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[1], 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

[1]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[2].

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[3], such as smart energy and building automation, and it may be implemented through a UDP[4].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[5] paradigm. According to this model, servers make resources available under a URL[6], 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

[2]Transport Layer Security
[3]Light-Weight Machine-To-Machine Communication
[4]User Datagram Protocol
[5]Representational State Transfer
[6]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[7]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[8] 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[9] 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.

4.1.3 Advanced Message Queuing Protocol

[7]Confirmable Message
[8]Acknowledgement
[9]Non-confirmable

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.

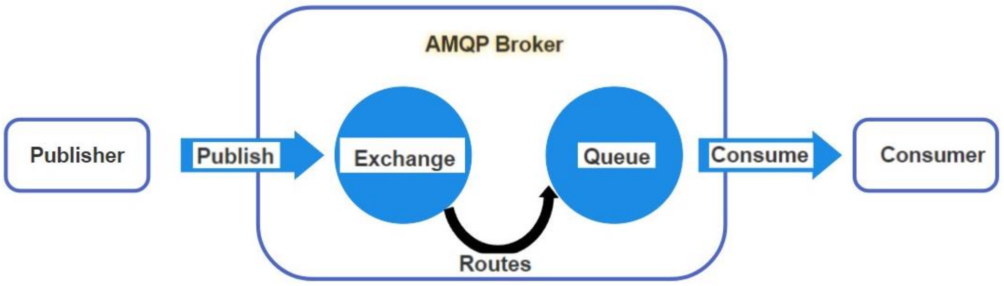


Figure 4.1: AMQP Architecture (Macarulla, 2016)

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[10] 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[11] 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.

[10]Data-Centric Publish-Subscribe
[11]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[12]	Task Scheduling	Optimized	Unoptimized	Optimized	Optimized	Unoptimized
PERA[13]	Task Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized
WRR[14]	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

[12]First-Come, First-Served
[13]Packetized Ensemble Resource Allocation
[14]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[15]	Resource Scheduling	Optimized	Unoptimized	Optimized	Optimized
MILP[16]	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[17]	Task	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized	Unoptimized
PTPN[18]	Resource	Unoptimized	Unoptimized	Optimized	Unoptimized	Optimized	Unoptimized
MCCV[19]	Resource	Unoptimized	Unoptimized	Optimized	Optimized	Unoptimized	Unoptimized
EDF &LFC[20]	Resource	Optimized	Optimized	Unoptimized	Optimized	Unoptimized	Unoptimized
DOTS[21]	Resource	Optimized	Unoptimized	Unoptimized	Optimized	Unoptimized	Unoptimized
TIPS[22]	Task / Resource	Unoptimized	Unoptimized	Optimized	Unoptimized	Unoptimized	Unoptimized

[15]Integer Linear Programming
[16]Mixed Integer Linear Programming
[17]Shortest Job First
[18]Preemptive Task Priority Network
[19]Minimum Critical-Cycle Variance
[20]Earliest Deadline First and Least Slack Time
[21]Dynamic Optimization of Time Sequences
[22]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[23]	Resource Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized	Unoptimized
FLPSO[24]	Resource Scheduling	Unoptimized	Unoptimized	Optimized	Unoptimized	Unoptimized	Unoptimized
FPFTS[25]	Resource Scheduling	Optimized	Unoptimized	Unoptimized	Unoptimized	Optimized	Optimized
EDA[26]	Resource Scheduling	Optimized	Unoptimized	Optimized	Unoptimized	Optimized	Optimized

[23] Rule-based Fuzzy Network
[24] Fuzzy Logic and Particle Swarm Optimization
[25] Fuzzy-Possibilistic Fuzzy Time Series
[26] 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.

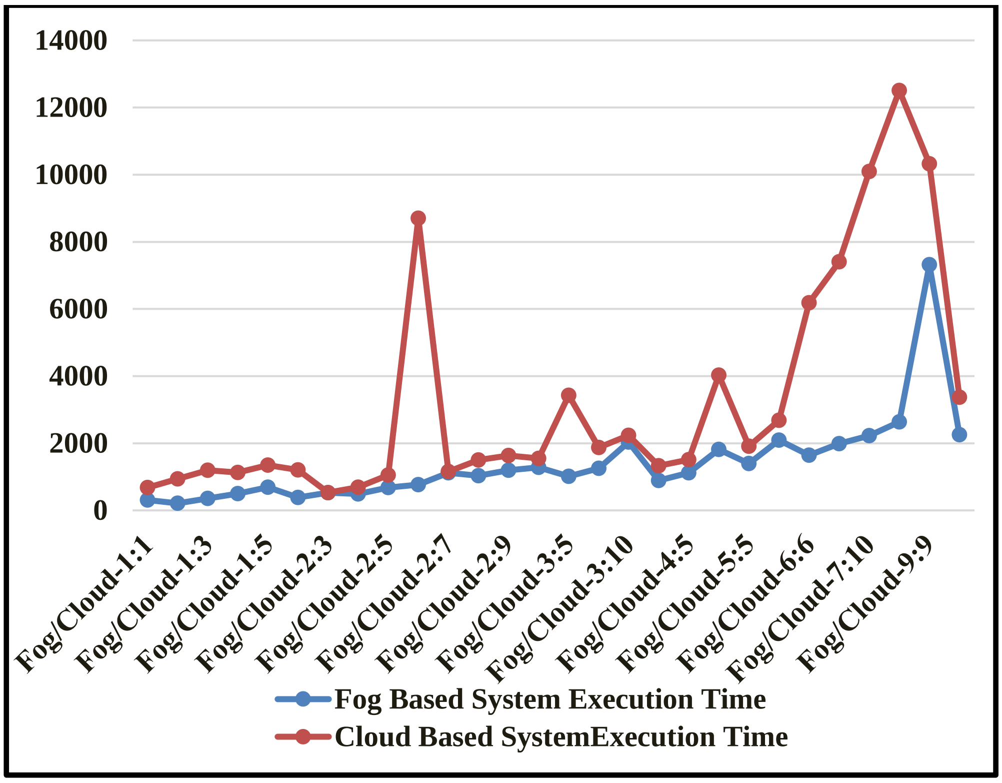
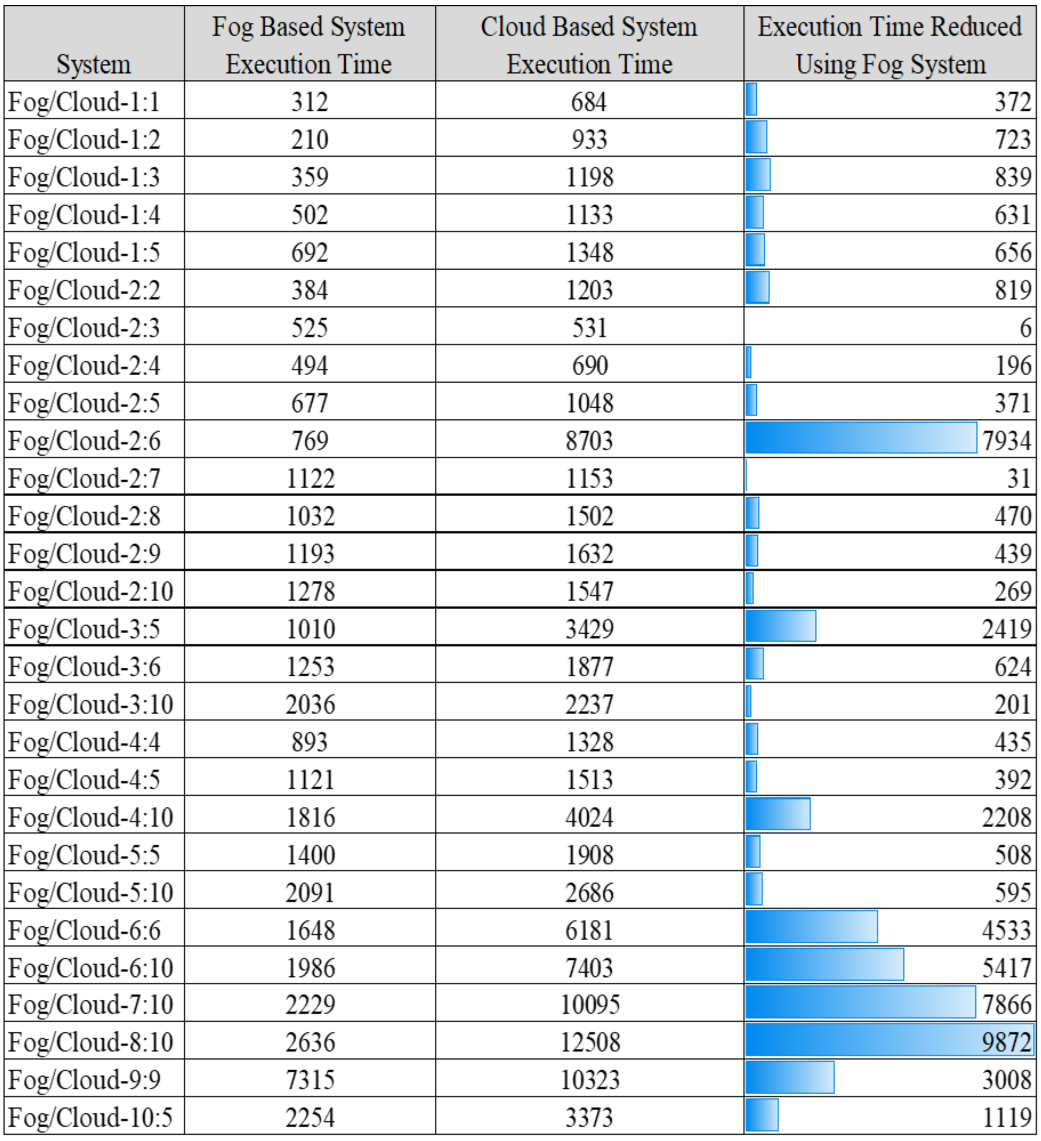


Figure 4.2: Fog Vs Cloud System Based on Average Execution Time

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 χ2			
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 (∑)	**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.

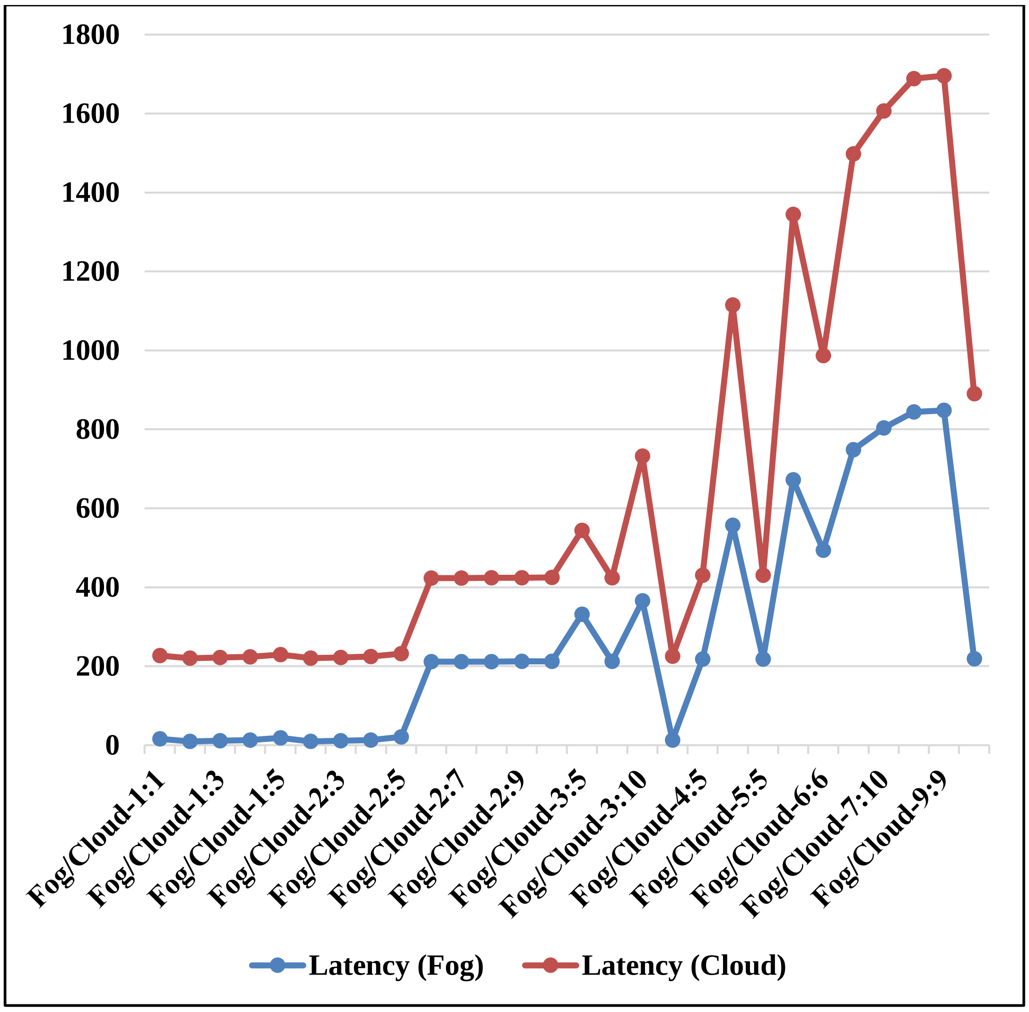


Figure 4.3: Fog Vs Cloud System Based on Latency

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.

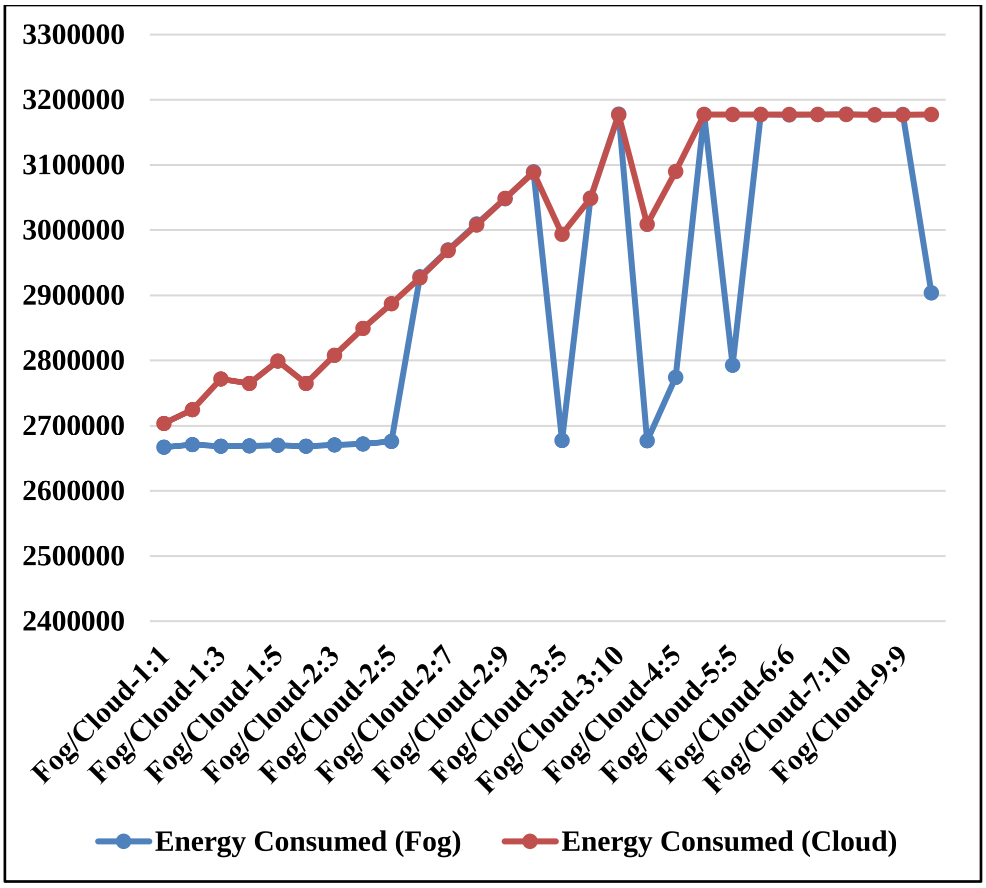


Figure 4.4: Fog Vs Cloud System Based on Energy Consumption (Joules)

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.

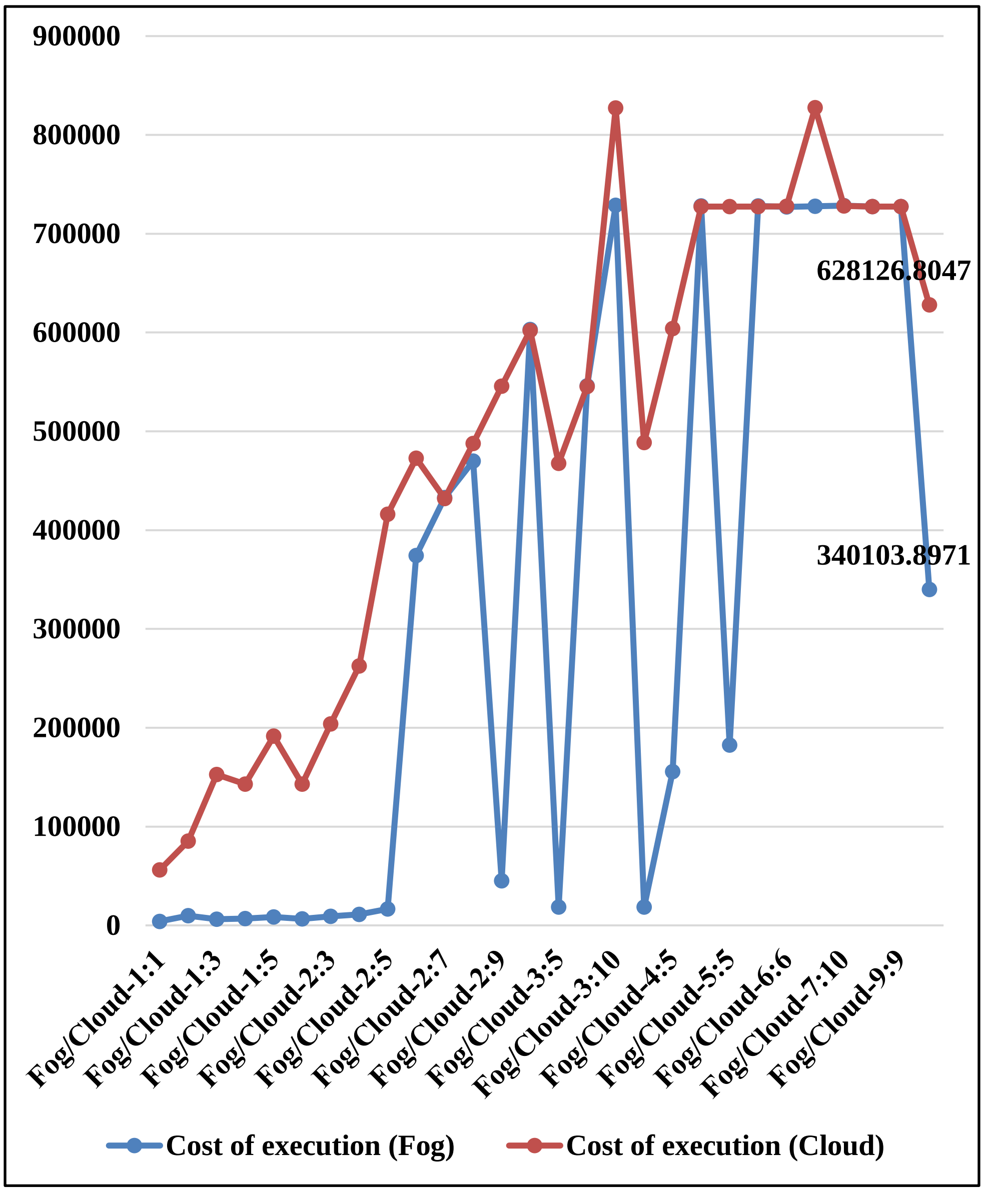


Figure 4.5: Fog Vs Cloud System Based on Cost of Execution (ms)

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_05: 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.

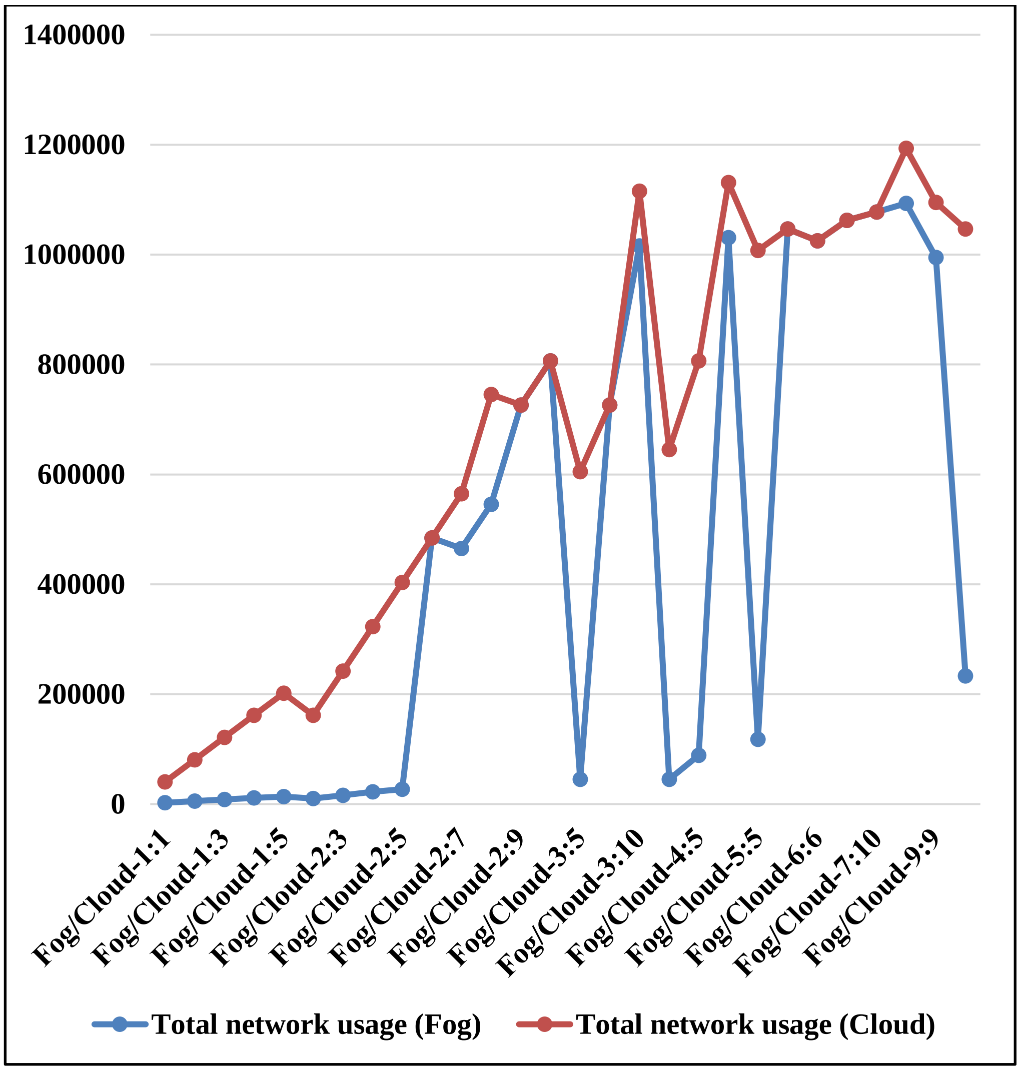


Figure 4.6: Fog Vs Cloud System Based on Total Network Usage (B/s)

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.

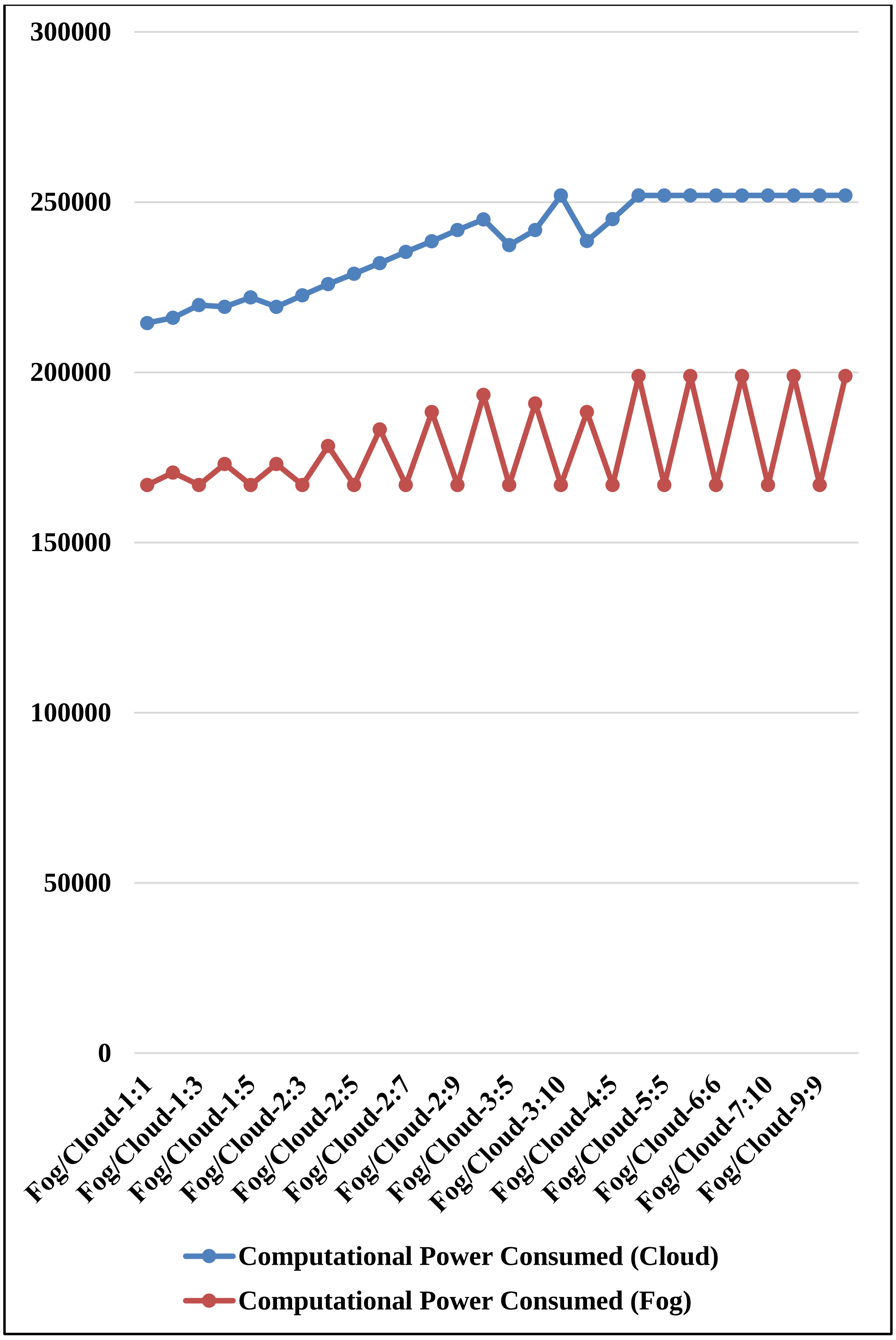


Figure 4.7: Fog Vs Cloud System Based on Computational Power (W)

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[27] 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

[27] 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[28] 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.

[28]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 $\longrightarrow$ No. of Areas

Energy Consumed $\longrightarrow$ No. of Cameras per Area

Energy Consumed $\longrightarrow$ Execution Time

Energy Consumed $\longrightarrow$ Average Loop Delay

Energy Consumed $\longrightarrow$ CPU Delay

Energy Consumed $\longrightarrow$ Latency

Energy Consumed $\longrightarrow$ Cost of Execution

Energy Consumed $\longrightarrow$ 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[29]) 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[30]	ROC[31] Area	PRC[32] 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.

[29]Logistic Regression
[30]Matthews Correlation Coefficient
[31]Receiver Operating Characteristic
[32]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.

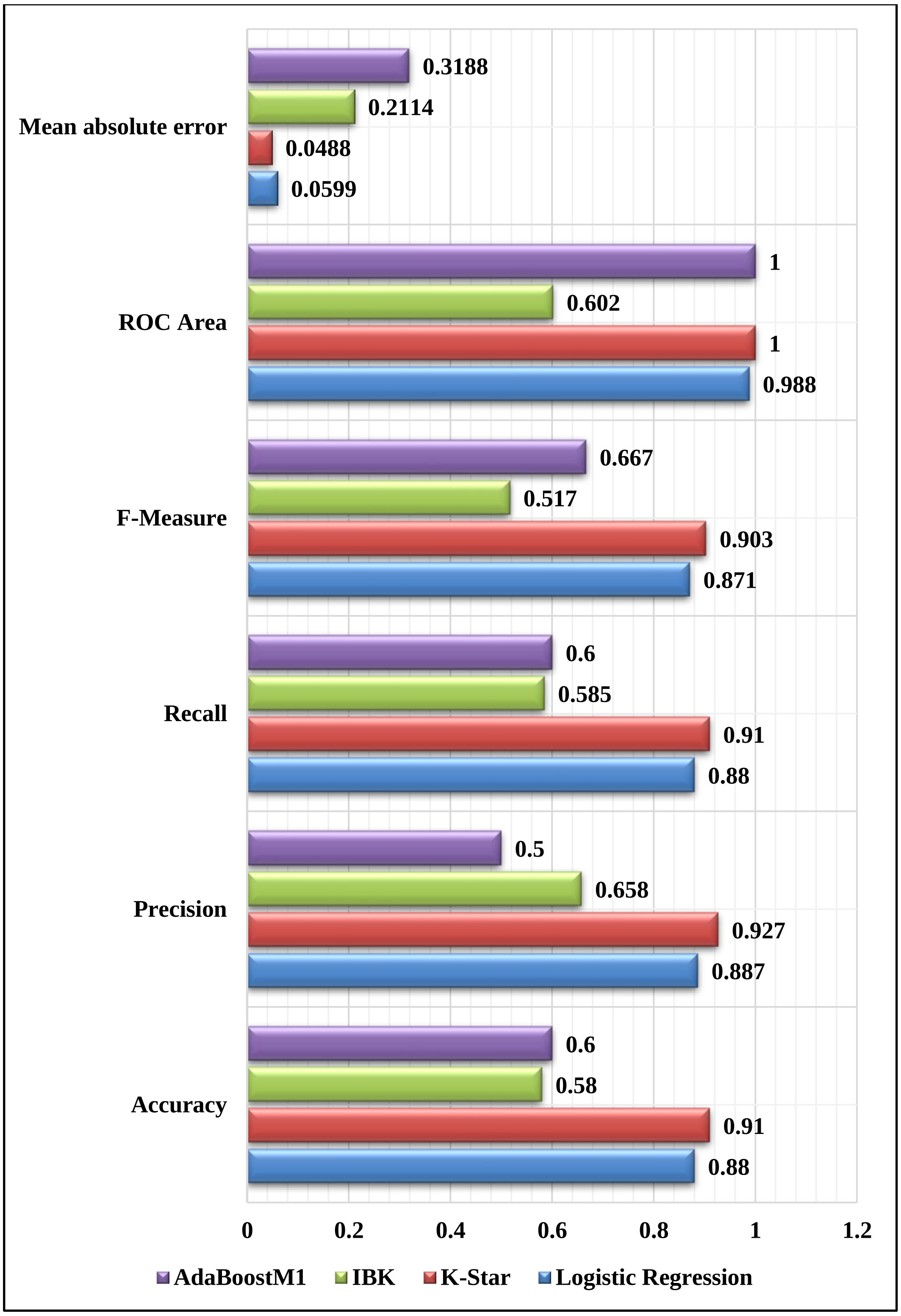


Figure 4.8: Evaluation of Classifier at 10-fold Cross-Validation based on Various Performance Measures

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.

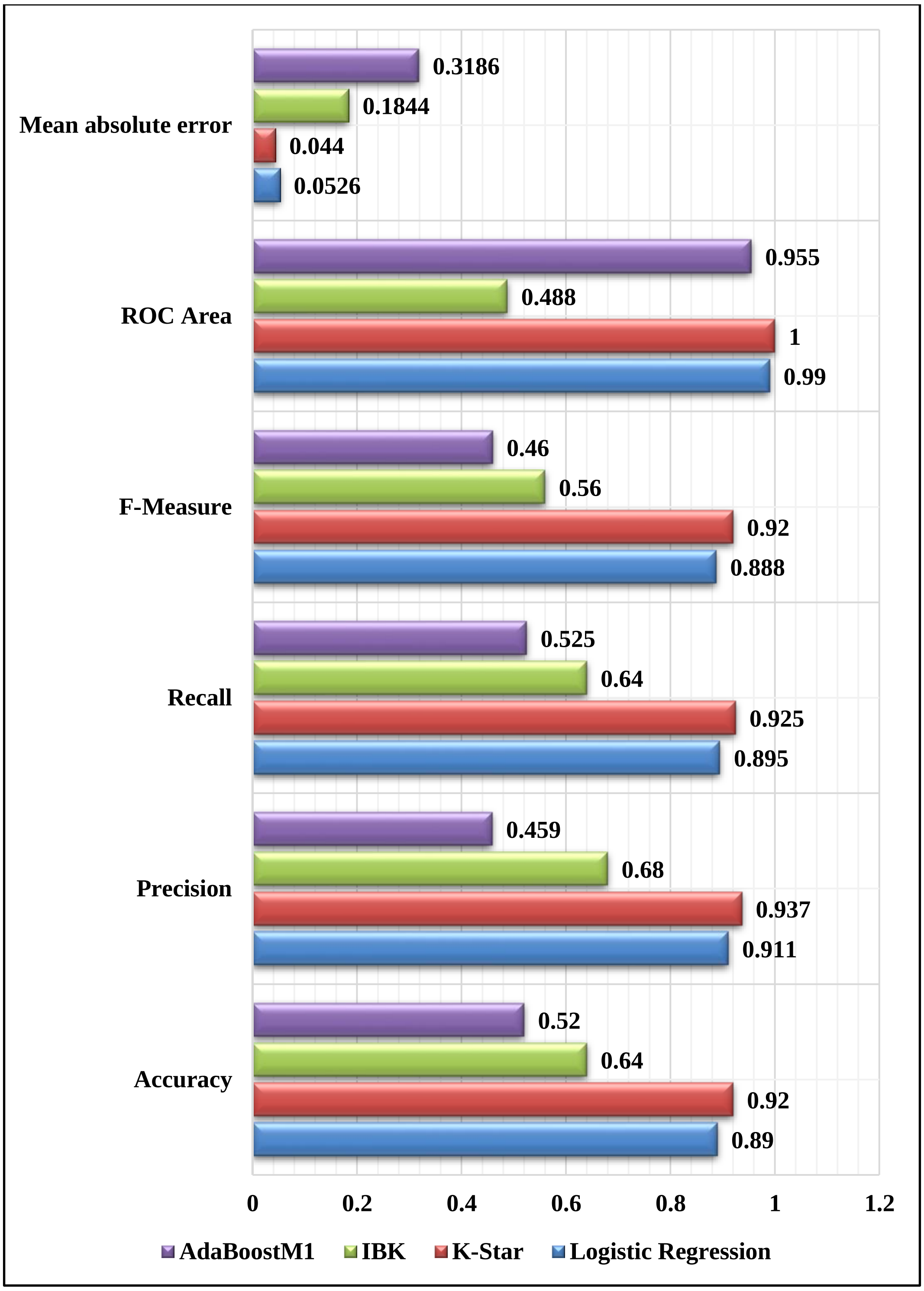


Figure 4.9: Evaluation of Classifier at 25-fold Cross-Validation based on various Performance Measures

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.

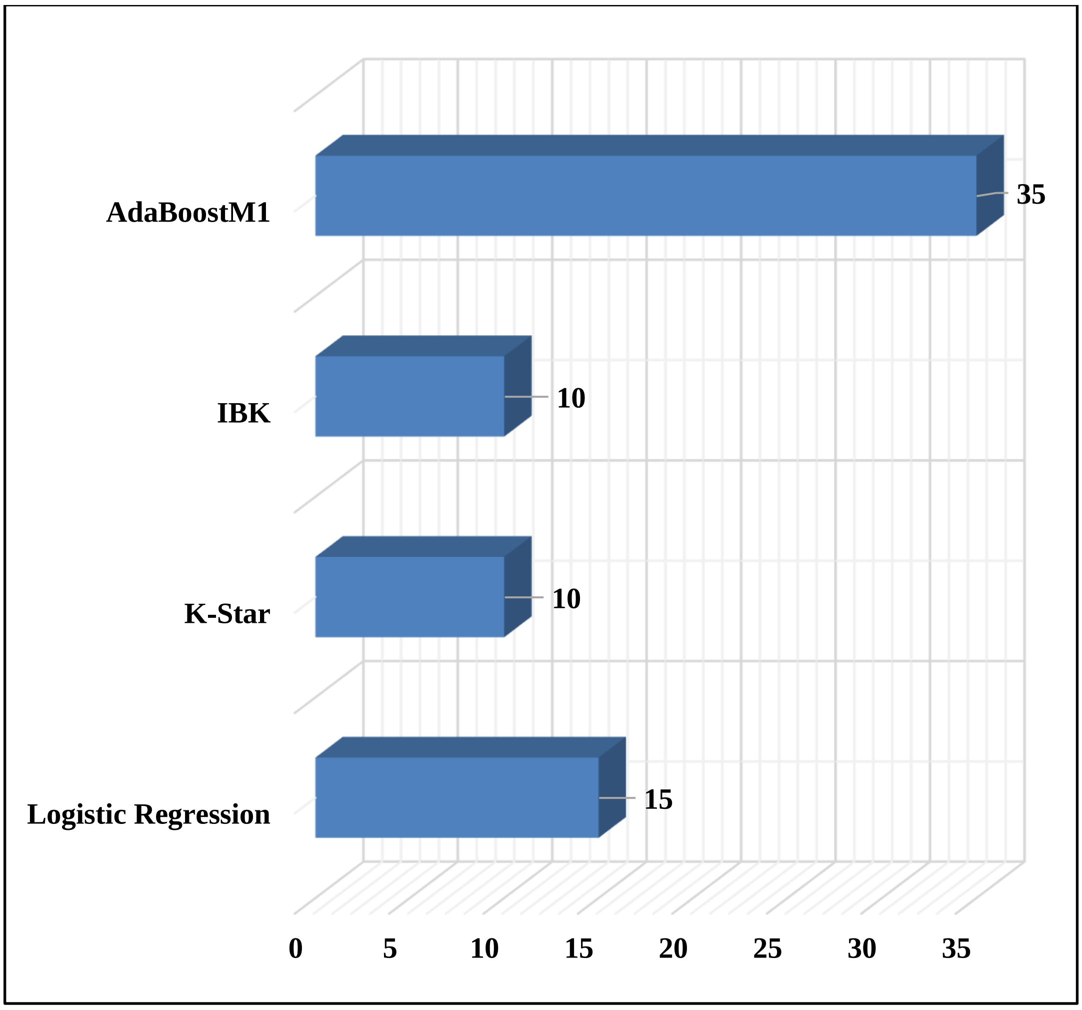


Figure 4.10: Average Execution Time (ms): 25 folds

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.

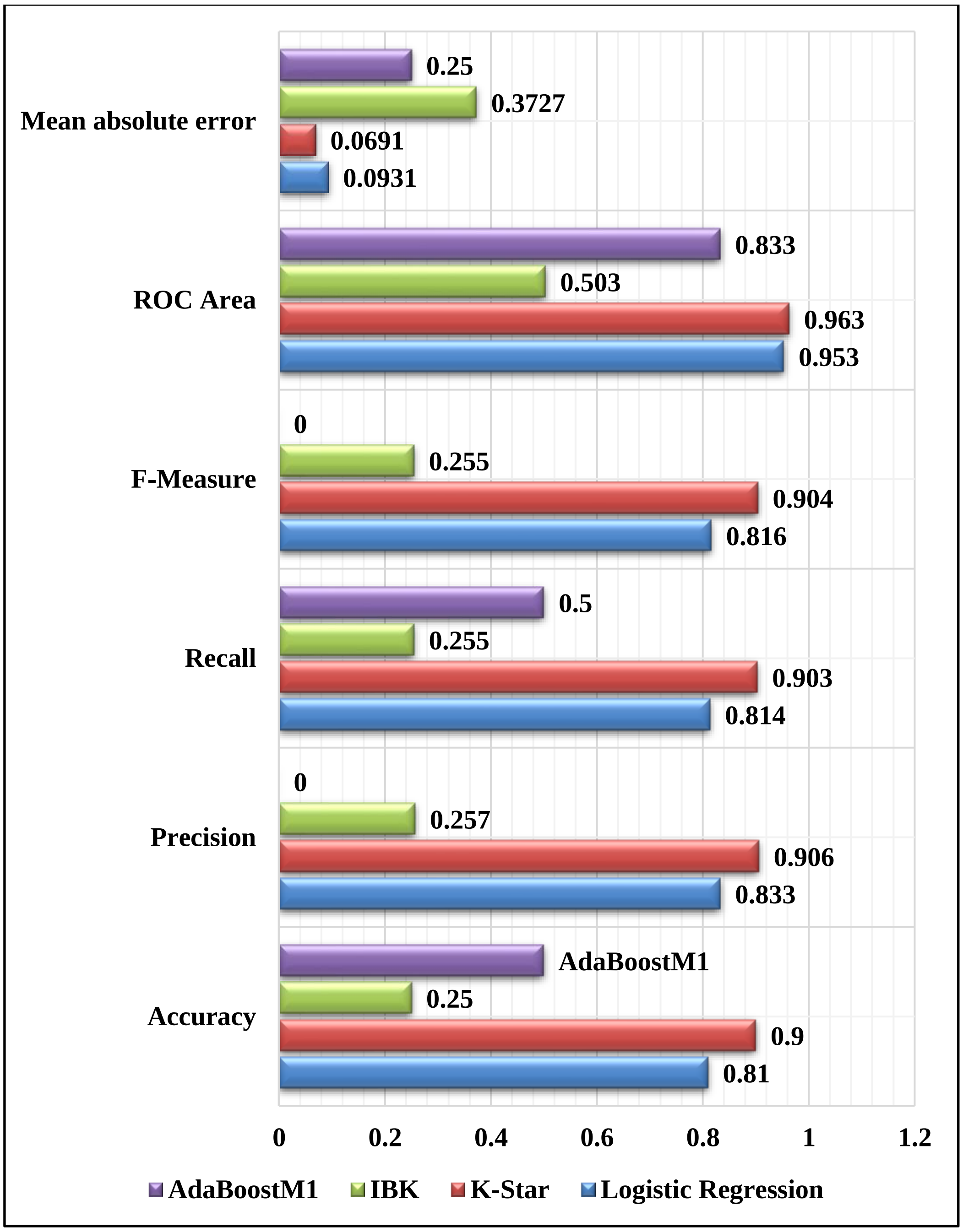


Figure 4.11: Evaluation of Classifier at 10-fold Cross-Validation Based on Various Performance Measures (Number of Tasks: 160 and Nodes: 4)

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.

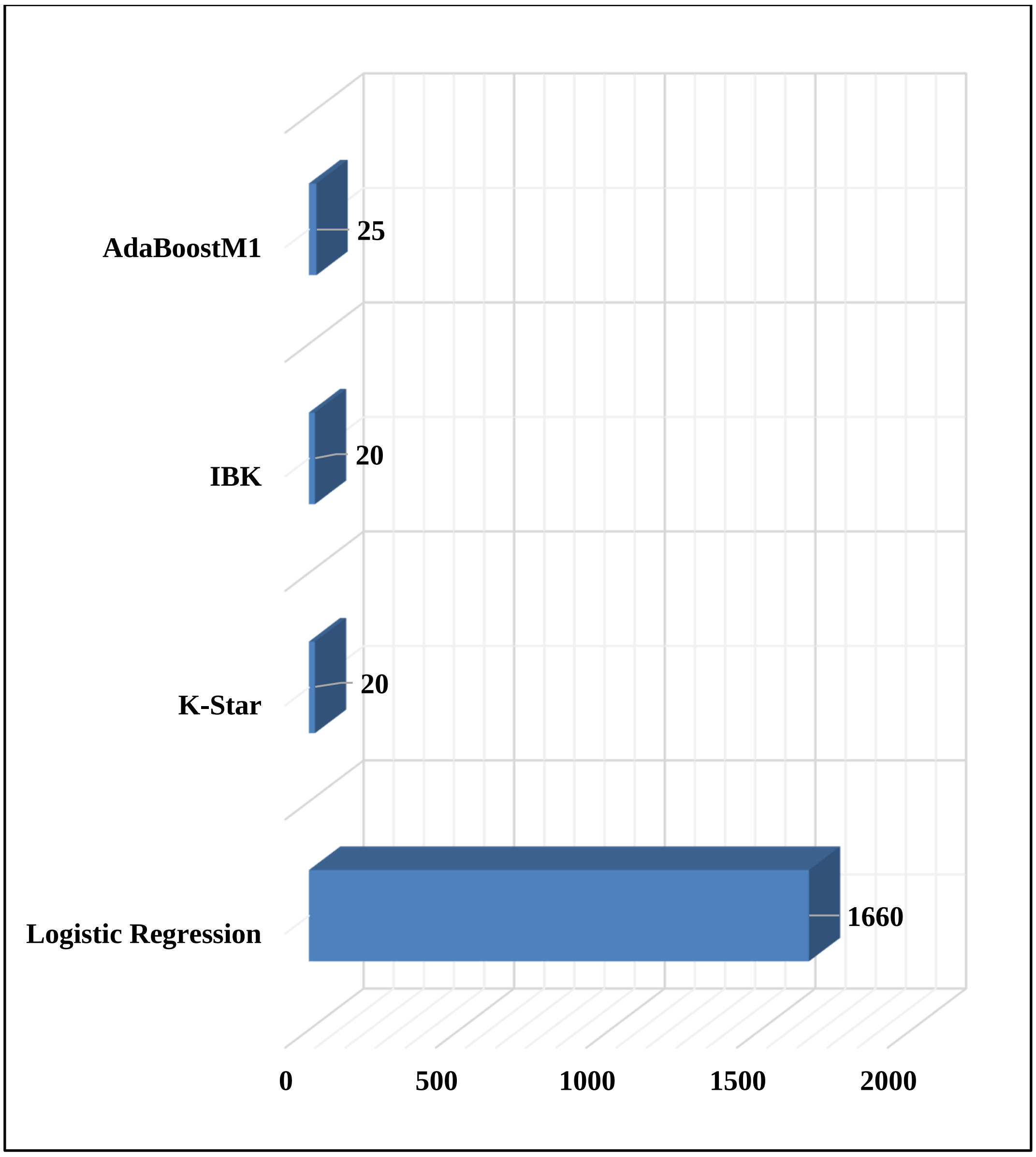


Figure 4.12: Average Execution Time (ms): 10 folds (Number of Tasks: 160 and Nodes: 4)

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.

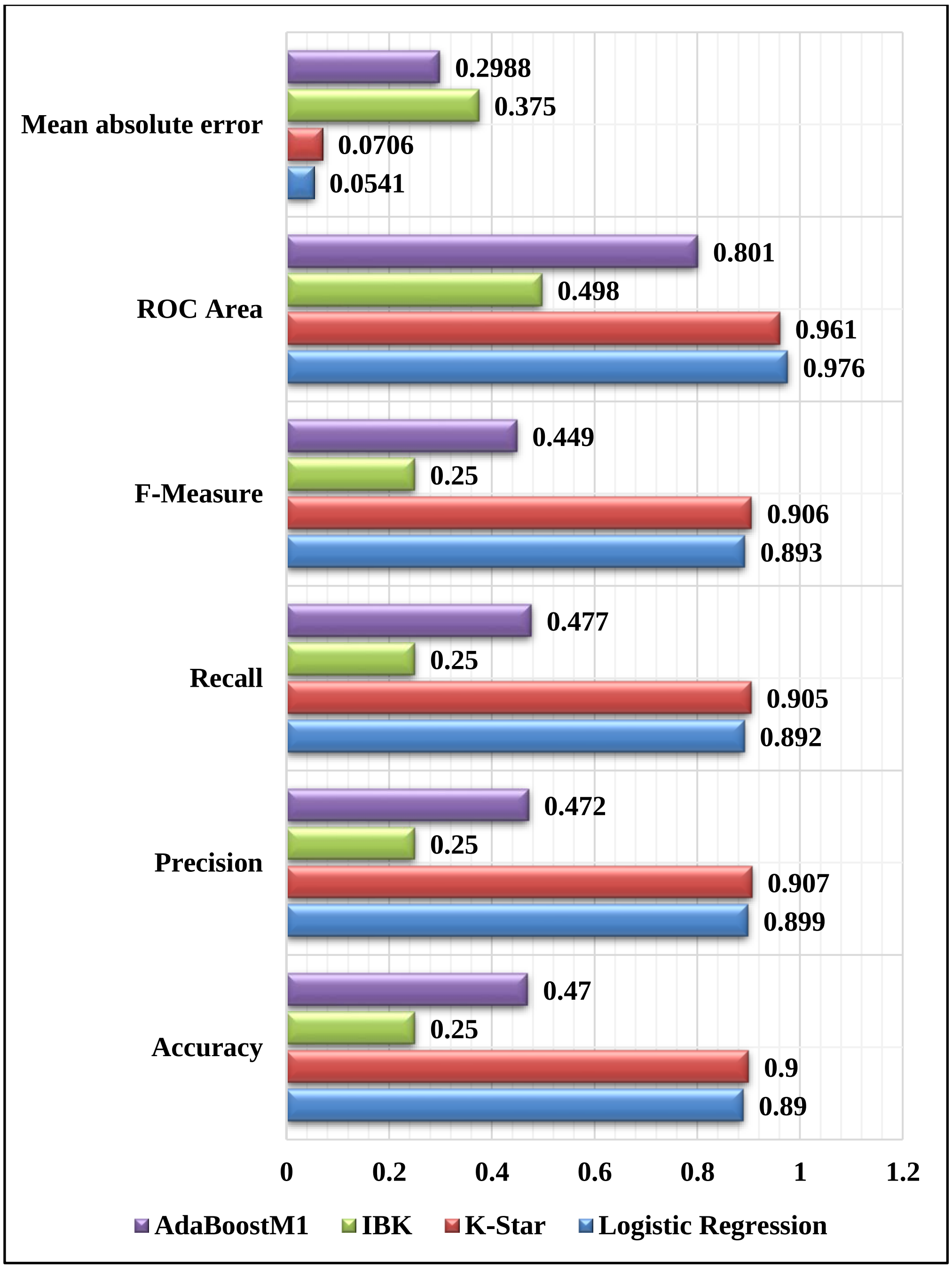


Figure 4.13: Evaluation of classifier at 25-fold Cross-Validation based on Various Performance Measures (Number of Tasks: 160 and Nodes: 4)

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.

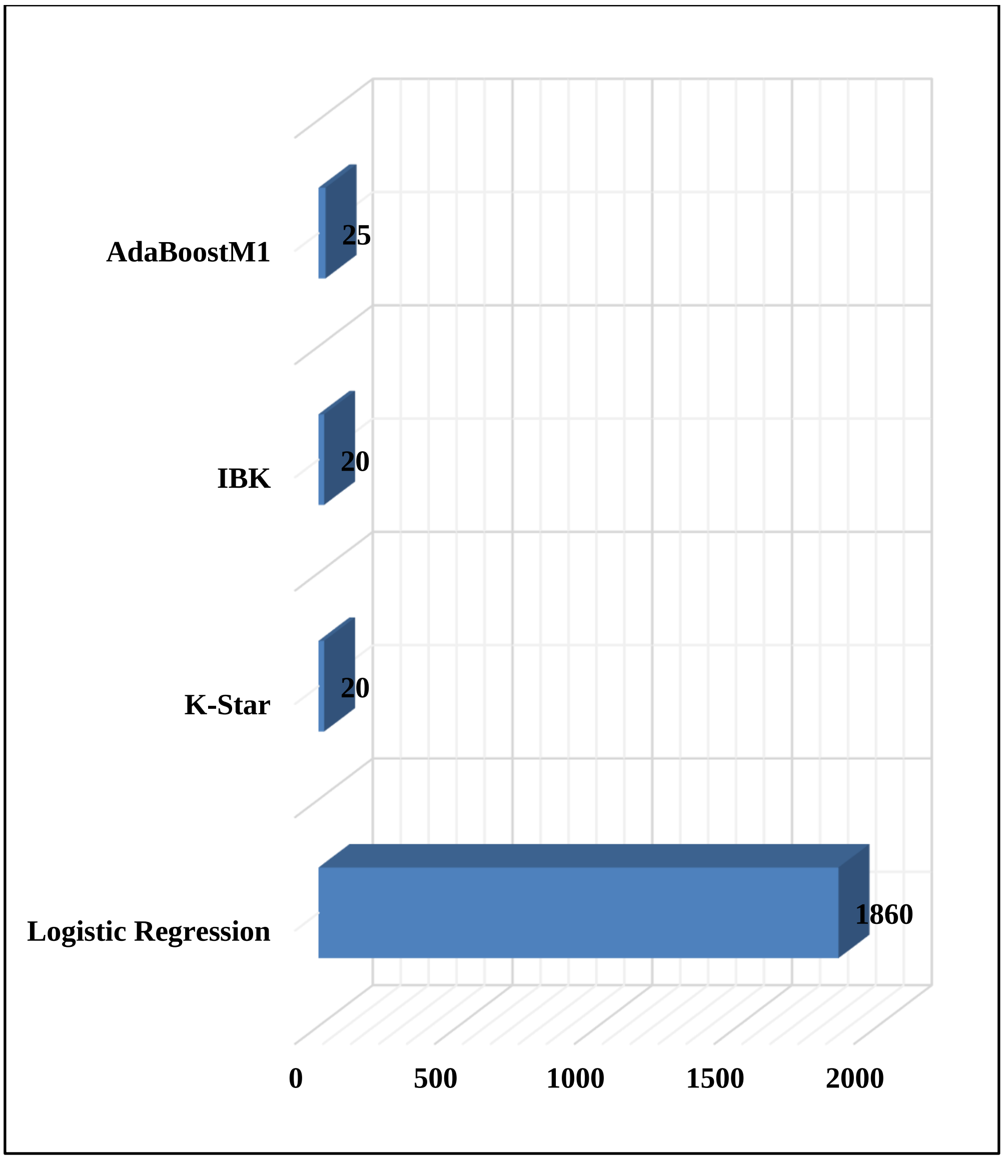


Figure 4.14: Average Execution Time (ms): 25 folds (Number of Tasks: 160 and Nodes: 4)

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[33], 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.

[33] 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.

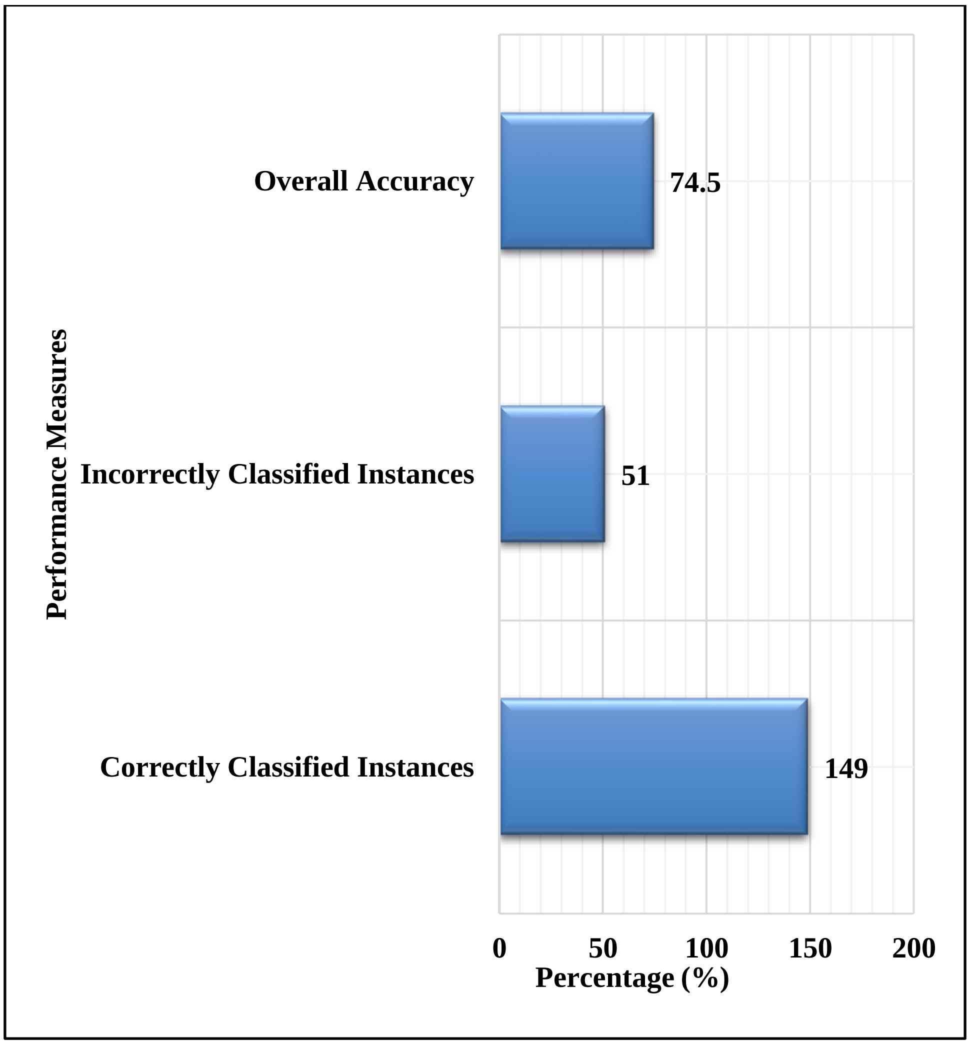


Figure 4.15: Overall Accuracy Canopy Clustering

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.

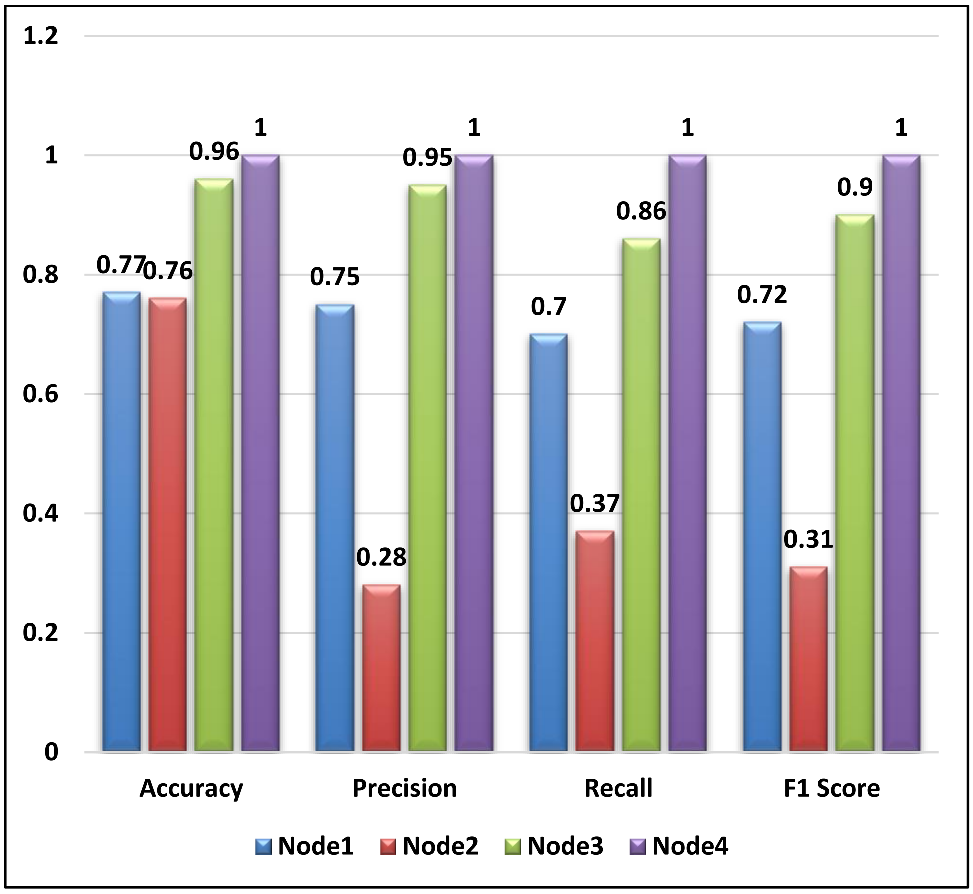


Figure 4.16: Class-wise performance measures

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.

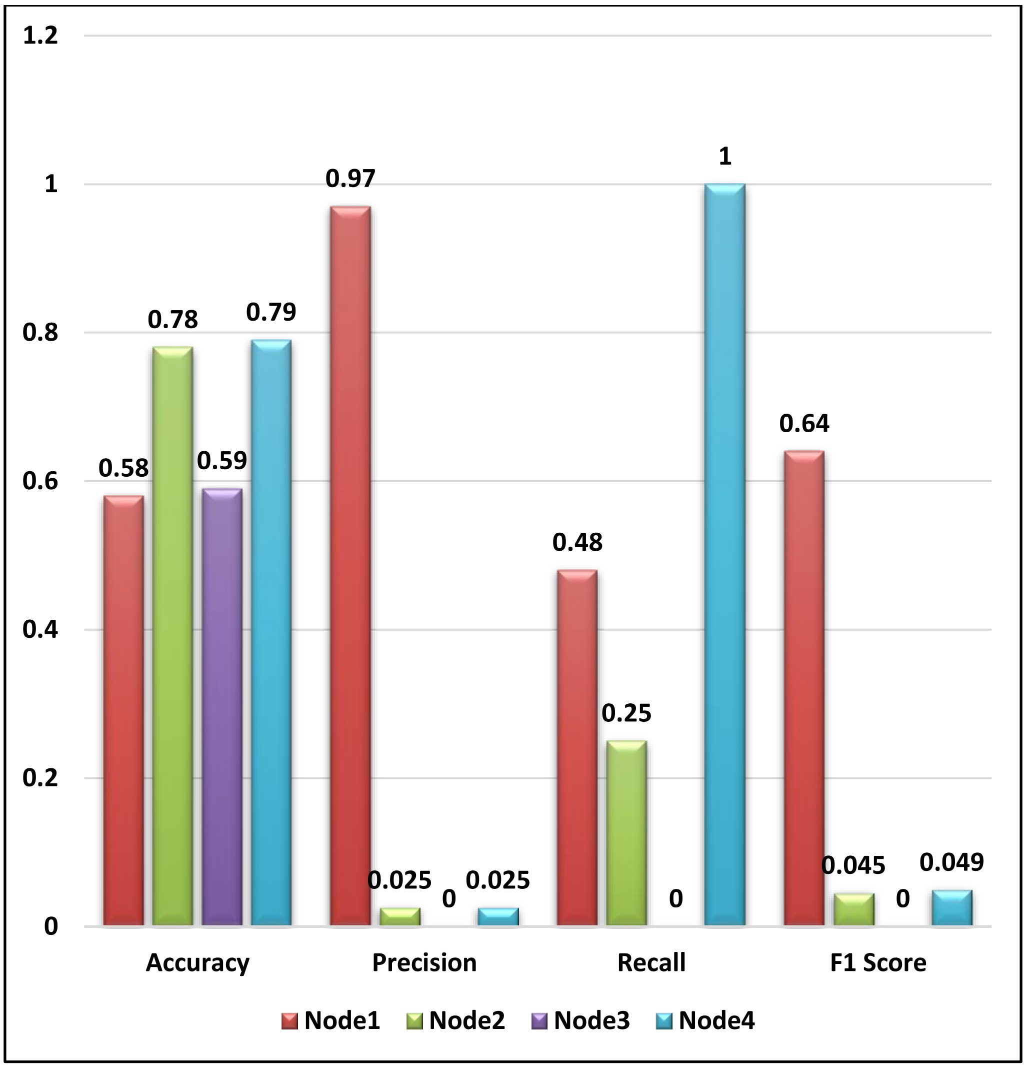


Figure 4.17: Class or Node-wiseHierarchical Clustering Performance Measures

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.

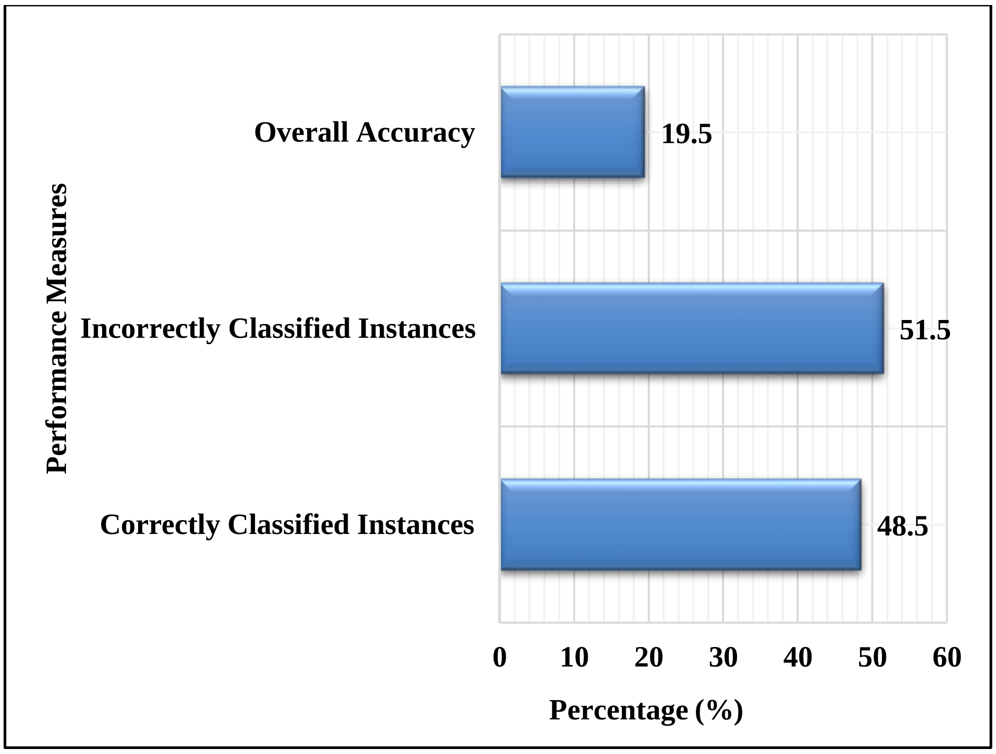


Figure 4.18: Overall Accuracy Make Density-Based Clustering

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.

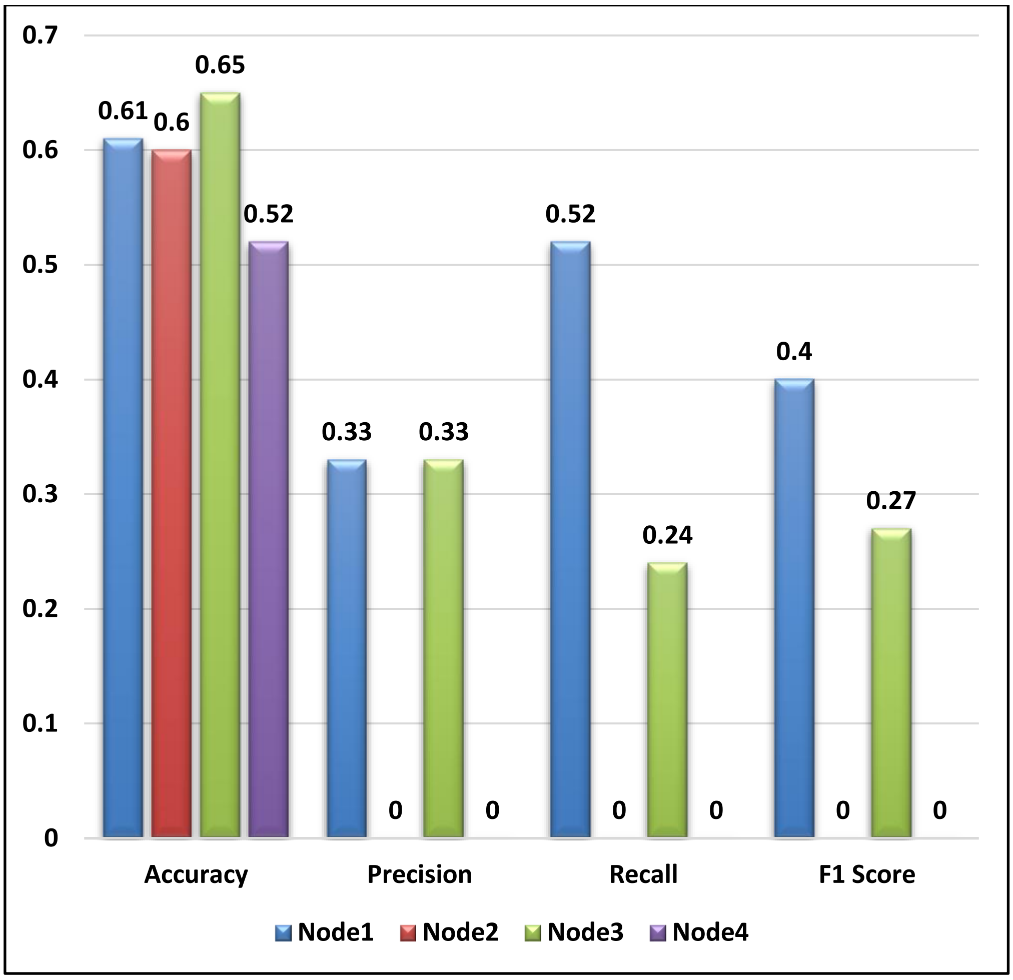


Figure 4.19: Class or Node wise Make Density-Based Clustering Performance Measures

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.