accessed on 1st June 2023.

[25] https://www.mdpi.com/2079-9292/12/7/1511/ accessed on 7th June 2023.

[26] https://pac.pogil.org/index.php/pac/article/view/304 accessed on 9th June 2023.

[27] https://www.researchgate.net/publication/377025158_An_Analysis_of_Methods_and_Metrics_for_Task_Scheduling_in_Fog_Computing accessed on 17th June 2023.

[28] https://www.sam-solutions.com accessed on 29th June 2023.

[29] https://appquipo.com/blog/develop-ai-based-oms-software/ accessed on 30th June 2023.

[30] https://www.academia.edu/119104943/Resource_Allocation_and_Task_Scheduling_in_Fog_Computing_and_Internet_of_Everything_Environments_A_Taxonomy_Review_and_Future_Directions accessed on 7th July 2023.

[31] https://www.ibm.com/topics/machine-learning-algorithms accessed on 14th July 2023.

[32] https://www.simplilearn.com/tutorials/machine-learning-tutorial/confusion-matrix-machine-learning accessed on 27th July 2023.

[33] https://www.researchgate.net/publication/377457479_Machine_Learning_Approaches_To_Predict_The_Stability_of_Smart_Grid accessed on 29th July 2023.

[34] https://www.devopedia.org accessed on 7th August 2023.

[35] https://www.geeksforgeeks.org/fuzzy-logic-introduction/ accessed on 3rd ssSeptember 2023.

[36] https://dl.acm.org/doi/10.1145/3513002 accessed on 25th September 2023.

[37] https://www.geeksforgeeks.org/3-layer-iot-architecture/?ref=ml_lbp accessed on 22nd October 2023.

[38] https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/ accessed on 5th November 2023.

[39] https://www.google.com/search?q=round+robin+scheduling+algorithm+in+machine+learning&rlz=1C1RXQR_enIN1102IN1102&oq=round+robin+scheduling+algorithm+in+ma&gs_lcrp=EgZjaHJvbWUqBwgBECEYoAEyBggAEEUYOTIHCAEQIRigATIHCAIQIRigATIHCAMQIRigATIHCAQQIRifBdIBCjE3NzkwajBqMTWoAgiwAgE&sourceid=chrome&ie=UTF-8 accessed on 18th November 2023.

[40] https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html accessed on 24th November 2023.

[41] https://www.geeksforgeeks.org/5-layer-architecture-of-internet-of-things/?ref=ml_lbp accessed on 28th November 2023.

[42] https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ accessed on 1st December 2023.

[43] https://www.javatpoint.com/confusion-matrix-in-machine-learning accessed on 6th December 2023.

[44] https://www.javatpoint.com/how-to-check-the-accuracy-of-your-machine-learning-model accessed on 26th December 2023.

[45] Sensors | Free Full-Text | Simulation Tools for Fog Computing: A Comparative Analysis (mdpi.com) accessed on 1st January 2024.

[46] https://www.geeksforgeeks.org/difference-between-sdn-and-nfv/ accessed on 11th January 2024.

[47] https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/why-sdn-software-defined-networking-or-nfv-network-functions-virtualization-now/ accessed on 17th January 2024.

[48] https://www.analyticsvidhya.com/blog/2015/12/improve-machine-learning-results/ accessed on 1st February 2024.

[49] https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/heuristic-function-in-ai accessed on 14sth February 2024.

[50] https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-heuristics/ accessed on 7th March 2024.

[51] https://www.comsoc.org/publications/magazines/ieee-communications-magazine/cfp/future-trends-fogedge-computing accessed on 15th April 2024.

[52] https://www.researchgate.net/publication/320855949_Fog_Computing_Issues_Challenges_and_Future_Directions accessed on 1st June 2024.

[53] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10099336/ accessed on 1st June 2024.

[54] https://www.researchgate.net/figure/iFogSim-Architecture-Adapted-from-29_fig1_369571348 accessed on 5th June 2024.

[55] https://www.researchgate.net/publication/349710974_LEAF_Simulating_Large_Energy-Aware_Fog_Computing_Environments accessed on 7th June 2024.

[56] https://www.geeksforgeeks.org/what-is-mipsmillion-of-instructions-per-second/ accessed on 7th June 2024

APPENDIX

Appendix: List of Publications and Conferences Attended

1) Suraj Rajaram Nalawade, Dr. Ashok Kumar Jetawat, "Use of Clustering Machine Learning Algorithms in Fog Computing for Task Scheduling and Resource Allocation" has been published in European Chemical Bulletin (ISSN: 2063-5346), Volume 11, Issue 8, 2022 Date of Publication: - August 2022.

2) Suraj Rajaram Nalawade, Dr. Ashok Kumar Jetawat, "A Comparative Study of Various Classification Machine Learning Algorithms in Fog Computing: Task Scheduling" has been published in Industrial Engineering Journal (ISSN 0970-2555), Volume: 52, Issue 5, No. UGC Care Approved, Group I, Peer Reviewed Journal 4, May: 2023.

3) "The Survey on Fog Computing and its Applications" International Virtual Conference on "Emerging Era of Applications of Computer, 15th -16th of January 2022 Organized by Pacific University Udaipur.

4) National Seminar on "Implementation of Academic Bank of Credit (ABC) in Higher Education Institutes" on 21st March 2023 Organized by Avinashilingam Institute for Home Science and Higher Education for Women University Udaipur.

5) IP Awareness Training Program under "National Intellectual Property Awareness Mission" Organized by Intellectual Property Office, India on 18, January 2023.