Journal of
Agriculture and Biology
ISSN: 3007-1755 (Print), 3007-1763 (Online)

Check for updates

ROOTS PRESS

Research Article

Reliable Edge Federation Formation for Smart Agriculture using Machine Learning

Usama Ahmed[1*], Afzaal Hussain[2], Muhammad Nasir Siddiqui[3], Abid Ali[4], Muhammad Adeel Zahid[5], Shehbaz Nazeer[2]

[1] *Department of Software Engineering, Government College University, Faisalabad, Pakistan.*
[2] *Department of Information Technology, Government College University, Faisalabad, Pakistan.*
[3] *Institute of Computing, MNS University of Agriculture, Multan, Pakistan.*
[4] *Department of Computer Science, University of Agriculture Faisalabad, Pakistan.*
[5] *Center for Data Science, Government College University, Faisalabad, Pakistan.*

ABSTRACT

In the domain of smart agriculture, the integration of edge computing has revolutionized data processing by bringing computational services in close proximity to sensors and movable equipment. The transition to edge computing reshapes the security landscape, where robust safeguards are essential not only for protecting data but also for ensuring reliable service delivery. In agriculture, reliability is critical, as insecure or untrustworthy ESPs can cause interruptions and downtime that undermine both federation performance and provider reputation. The overall challenge is to form a resilient edge federation that enables the seamless sharing of services from trustworthy providers to clients, such as farm equipment and monitoring systems. To address this, a new algorithm Edge Nearest Neighbor (ENN) as an extension of the k-nearest neighbor (kNN) machine learning approach has been presented. The proposed algorithm selects highly reliable ESPs based on user requirements to form a federation, ensuring smart services are delivered without network interruption. This enables precision agriculture systems to reliably monitor and maintain large arable lands effectively. Experiments demonstrate that ENN quickly forms a federation of reliable ESPs with high accuracy, providing a secure and dependable foundation for smart agriculture.

Keywords: Smart Agriculture, IoT, Edge Computing, Reliability.

OPEN ACCESS

Correspondence
Usama Ahmed
usamaahmed@gcuf.edu.pk

Article History
Received: May 22, 2025
Accepted: August 27, 2025
Published: September 09, 2025

INTRODUCTION

The global agricultural sector faces the unprecedented challenge of feeding a growing population amidst the pressures of climate change, water scarcity, and limited arable land. In response, traditional farming is rapidly evolving into smart agriculture, a paradigm that leverages advanced information and communication technologies (ICT) to optimize productivity, efficiency, and sustainability (Choudhary et al., 2025). This transformation is driven by the deployment of expansive Internet of Things (IoT) networks, comprising myriad sensors and actuators that generate immense volumes of data on soil, crops, and micro-climates in real-time, providing the foundation for machine learning–driven analysis and decision-making (Özden and Karadoğan, 2024; Umarani and Baskaran, 2024).

However, the vast, continuous data streams generated by these distributed agricultural IoT (Agri-IoT) devices pose a significant computational challenge (Kumar et al., 2024). While cloud computing has been a historical solution, its limitations, notably latency, bandwidth costs, and reliability issues in remote rural areas, are now major bottlenecks for time-sensitive agricultural operations (Dhifaoui et al., 2024). This is particularly critical for applications like real-time automated

weeding, drone-based spot spraying, and instantaneous disease detection, where delays of even a few seconds can render an intervention ineffective. Edge computing has emerged as the pivotal enabler to overcome these bottlenecks (Dong et al., 2024; Sharma et al., 2024). By relocating computational and storage resources directly to the farm, near the data sources, it facilitates instant data processing, ensures operational resilience without constant internet connectivity, and significantly reduces the volume of data transmitted to the cloud (He et al., 2024). It works including both downstream and upstream data in support of cloud and IoT applications, accordingly. Any computing or networking equipment that sits among cloud based datacenters and data sources is referred to as an edge device. A smartphone operating among the cloud and body sensors, or a micro-datacenter (mDC) or cloudlet located between a mobile device and the cloud, is an example of an edge device (Shi and Dustdar, 2016). The end device in edge computing not only consumes but also generates data. Devices at the network edge not only request services and information from the cloud, but also manage computing tasks such as data storage, processing, load balancing, and caching. The edge has to be well designed to handle the tasks efficiently, reliably, securely and with the awareness of the privacy.

The origins of edge computing may be traced back to the late 1990s, when Akamai introduced content delivery networks (CDNs) to improve online speed (Dilley et al., 2002). A CDN uses nodes to prefetch and cache web content at the edge, close to customers. The edge nodes can also modify data, for as by displaying location-relevant advertisements. The use of cloud computing platform, edge computing improves and extends the CDN concept. The proximity of cloudlets to end customers is critical, much as it is for CDNs. A cloudlet, like cloud computing, may run malicious program instead of primarily storing web information. For security, isolation, resource management, and metering, the code is frequently enclosed in a lighter-weight container or a virtual machine (VM).

Edge computing has a number of advantages. Different researchers displayed that for wearable cognitive-assistance systems cloudlet to offload computing actions can be used to improve reaction times while using less energy. CloneCloud technique reduces reaction times and power consumption in edge computing applications that have been evaluated (Chun et al., 2011). Experts in edge computing should be aware of a number of challenges, including the system's dependability. Because of their limited battery life or insufficient wireless connections, energy-constrained edge devices may fall short of meeting milestones. Developers should however allow the system to perform its core functions in these situations. Additional concerns which has been considered as main challenges are security and privacy. On one side, edge computing protects data better than cloud computing since processing takes place closer to the data origin. On the other side, it is more difficult to support the security and privacy in edge computing due to network architecture, sensor unreliability, and the large quantity of low-cost personal devices in the organization. The architecture of the edge computing has been shown in figure 1.

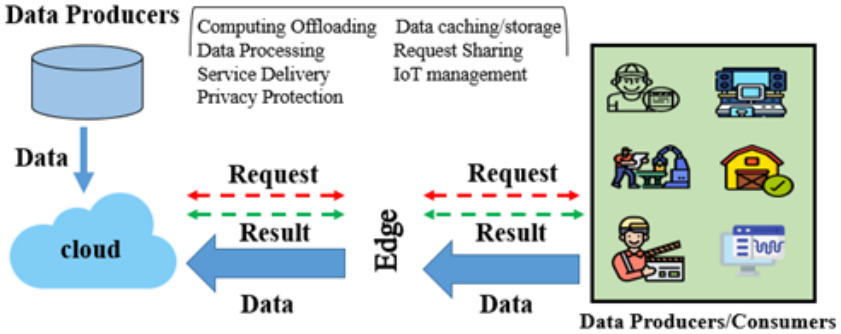

Figure 1 Architecture of Edge Computing

The two-way computation streams in edge computing are depicted in Figure 1. In the edge computing architecture, the objects are not just data consumers; they're also data generators. Things at the edge can not only demand services and content from the cloud, but they can even use the cloud to execute computational tasks. Computing offloading, data storage, caching, and processing, as well as distributing demands and delivering services from the cloud to the user, may all be done at the edge. To accomplish these responsibilities in the network, the edge must first be well-designed to meet the needs of services such as privacy protection, security, and reliability (Shi et al., 2016). In smart agriculture, such reliability directly supports critical tasks like crop disease detection, irrigation control, and drone-based monitoring, where even minor service disruptions can lead to significant losses.

The reliability of the edge federation can be damaged if the services are distributed by the insecure and untrustworthy edge provider. In this research, edge federation formation using machine learning based on reliability has been proposed to overcome the challenge of the reliability in the edge computing. This research has proposed ENN algorithm which is an extension of k-nearest neighbor (KNN). ENN is a machine learning approach will be used to overcome the problem of unreliable ESPs and enhance the reliability of edge federation. KNN is a learning method for categorization based on instances. KNN takes all the data into consideration and uses k nearest neighbors to classify the unknown sample and assign it to a class based on its nearest neighbor using the Euclidean distance. However, the conducted research does not work on labeling the unknown ESPs because it takes the reliability into consideration of the ESPs to form an edge federation. ENN uses the two different distance formula to calculate the reliability of ESPs against a vector of parameters. Experiments have been performed in google colab and observed that ENN algorithm formed federation of reliable ESPs in shorter span of time than kNN algorithm.

LITERATURE REVIEW

his research mainly concentrates in this section on presenting the previous work has been done on cloud and edge federation formation. In (Qammar et al., 2024) Qammar et al. address two critical challenges in federated learning (FL) i.e. the single-point-of-failure risk, and accuracy and privacy issues arising from random edge node selection. To mitigate these, they propose a blockchain-powered framework featuring three smart contracts—for node registration, forward bidding-based node selection, and payment settlement—and employ CKKS fully homomorphic encryption to secure model updates. Evaluations on benchmark datasets demonstrate improved model accuracy, reduced loss, and enhanced privacy compared with state-of-the-art approaches.

The authors in (Hammoud et al., 2020) have presented the cloud federation formation using genetic algorithm to maximize the total payoff of federations. This paper extends the genetic algorithm as an evolutionary game to boost the payoff whereas balancing the stability between federations. In (Ray et al., 2018), a broker based architecture in which the cloud federation formation has been modeled as hedonic coalition game. The main goal is to get the most appropriate and stable federation of trusted CSPs which will increase the satisfaction level of every single ESP based on QoS and payoff. A cloud federation formation model has been developed by (Halabi et al., 2018) that considers the security risk stages of ESPs by measuring the security of ESPs as per too well determined assessment principles associated to security risk prevention and mitigation. The Cloud federation formation process has been modeled as a hedonic coalition game based on the security risk stages and CSPs's reputation. The authors proposed a cloud federation formation algorithm which allows CSPs to collaborate while considering that the security risks have been introduced to their infrastructures, and minimize the collaboration with unwanted CSPs.

The authors (Ray et al., 2018) have modeled the problem of establishing cloud federation between CSPs as hedonic coalition game using a useful function based on payoff and migration cost. The main objective is to maximize the former and minimize the latter. They have proposed an algorithm to resolve the hedonic game and developed a comparison of its performance with existing game theory based federation formation models. They (Hassan et al., 2016) have proposed the mechanism of cloud federation formation using a trust-based cooperative game theory which allows CSPs to convincingly develop a federation based on the maximization of payoff and minimization of penalty in a result of selecting the responsible CSPs. The framework of cloud federation formation (Ray et al., 2019) has been established as a multi-objective optimization problem using the trade-off among payoff and QoS. The proposed algorithm attempts to increase payoff of the federation whereas balancing stability among the payoff and QoS of the participants of federation. They have also used Linear Scalarization besides ε-constraint technique to search the Pareto optimal result. They have proposed a heuristic algorithm for federation formation after the integer linear program.

The problem (Hassan et al., 2015) has been addressed of efficient federation formation mechanism between media CSPs that encourages them to collaborate to manage the demand of dynamic resources of customers for streaming multimedia workloads. They have designed a cloud federation formation on the basis of cooperative game theory. By cooperating to the responsible providers, their proposed mechanism focuses on forming federations for minimization of the penalties caused by violation of service quality by unreliable providers and maximize the payoff. The authors (Dhole et al., 2016) have proposed a game theoretic mechanism for cloud federation formation based on trust among CSPs. This approach will permit the ESPs to make decision individually on the basis of their own decisions relying on the payoff included into the cloud federation. Their mechanism considers the trust among the CSPs that are participating in the federation formation technique.

Due to the heavy traffic on cloud computing more connections have interrupted and downtime takes more time between client and ESPs. The probability of unsecure services has been increased in recent years.

Table 1 Consolidated Literature review

Ref.	Capital	QOS	Trust & Fairness	Reliability	Methodology
(Qammar et al., 2024)	**C**	**E/S**	×		Blockchain
(Hammoud et al., 2020)	**P**	**S**	×	×	GA & Evolutionary game theory
(B. Ray et al., 2018)	**P**	**A**	**T**	×	Hedonic Coalition Game
(Halabi et al., 2018)	×	**A/R**	×	×	Cooperative Game
(B. K. Ray et al., 2018)	**P/C**	×	×	×	Hedonic Coalition Game
(Hassan et al., 2016)	**P**	**S/E**	×	×	Hedonic Coalition Game
(B. K. Ray et al., 2019)	**P**	**A/R/S**	×	×	Linear Scalarization, ε-constraint method & The integer linear program
(Hassan et al., 2015)	**P**	**E**	×	×	Hedonic Coalition Game
(Dhole et al., 2016)	**P**	**S**	**T/F**	×	Game Theoretic Approach

Legend: **P**= Payoff **C**=Cost **A**=Availability **E**= Efficiency **S**= Stability **R**= Response Time **Re**= Reliability

A comparison has been performed of the research works and demonstrated in Table 1. The paper (Hammoud et al., 2020) has focused on payoff and stability and the authors (Ray et al., 2018; Ray et al., 2019) have developed the cloud federation on the basis of availability, payoff and cost respectively. This paper (Halabi et al., 2018) has used the parameters of availability and response time but they (Hassan et al., 2016) have focused on payoff, stability and fairness of payoff distribution among the CSPs in cloud federation formation.

Table 1 clearly show that reliability parameter have not used by the researchers in developing the cloud and edge computing. Due the excessive use of edge federations the reliability becomes very import part of the cloud federations without reliability the data of the user can be mishandled, the connections interruption and downtime can be increased. Individual records cannot be mishandled by edge service providers without their approval, and actions should be taken to keep secure identity of a user, while the user activities should be hidden and untraceable. Negligent entities having individual's data might decide to outsource them to an external institution for getting personal intents, actions, or interests to extend their commercial market. Moreover, Foes can extract personal credentials from unreliable system to interrupt their confidentiality. These actions have known as criminal practices, and novel rules are dedicated on mitigating these incidents (Ren et al., 2019).

In this paper, the reliable ESPs will be selected first and then on the basis of selected ESPs edge federation will be formed based on their reliability level (how reliable the federation is). Thus, this will help the edge service providers to maintain reliable connection and try to avoid or reduce the downtime. It also helps the ESPs to provide secure services to the clients to perform their desired tasks.

PROPOSED SOLUTION

In this section, initially the architecture of edge federation and reliability parameters has been elaborated. After that the methodology has been presented to form edge federation for reliability using the proposed ENN algorithm which has been described briefly.

System Architecture

The proposed architecture consists of user, edge orchestrator, edge federation and cloud which has been shown in figure 2. Edge Orchestrator is the main part of the system for calculating the reliability of ESPs. It evaluates the schedule

for traffic redistribution based on the obtained traffic pattern and users' spatiotemporal data. It calculates the reliability of the ESPs using the ENN algorithm based on user request and verify the results from cloud because edge orchestrator has been associated with cloud. While edge infrastructure providers (EIPs) deploy the demanded services at the edge based on the schedule.

Edge has different kind of ESPs who have resources and provides different services to the end user. ESPs have different attributes i.e. service resources availability, service resources reliability and service resource security. The attributes have different parameters i.e. CPU frequency, Memory Size and Storage size parameters fall under the umbrella of service resource availability attributes. Service resource security includes authentication type, authorization type and self-security competence parameters etc. The parameters for service resource reliability are average CPU utilization, average RAM utilization, average Storage utilization, average network Bandwidth utilization and many more. There is an immense pool of ESPs which are reliable and unreliable ESPs for federation formation for different requirement of users.

End users are the consumers of the services which have been provided by the ESPs. End users request for the services to edge orchestrator then the service providers are assigned by the edge orchestrator. End user consumes the services of ESPs having different resources.

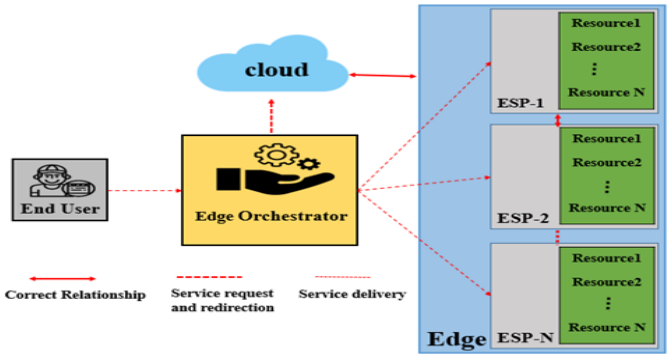

Figure 2 Reliable edge federation Architecture

For better understanding, a comprehensive example of service redirection has been described in figure 3. This mechanism is different from the traditional mechanisms. (1) The end user requests for the connection to edge orchestrator at a specific area. Edge orchestrator also gets the attributes of demanded ESPs from user. (2) Edge orchestrator applies the ENN algorithm to find out the demanded ESPs in the immense pool of edge. (3) ENN gets the demanded ESPs from edge. (4) When requested ESPs have been found according to the given attributes then all resultant ESPs have been passed to cloud for verification. (5) Once selected ESPs verified from the cloud (6) then the edge federation of that ESPs have been formed by edge orchestrator and provided to end user as shown in Figure 3, there are two different federations have been formed on the basis of two different conditions. Thus, the edge federation is formed based on reliability due to which a high reliability can be achieved.

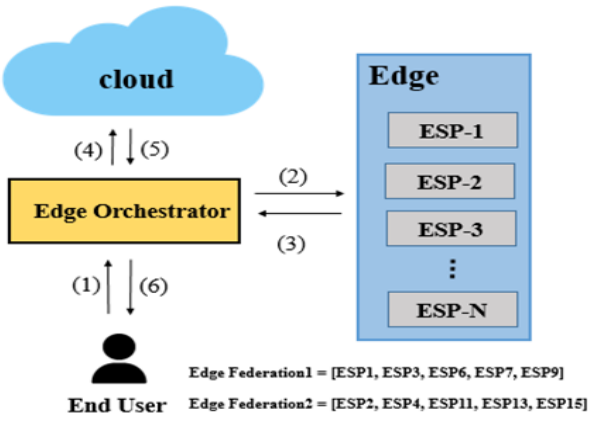

Figure 3 Service redirection of reliable edge federation

Reliability Parameters

Evaluating the reliability of edge computing is a difficult job due to its complexity and flexibility. Reliability measurement based on a certain time frame. There are four indirect operators, which are the key indicators of reliability. The parameters which have been used to enhance the reliability of edge federation are: average Network Bandwidth, average CPU utilization, average Memory utilization and average Storage utilization.

Table 1: Reliability parameters

Symbol	Explanation
A_1	Average Network bandwidth
A_2	Average CPU utilization
A_3	Average Memory utilization
A_4	Average Storage utilization

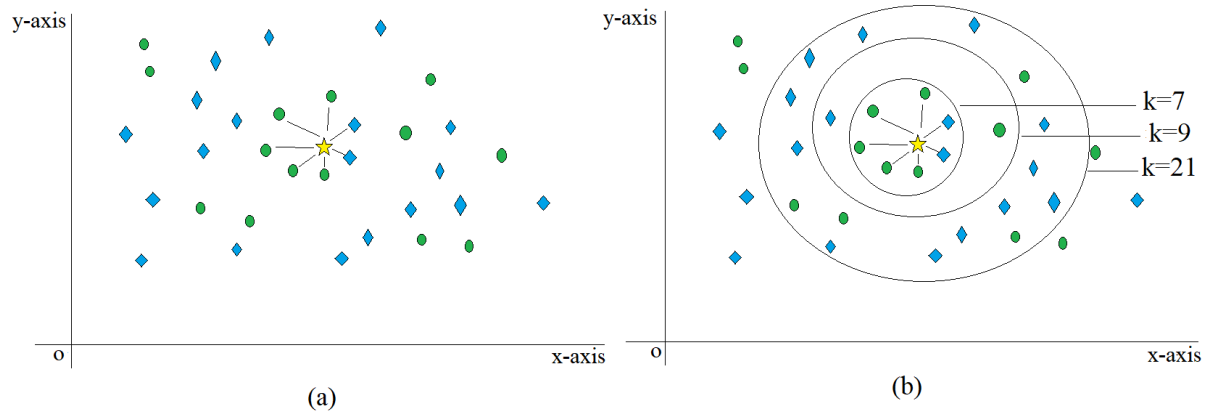

Figure 4(a) Selecting Neighbors in kNN (b) Neighbors Selected according to k value

The values are based on the statistical outcomes inside a certain time frame Δ*t*. For example, for a resource performing *n* computing tasks within time frame Δ*t*, then

$$A_1(\Delta t) = \sum_{j=1}^{n} M(j)/n \quad (1),$$

$$A_2(\Delta t) = \sum_{j=1}^{n} N(j)/n \quad (2)$$

$$A_3(\Delta t) = \sum_{j=1}^{n} O(j)/n \quad (3),$$

$$A_4(\Delta t) = \sum_{j=1}^{n} P(j)/n \quad (4)$$

Where *M (j*) represents the j-th measured value of Network Bandwidth, *N (j)* represents the j-th measured value of CPU usage rate, *O (j)* represents the j-th measured value of Memory utilization rate, and *P (j*) represents the j-th measured value of Storage utilization rate.

The connections between user and edge has been categorized as short-term connection, medium term connection, and long term connection according to time. Whenever the user wants to connect for a short period to a reliable ESP because the user might be a mobile device and does not stay for a long time. In this scenario, the data of the last 5 minutes interval will be considered because the user has requested for short time connection to perform the activities. If a user desires to connect for a medium period to a reliable ESP because the user wants to spend some time there. Then last 30 minutes interval's data will be taken into consideration for calculation because the user has requested for medium-time connection to perform the activities. When a user requests to connect for a long period to a reliable ESP because the user is staying for long time. Then last 60 minutes interval's data will be selected for calculation because the user has demanded for long-time connection to perform the activities.

Edge nearest Neighbor (ENN) Algorithm

The proposed ENN algorithm extends the traditional kNN approach by incorporating reliability parameters and user-defined conditions, enabling the formation of trustworthy edge federations for smart agriculture.

The kNN algorithm compiles all of the data into a database and uses k nearest neighbors to classify the unknown sample. Figure 8 shows how an unknown sample is categorized by the majority vote of nearest neighbors. KNN only considers the distance of the unknown sample while computing K; it ignores the distance between samples. Once a sample is allocated to a class, it is impossible to draw any conclusions about the samples' connection or membership. (Learning, 2020).

The choice of *k* in the *k* closest neighbor classification has an important impact in the categorization of unknown samples as shown in figure 4. After finding the nearest neighbors, the nearest neighbors have been labelled according the nearest neighbors majority. Value of *k* affects the classification of unknown sample because if *k* is 7 & 9 then the unknown sample belongs to green category. If *k* value has been set high to 21 then unknown sample belongs to blues because number of blue neighbors have been increased. When *k* is 21 then number of blue class are higher than number of green class depicting that value of *k* affects the performance of algorithm.

KNN is used to classify an unknown sample and assign it to a class based on its nearest neighbor. However, our proposed research does not work on labeling the unknown sample. Hence extending the concept of KNN with the proposed method named Edge Nearest Neighbors (ENN). In this proposed ENN, the edge nearest neighbor works on the basis of the requirements of the user. ENN provides the best solution to the user according to the user's requirements and conditions. The proposed ENN provides the best reliability on user's demand because it works on the request of the user which has been given in the form of a vector having attributes of average CPU utilization, average RAM utilization, average Storage utilization, and average Network Bandwidth. When the requirement vector have been given to ENN, the first process has been activated which is to determine the vector's nearest neighbor with the value of E which represents the number of nearest neighbor edges.

For example, the user requests for a reliable edge federation by given requirements in the form of vector $X=[x_1, x_2, x_3, x_4]$ which is also called a test vector and E is 15. On this given vector X, the nearest neighbors have been found of vector X from the point Y= [y_1, y_2, y_3, y_4] by calculating the distance of point X and Y from Equation 5.

$$D_1(i_m, j_m) = \sqrt{\sum_{m=1}^{n} (i_m - j_m)^2} \quad (5)$$

After applying the above equation 5, the distance have been generated for all edge providers from the immense pool of edges against the target vector X. Every edge have been sorted on the basis of distance from X to the edge in ascending order. After sorting the every edge, the E edges have been picked from the sorted edges as shown in figure 5.

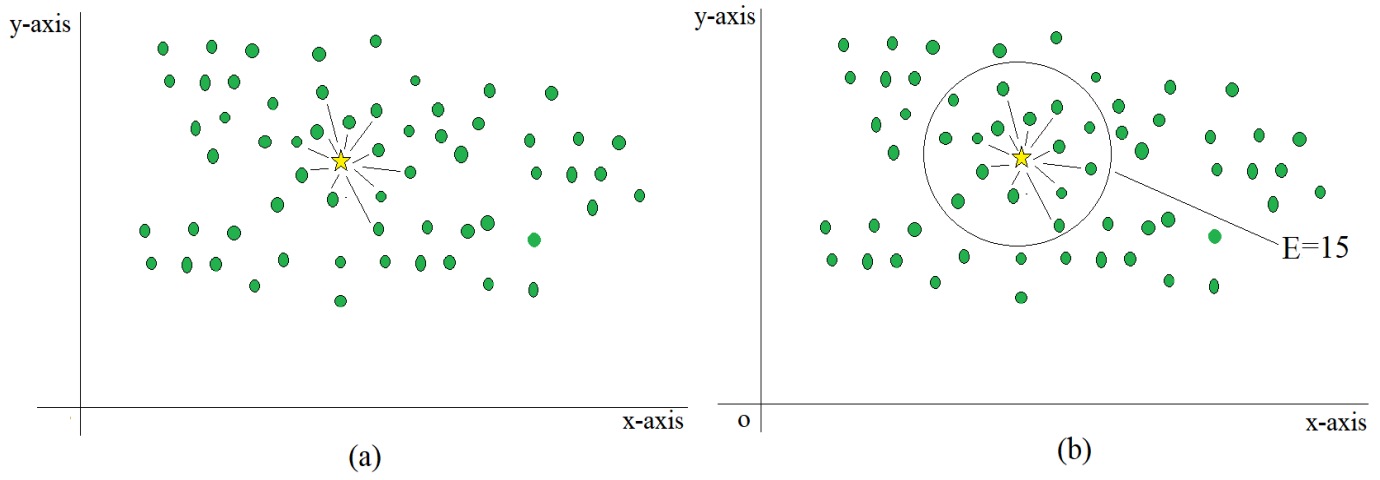

Figure 5 (a) Selecting Neighbors in ENN (b) Neighbors Selected according to E value resembling the kNN

In KNN, it assign the class or label to unknown sample on basis of neighbors but in ENN, the problem of reliability based edge federation formation has been resolved on the basis of user requirements. According to user requirements, the different problem scenarios can take place i.e. user can ask to find the nearest neighbors of test vector which have higher values (edges having high values than target vector X), lower values than test vector (edges having low values than target vector X) or values equal to test vector (edges values are equal to target vector X) as shown in figure 6. User can ask for such request to maintain the threshold.

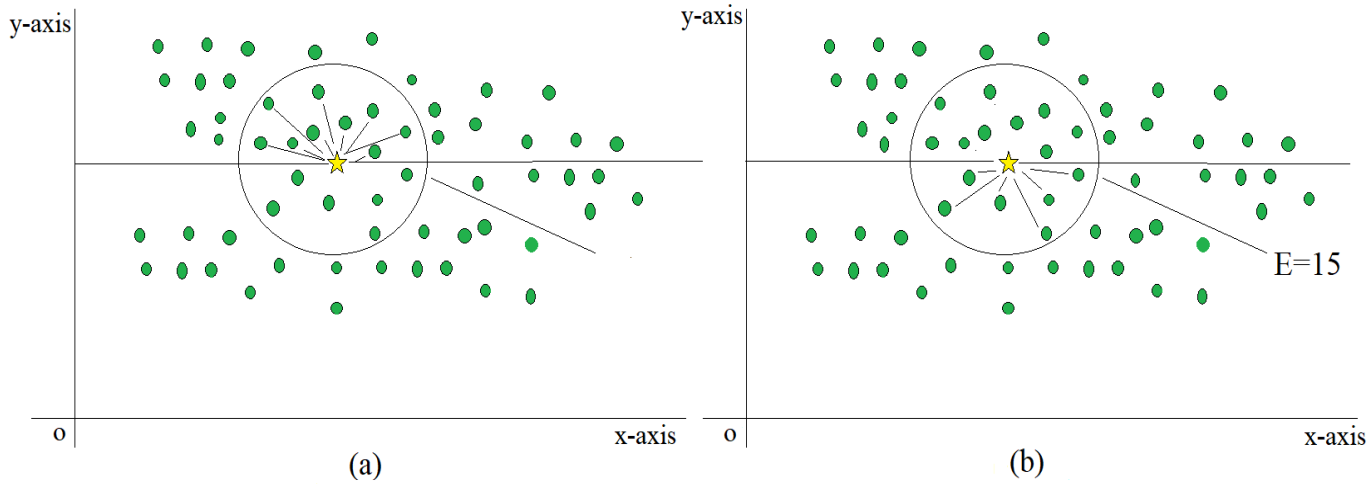

Figure 6: Selection with ENN for (a) ESPs of higher values than target (b) ESPs of lower values than target

The nearest neighbors have been found from the target value are unable to meet the user requirement of getting only those neighbor in which ESPs have higher values or lower values than the target vector. In order to accomplish the required objective, another distance formula has been applied on the sorted ESPs to get the specific ESPs which are above, below or equal to the target point.

$$D_2(i_m, j_m) = \sqrt[3]{\sum_{m=1}^{E} (j_m - i_m)^3} \quad (6)$$

By applying Equation 6, the results have been generated which can be positive values, negative values, and 0. Where positive values of neighbors are higher, negative values of neighbors are lower than the target vector but 0 values of neighbors show that they are equal to the target vector. So, when the user request for the scenario of higher values that the user needs the edges having greater values than the target vector then the positive values are considered only and negative and 0s values have been neglected. For the Scenario of higher values when the user requests that the user needs the edges having lower values than the target vector then the only negative values are considered and positive and 0s values have been neglected.

There is another scenario which can be occurred that user might request for the edges which has equal values to target vector. In this scenario, the neighbors having 0 values will be considered and other all values will be neglected to meet the user requirement. ENN meet the user requirement of any type as shown in algorithm.

First of all, the dataset of ESPs have been initialized along with the user requirement target vector X. After initialization of data and target vector then the distance D1 have been calculated between target vector and each ESP. In step 3, the all calculated distances have been sorted in ascending order to get the nearest neighbors of target vector. After sorting distances, value of E has been initialized and **E**-ESPs corresponding to these **E**-distances have taken for further calculations. In step 7, D_2 distances have been calculated for sorted ESPs from target vector. After applying second distance formula, the values have been obtained which can be positive, negative or 0 so take the values of ESPs according to user requirement.

ENN Algorithm

1) Initialize ESPs, target vector X_i
2) Calculate $D_1(i_m,j_m)=\sqrt{\sum_{m=1}^{n}(i_m-j_m)^2}$ m=1, 2...n; *where* D_1 is distance between the ESPs and X which have +ve and 0 output values.
3) Sort the calculated n distances between X and ESPs in ascending order.
4) Initialize value of E and E should be a +ve integer. *Find* those E-ESPs corresponding to these E distances from this sorted list.
5) Apply $D_2(i_m,j_m)=\sqrt[3]{\sum_{m=1}^{E}(j_m-i_m)^3}$ on sorted E-ESPs. *Where* D_2 another distance formula which have +ve,-ve and 0 output values.
6) Return the positive, negative or 0 values as per requirement

To understand the working of ENN, there is an example that user gives the requirement in the form of vector X= [78, 43, 48, 12]. User also demands for 6 edges which near to vector which means the value of E will be 6. The total number of ESPs in the pool are 10 which has been picked for the implementation of the ENN. The value of ESPs are shown in the Table 2.

Table 2 Values of ESPs

Node ID	CPU	RAM	DISK	NETWORK
1	80	45	50	10
2	76	41	46	14
3	81	46	51	11
4	82	47	51	12
5	75	41	45	9
6	78	43	48	12
7	76	39	40	10
8	78	43	48	12
9	90	45	48	11
10	75	45	41	13

ENN gets the target vector from the user and then finds the nearest neighbors of target vector by applying distance equation1. The distance has been obtained by applying equation1 on the given ESPs as mentioned in Table 3.

Table 3 D1 distance Applied

Node ID	CPU	RAM	DISK	NETWORK	D1
1	80	45	50	10	4
2	76	41	46	14	4
3	81	46	51	11	5.29
4	82	47	51	12	6.40
5	75	41	45	9	5.57
6	78	43	48	12	0
7	76	39	40	10	9.38
8	78	43	48	12	0
9	90	45	48	11	12.21
10	75	45	41	13	7.94

The calculated values has been sorted in ascending order to get 6 nearest neighbors of target vector. After sorting the distance1, the first 6 ESPs have been selected for further calculations because the value of E is 6 which has been shown in Table 4.

Table 4 sorted top 6 ESPs

Node ID	CPU	RAM	DISK	NETWORK	D_1
6	78	43	48	12	0
8	78	43	48	12	0
1	80	45	50	10	4
2	76	41	46	14	4
3	81	46	51	11	5.29
5	75	41	45	9	5.57

User can request for the edges from 6 nearest edges which are above, below or equals to the target value to maintain the threshold. So, the distance formula D_2 has been implemented on the 6 edges to get the edges above, below and equals to the target vector. The distance2 calculation has been perform on the sorted ESPs values of distance1 as shown in Table 5.

Table 5 D2 applied on sorted 6 ESPs

Node ID	CPU	RAM	DISK	NETWORK	D_1	D_2
6	78	43	48	12	0	0
8	78	43	48	12	0	0
1	80	45	50	10	4	2.52
2	76	41	46	14	4	-2.52
3	81	46	51	11	5.29	4.31
5	75	41	45	9	5.57	-4.47

After calculating the distance by apply distance2 on sorted ESPs and the obtained values are in the form of 0s, positive and negative numbers. To satisfy the request of user to take above ESPs values than target vector, ESPs having the positive values will be considered. In this example, the ESP1 & ESP3 will be reserved for user condition of getting above ESPs values than the target vector because both have the positive value. ESPs having the negative values will be considered to satisfy the need of user of having below ESPs values than target vector. In this example, the ESP2 & ESP5 will be reserved for user wish of getting below ESPs values than the target vector because both have the negative value. When user asks for equal values to target vector to maintain the threshold then the ESPs having 0 values will be considered. Here, ESP6 and ESP8 will be selected because both ESPs values are equals to 0.

IMPLEMENTATION

The proposed algorithm has been implemented on the dataset of ESPs to enhance the reliability of the edge federation. The dataset of edge computing has been taken from the kaggle repository. The dataset consists of the 1000 edge provider's resources usage in which every ESPs contains a series of data points of resource usage 0 to 300 minutes. The data includes the details of data points which are average CPU utilization, average RAM utilization, average Storage utilization, and average Network Bandwidth. There are different types of connections according to time and categorized as short-term connection, medium term connection, and long term connection. Whenever the user wants to connect for a short period to a reliable ESP because the user might be a mobile device. Then the data of the last 5 minutes interval will be considered because the user has requested for short time connection to perform the activities. So, the ENN has been applied to the data of the last 5 minutes interval and offers reliable ESPs according to user need. If a user desires to connect for a medium period to a reliable ESP because the user wants to spend some time there. Then last 30 minutes interval's data will be taken into consideration for calculation because the user has requested for medium-time connection to perform the activities. So, the ENN has been applied to the data of the last 30 minutes interval and provide reliable ESPs according to user conditions. When a user requests to connect for a long period to a reliable ESP because the user is staying for long time. Then last 60 minutes interval's data will be selected for calculation because the user has demanded for long-time connection to perform the activities. The ENN has been applied to the data of the last 30 minutes interval and comes up with reliable ESPs according to user request.

RESULTS

ENN has been implemented to form the edge federation on the above mentioned dataset of ESPs. Top resultant ESPs, who have performed exceptionally, have been selected for federation formation of ENN. The most reliable ESPs have been gained by giving a target vector which has been a user requirement. The given target vector is Y = [70, 48, 58, 12] which has been given to data. After applying ENN, the different number federations have been formed i.e. federation of 3 ESPs, 5 ESPs, 7 ESPs and upto user's requirement. To get 3, 5, 7 ESPs federation, the value of E will be changed because value of E will be 3 for federation of 3 ESPs and value of E will be 5 for federation of 5 ESPs respectively. The value of E also affects the performance of the model because E represents the total number of neighbor ESPs. These federations of ESPs have shown in Table 6. To get the federation of 3 ESPs, the first three ESPs has been selected and to form federation, the top 5 performer 5 ESPs has been picked to form the federation of 5 ESPs. The all 7 ESPs has been select for the sake of a federation formation of 7 ESPs. All these single federations of 3 ESPs, 5 ESPs and 7 ESPs have been formed by giving the target vector Y.

Table 6 Values of 7 ESPs based on target vector X

ESP ID	CPU	RAM	DISK	NETWORK	D_1	D_2
861	69	51	57	11	3.46	2.88
799	69	51	56	12	3.74	2.62
125	67	49	59	10	3.87	-3.21
130	68	51	56	10	4.58	1.44
692	70	52	55	12	5	3.33
575	71	51	54	10	5.48	-3.53
126	70	52	54	11	5.75	-1
Target Vector Y = [70, 48, 58, 12]						

ENN has been also implemented using different multiple requirement conditions of user to form different number of federations. Different target vectors Y = [70, 48, 58, 12], H= [66, 52, 50, 9] and G= [75, 55, 65, 8] have been taken as a user requirement. For target vector Y, the values has been already shown in the table. The Nearest ESPs of target vector H has been displayed below in the Table 8.

Table 7 values of 7 ESPs based on target vector X

ESP ID	CPU	RAM	DISK	NETWORK	D_1	D_2
653	68	49	50	10	3.74	-2.62
550	67	50	51	13	4.69	3.87
694	70	49	50	7	5.39	3.07
578	69	48	51	11	5.48	-3.04
696	71	51	48	8	5.57	4.86
133	68	47	49	11	5.83	-4.79
695	70	52	54	11	6	5.14
Target Vector H= [66, 52, 50, 9]						

The target vector H's nearest ESPs are demonstrated in Table 9. These ESPs of each condition form federations. User asks for ESPs having greater values, equal, and lower values than target vectors then ESPs with the positive values, 0s and negative values will form federation respectively. In target vector H's table, the ESP 550, 694, 696 & 695 will be assigned when user demand for ESPs higher values than target vector otherwise for lower values than target vector, the ESP 653, 578 & 133 will be assigned.

Table 8 values of 7 ESPs based on target vector X

ESP ID	CPU	RAM	DISK	NETWORK	D_1	D_2
560	75	57	64	10	3	2.47
690	71	56	64	8	4.24	-4

652	72	57	63	9	4.24	-2.96
807	72	58	65	10	4.69	2
579	72	58	66	10	4.80	2.08
132	75	59	62	8	5	3.33
135	74	59	62	8	5.10	3.30
Target Vector G= [75, 55, 65, 8]						

ENN has been implemented along with sole service parameter in which ENN shows better results than sole parameter shown in Figure 7 (a) & (b). Average CPU utilization, average RAM utilization, average Storage utilization and average Network Bandwidth all of them have shown major difference between their outcome upper and lower values than the target value of cpu, ram, disk and network of vector Y respectively as displayed in figure 7.

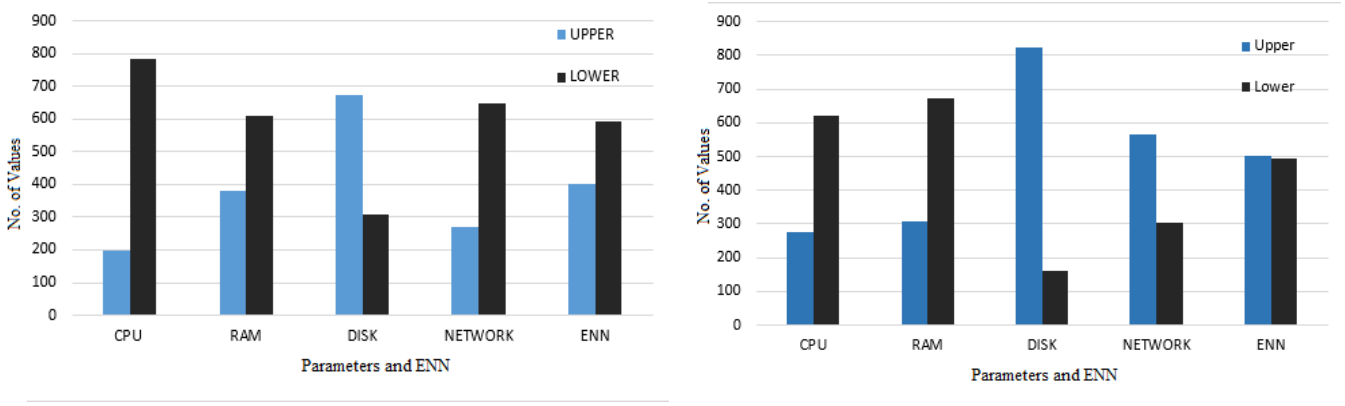

Figure 7. (a) comparison of target vector Y and their sole parameters (b) comparison of target vector H and their sole parameters

Every sole parameter has displayed different values because every sole threshold values is different and the condition of getting reliable ESPs is single. For example, if the value of average CPU utilization from vector Y is considered as a sole condition to get reliable ESPs. When it has been applied the values will be generated only on the basis of average CPU utilization parameter. If the CPU value is 70 then the all ESPs will be judged as high or low values accordingly. But other features values will be neglected which may also cause an uncertain situation for the federation formation process. Due to this particular reason many bad and untrustworthy ESPs will be selected for federation formation which has been not considered good.

For example, may be an ESP have been selected due high CPU value but it has low values of ram, network bandwidth and storage that may also leads to untrustworthy situation. ENN has taken all parameters into consideration so due to all parameters, the condition of getting reliability of ESPs on the basis of all parameters becomes strict. If an ESP has been selected as reliable then it will meet the threshold values of all parameters and if one of all parameter's threshold value is missed then the ESP will not be declared as reliable. Due to this strict mechanism, it provides much better results than all sole parameters. ENN has outperformed the sole parameter threshold value of target vector Y.

When the ENN and sole parameters have been implemented again using the values of different target vector. The value of target vector H has been used and the results are shown in Figure 7 (b). Average CPU utilization, average RAM utilization, average Storage utilization and average Network Bandwidth showed high variance between their lower and upper values from the target. ENN outperformed all sole parameters here again by showing low variance as compared to sole parameters which shows the ENN accuracy.

ENN has formed federation of ESPs but problem is that whether the selected ESPs stay reliable for next period of time. As mentioned earlier, there are total three types of user can be faced due to which the data has been taken of average of last 5 mins, 30 mins and 60 mins. So, to make sure that ESPs whether maintain their reliability or display drop down in their reliability values, the further average of 5 mins, 30 mins and 60 mins checked of ESP 1 to 7 upto different times of interval. We have observed that all selected ESPs have maintained their values of parameters and threshold values throughout several intervals.

For average 5 min values of 7 ESPs has been tested on data to 35 mins which means 6 more chunks of average last 5 mins. The all seven ESPs have maintained their values upto 35 mins. The ESP1 have maintained their values which have been also shown in Figure (a). Each line in the graph represents an attribute of the ESP. The average value of each ESP's attribute values have also been displayed in graph which named as threshold.

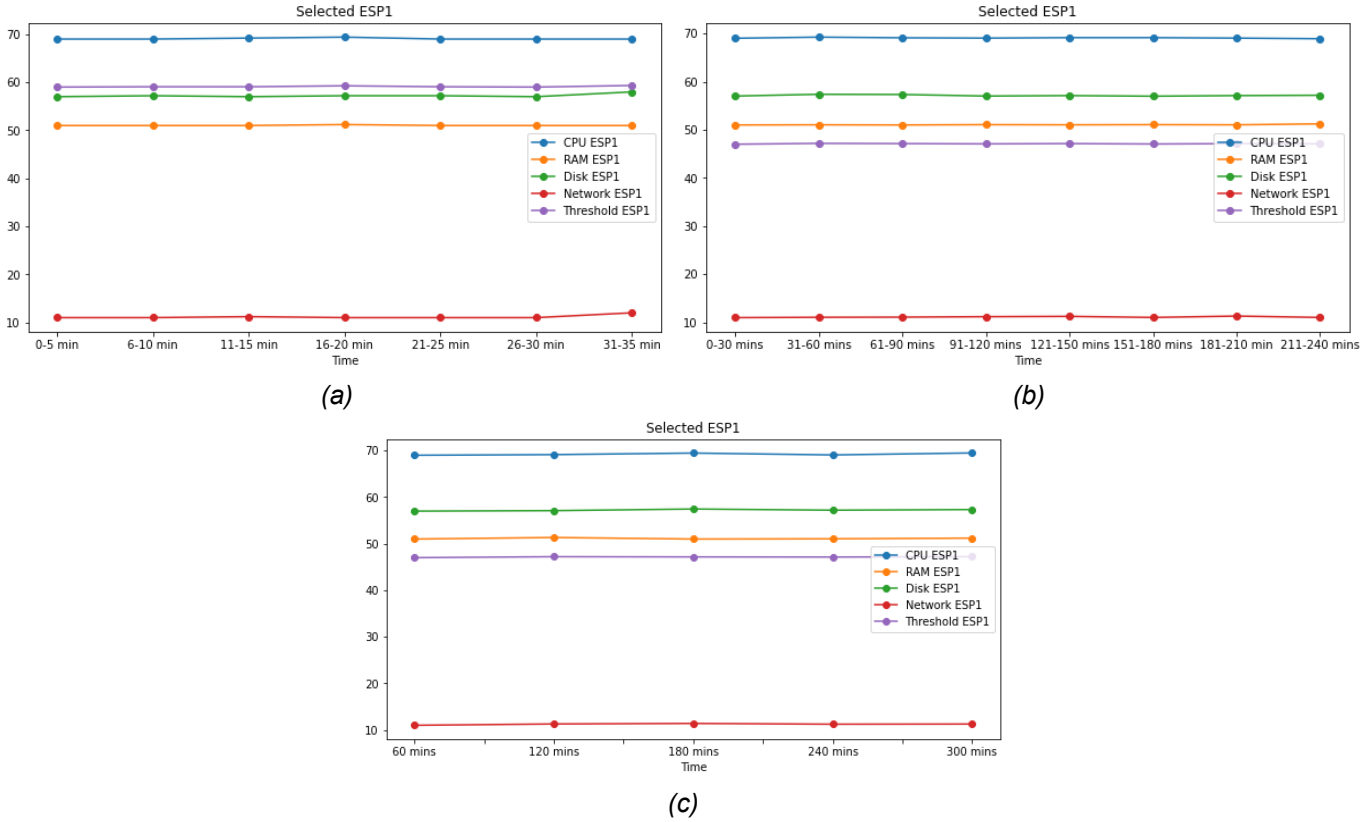

Figure 8 (a) ESP maintained value for average 5 mins (b) ESP maintained value for average 30 mins (d) ESP maintained value for average 60 mins

Seven ESPs has been verified on data to 240 mins which means 8 chunks of average 30 mins to check whether ESPs maintain their values or not. These seven ESPs have shown excellent results by maintaining their values in 8 chunks upto 240 mins. The ESP1 maintained values have been also shown in Figure 8 (b). The values for these seven ESPs have been taken from data upto 300 mins and divided into five chunks to take average 60 mins value of data to check whether ESPs maintain their values or not. Each ESP from the selected seven ESPs has exhibited better reliability by maintaining their values in five chunks of data which have made of 300 mins. The ESP1 maintained values have been also shown in Figure 8 (c).

All ESPs of the formed federation of seven ESPs of average 5 mins, average 30 mins and average 60 mins have maintained their values but when the federation formed of high number of ESPs then some ESPs showed little drop down in their parameter values but they have maintained threshold values. Some of them also drop their parameter values and also threshold values. The two ESPs have not maintained their threshold and parameter values after three intervals of time in average 30 mins and average 60 mins when federation of 101 ESPs have been formed. The 3 ESPs have maintained their threshold value but the value of parameters changes after 4, 3 and 3 intervals of time of average 5 mins, average 30 mins and average 60 mins respectively. The 3 ESPs have maintained their threshold so the conclusion can be made about them that they have maintained their threshold reliability.

The proposed ENN algorithm has successfully formed an edge federation of ESPs based on reliability because the selected ESPs maintain their threshold values. It maintained efficient execution time even as the number of ESPs increased, ensuring timely federation formation at scale. To validate this, different federations have been formed on different conditions which has been displayed in Figure 9 (a), (b) & (c). Federations ranging from 3 to 101 ESPs were formed, and while execution time increased with federation size, the growth was minimal, demonstrating the scalability and efficiency of ENN. As observed, the time taken for federation of 3 ESPs is 0.651 and for federation of 101 ESPs

is 0.753. So, formation time of federation of 101 ESPs and 3 ESPs has minor difference which is 0.102. The execution time of federation formation of different ESPs has been also displayed in the Figure 9 (a) in the line displays the execution time which has been gradually increasing due to increase in the amount of ESPs taking part in federation formation process. The y-axis displays the values of the execution time and values of the ESPs which has been taking part in forming the federation has been displayed on the x-axis.

The two federations have been formed of different ESPs for two different conditions target vector Y and H i.e. 2 federations of 3 ESPs, 2 federations of 5 ESPs, 2 federations of 7 ESPs and so on up to federation of 101 ESPs. Line in graph in Figure 9 (b) displays the execution time which is increasing due to increase in number of ESPs in formation process of 2 federations. The execution time and the number of ESPs in forming federation are directly proportional. ENN has also formed 2 federations of 3 ESPs on the basis of two different target vectors in which taken time is 1.152 and for 101 ESPs is 1.389. The difference between time of 3 ESPs and 101 ESPs is 0.237 which is less difference time if the difference of 3 ESPs to 101 ESPs has taken into consideration in formation of 2 federations. The difference of time of 2 federations and single federation of 3 ESPs is 0.501 and the difference of time of 2 federations and single federation of 101 ESPs is 0.636 which means ENN has taken less time to form 2 federations at the same time as compared to single federation.

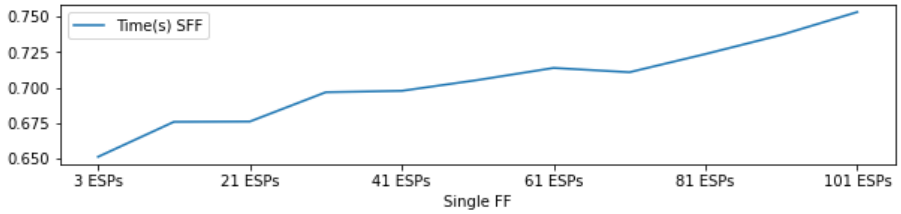

Figure 9 (a) time taken for single federation formation

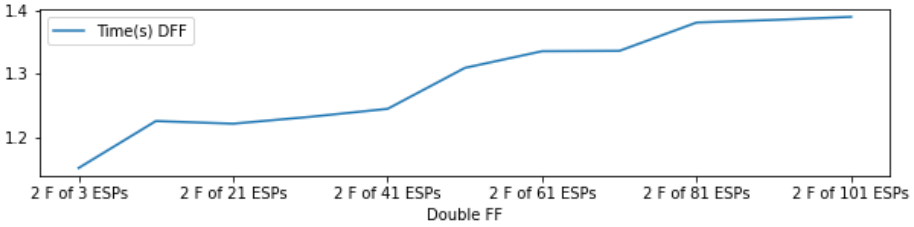

Figure 9 (b) time taken for 2 federations formation at same time

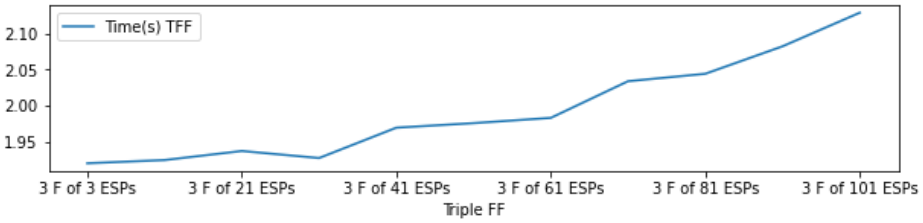

Figure 9 (c) time taken for 3 federation formation at same time

In Figure 9 (c), the 3 federations haves been formed of different ESPs for three different conditions target vector Y, H and G i.e. 3 federations of 3 ESPs, 3 federations of 5 ESPs, 3 federations of 7 ESPs and so on up to federation of 101 ESPs. Line in graph display the execution which is increasing due to increase in number of ESPs in formation of 3 federations. The execution time and the number of ESPs in forming federation are directly proportional. ENN has formed 3 federations of 3 ESPs on the basis of three different target vectors in which taken time is 1.920 and for 101 ESPs is 2.109.

The difference between time of 3 ESPs and 101 ESPs is 0.189 which is less difference time if the difference of 3 ESPs to 101 ESPs has taken into consideration in formation of 3 federations. The time taken for 3 federations is 1.920 which is not three times of the time taken for formation of single federation of 3 ESPs. In forming 3 federations of 101 ESPs has taken also less time than three times of single federation formation of 101 ESPs which means ENN has taken less time to form 3 federations at the same time as compared to single federation. The ENN has been performed much

better in term of time when the number of target vectors increased. The results show that ENN maintains efficiency as federation size grows, suggesting strong potential for scalability to thousands of ESPs in real-world agricultural environments.

CONCLUSIONS

Edge computing presents a vital computing paradigm for smart agriculture, where the reliability of edge services is critical. However, this reliability can be compromised if insecure and untrustworthy edge service providers (ESPs) are selected, potentially damaging the reputation of both the federation and the ESPs. This work addresses the problem of forming an edge federation that enables trustworthy providers to deliver services to clients—such as autonomous tractors, irrigation systems, and monitoring sensors—with minimal connection interruption or downtime. To tackle unreliable ESP selection during federation formation and enhance edge reliability, we propose the Edge Nearest Neighbor (ENN) algorithm. Experiments were conducted using a dataset of ESPs characterized by various attributes. During federation formation, four key parameters were considered: average CPU utilization, average RAM utilization, average storage utilization, and average network bandwidth. The proposed ENN technique successfully resolved the challenge of forming a reliable federation, efficiently creating federations based on different user-defined target vectors. The results demonstrate that ENN performs exceptionally well, forming federations rapidly and responsively, even under varying temporal demands, indicating its suitability for federations involving thousands of ESPs in agricultural settings.

For future work, additional factors such as cost, profit, and quality of service (QoS) can be integrated into the federation formation process. In agriculture, cost is a critical concern, as farmers and enterprises often operate with limited budgets and require affordable yet reliable solutions. Balancing low cost with high reliability is therefore essential. At the same time, ESPs aim to maximize profit while sustaining QoS and reliability, making profit optimization an important consideration. Moreover, applications such as irrigation control, crop monitoring, and drone-based spraying demand strict QoS guarantees, highlighting the potential of QoS-aware federation formation as a promising research direction.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

AUTHOR CONTRIBUTIONS

All the authors actively participated in finalizing the manuscript. All authors reviewed and approved the final version of the manuscript for publication.

COMPETING OF INTEREST

No potential conflict of interest was reported by the authors.

REFERENCES

Choudhary, V., Guha, P., Pau, G., et al 2025. An overview of smart agriculture using internet of things (IoT) and web services. *Environ. Sustain. Indic.*, 100607.

Chun, B.G., Ihm, S., Maniatis, P., et al 2011. *Clonecloud: elastic execution between mobile device and cloud.* Paper presented at the Proceedings of the sixth conference on Computer systems.

Dhifaoui, S., Houaidia, C., Saidane, L.A. 2024. Computing paradigms for smart farming in the era of drones: A systematic review. *Ann. Telecommun., 79*(1), 35-59.

Dhole, A., Thomas, M.V., Chandrasekaran, K. 2016. *An efficient trust-based Game-Theoretic approach for cloud federation formation.* Paper presented at the 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS).

Dilley, J., Maggs, B., Parikh, J., et al 2002. Globally distributed content delivery. *6*(5), 50-58.

Dong, S., Tang, J., Abbas, K., et al 2024. Task offloading strategies for mobile edge computing: A survey. *Comput. Netw. 254*, 110791.

Halabi, T., Bellaiche, M., Abusitta, A. 2018. *A cooperative game for online cloud federation formation based on security risk assessment.* Paper presented at the 2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom).

Hammoud, A., Mourad, A., Otrok, H., et al 2020. Cloud federation formation using genetic and evolutionary game theoretical models. *104*, 92-104.

Hassan, M.M., Abdullah-Al-Wadud, M., Almogren, A., et al 2016. QoS and trust-aware coalition formation game in data-intensive cloud federations. *28*(10), 2889-2905.
Hassan, M.M., Alelaiwi, A., Alamri, A. 2015. *A dynamic and efficient coalition formation game in cloud federation for multimedia applications.* Paper presented at the Proceedings of the International Conference on Grid Computing and Applications.
He, Q., Zhao, H., Feng, Y., et al 2024. Edge computing-oriented smart agricultural supply chain mechanism with auction and fuzzy neural networks. *J. Cloud Comput. 13*(1), 66.
Kumar, V., Sharma, K. V., Kedam, N., et al 2024. A comprehensive review on smart and sustainable agriculture using IoT technologies. *Smart Agricultural Technology, 8*, 100487.
Learning, G. 2020. A Quick Introduction to KNN Algorithm. *Great Learning Organization.* from https://www.mygreatlearning.com/blog/knn-algorithm-introduction/
Özden, C., Karadoğan, N. 2024. Wheat yield prediction for Turkey using statistical machine learning and deep learning methods. *Pak. J. Agric. Sci., 61*(1).
Qammar, A., Naouri, A., Ding, J. et al 2024. Blockchain-based optimized edge node selection and privacy preserved framework for federated learning. *Cluster Computing, 27*(3), 3203-3218.
Ray, B., Saha, A., Khatua, S., et al 2018. Quality and profit assured trusted cloud federation formation: Game theory based approach.
Ray, B.K., Saha, A., Khatua, S., et al 2019. Toward maximization of profit and quality of cloud federation: solution to cloud federation formation problem. *75*(2), 885-929.
Ray, B.K., Saha, A., Roy, S.J.C.C. 2018. Migration cost and profit oriented cloud federation formation: hedonic coalition game based approach. *21*(4), 1981-1999.
Ren, J., Zhang, D., He, S., Zhang, Y., et al 2019. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *52*(6), 1-36.
Sharma, M., Tomar, A., Hazra, A. 2024. Edge computing for industry 5.0: Fundamental, applications, and research challenges. *IEEE Internet of Things Journal, 11*(11), 19070-19093.
Shi, W., Cao, J., Zhang, Q., et al 2016. Edge computing: Vision and challenges. *3*(5), 637-646.
Shi, W., Dustdar, S.J.C. 2016. The promise of edge computing. *49*(5), 78-81.
Umarani, C., Baskaran, K. 2024. An Explainable Deep Learning Model for Identification and Classification of Herbal Plant Species Based on Leaf Images. *Pak. J. Agric. Sci, 61*(3).