Hindawi
Complexity
Volume 2021, Article ID 6691571, 16 pages
https://doi.org/10.1155/2021/6691571

Research Article

Intelligent and Smart Irrigation System Using Edge Computing and IoT

M. Safdar Munir, Imran Sarwar Bajwa, Amna Ashraf, Waheed Anwar, and Rubina Rashid

Department of Computer Science, The Islamia University of Bahawalpur, Bahawalpur, Pakistan

Correspondence should be addressed to Imran Sarwar Bajwa; imran.sarwar@iub.edu.pk

Received 17 December 2020; Revised 8 February 2021; Accepted 18 February 2021; Published 28 February 2021

Academic Editor: Abd E.I.-Baset Hassanien

Smart parsimonious and economical ways of irrigation have build up to fulfill the sweet water requirements for the habitants of this world. In other words, water consumption should be frugal enough to save restricted sweet water resources. The major portion of water was wasted due to incompetent ways of irrigation. We utilized a smart approach professionally capable of using ontology to make 50% of the decision, and the other 50% of the decision relies on the sensor data values. The decision from the ontology and the sensor values collectively become the source of the final decision which is the result of a machine learning algorithm (KNN). Moreover, an edge server is introduced between the main IoT server and the GSM module. This method will not only avoid the overburden of the IoT server for data processing but also reduce the latency rate. This approach connects Internet of Things with a network of sensors to resourcefully trace all the data, analyze the data at the edge server, transfer only some particular data to the main IoT server to predict the watering requirements for a field of crops, and display the result by using an android application edge.

1. Introduction

Agriculture is the major resource of living wage in Pakistan. A smart, intelligent, and fully automated agricultural system was required and extremely desirable in some last decades when our population grew exponentially in comparison to the natural resources we have in our country, Pakistan. For this purpose, an IoT-based smart watering system has been achieved in the recent years of constant threat of losing water. This agricultural industry has two particulars. The plastic tunnel farming is divided into low, high, and walk-in tunnels. It is convenient to sow, spray, and harvest in the high tunnel than in low and walk-in tunnels due to its broader size. Traditional farming, on the contrary, is the most unpredictable and becomes the cause of more water wastage. The issue we are going to deal with in this paper regarding smart irrigation is any application designed and used for the smart watering system still needs to be more efficient and timely. Technically, it means that just cloud computing is not enough for a large-scale IoT application. There should be something like more efficient and fast application using a better architecture to handle different types of data coming from different sources (sensors). The main purpose of a quick and smart irrigation system is the consumption of water so frugally to execute the need of water more timely for a field of plants and to save inadequate sweet water reservoirs. To handle this rigorous matter, many sensor-based smart irrigation systems with their mobile applications have been designed in different times, but still, there is a question on their reliability when data grow and thus the latency rate of IoT devices. Like in preceding papers, the input parameters humidity, temperature, soil moisture, and light intensity were used, and a decision of watering plants or not was made on the basis of a fuzzy logic [1]. The same fuzzy logic has been applied to many healthcare systems, in which use of biosensors helped monitoring temperature, blood pressure, oxygen, and infection status of the wound [2]. Similarly, in fire alarming applications, this

technology helped a lot in 2018 [3] and 2019 [4]. Now, we come up with a new technology that is the combination of machine learning technique and semantics for some input parameters such as climate type, crop type, and soil type with the sensors' output: temperature, humidity, and soil moisture.

A smart irrigation system with the application of edge computing is required because the research studies on irrigation systems until now are not much efficient that they could not be implemented on large-scale systems and have less efficiency due to overburdenized sensors for all sensing data. So, a new intelligent and smart system should be designed.

Our research found some grounds due to which improvements in the existing system are mandatory:

(i) Existing smart irrigation systems either spotlighting on lesser parameters like soil moisture, air moisture/humidity or they are presenting a fuzzy logic (implemented in matlab) to produce an output decision or some are using simple machine learning algorithm to predict about water need for plants. A system which does not encounter the latency rate cannot provide the reliable solution.

(ii) Skipping important parameters such as soil strata and crop type can lead to an imperfect watering system for plants.

(iii) Unwanted data loading on the IoT server due to continuous throw of sensor data becomes a cause of less efficiency of the IoT server. An intelligent irrigation system should never halt due to overburden of data.

(iv) As newest expertise has come into sight due to progression in each and every field, therefore, we also have to change our classical method of irrigation to advanced, smart, and perfect and simple knowledge database for plant's data to powerful ontology-based semantics.

There are some main aspects, which we are going to concentrate on in our anticipated approach:

(i) Three sensors are used in our approach: a soil moisture sensor, humidity and temperature sensor, and light sensor. Furthermore, ontology is used for plant species data, different soil types, and different climate types.

(ii) This approach focuses on an intelligent technique, i.e., machine learning, to decide watering requirements for a particular plant, and by considering many other suitable parameters for the plant growth, i.e., climate, weather, and soil type, we are going to design a smart irrigation system in a different and more efficient way.

(iii) Our proposed smart system by design focuses on system reliability as if a sensor for some reason is not working at a particular time and was working an hour before, then the value it measured before an hour will be used by our trained model to produce the result because no drastic change can occur in other parameters in just an hour. It makes our system user friendly and more efficient.

(iv) The proposed approach is structured to come upon the problems of the obsolete irrigation method smartly.

2. Related Works

Traditional tunnel farms, all over the world, use drip irrigation or a sprinkler irrigation method. These are better than normal flooding methods. Various irrigation methods provide different water consumption levels and energy competence [5]. The surface irrigation and level irrigation methods provide low water and energy efficiency. The subirrigation, overhead irrigation, and sprinkler irrigation methods provide low-to-medium efficiency. The sprinkler and drip irrigation methods provide similar energy efficiency, but drip irrigation is more water efficient than sprinkler irrigation [6].

To increase crop production and decrease costs efficiently, the management of freshwater smartly is indispensable. The powerful use of technologies provides the precise amount of water required for plants. The SWAMP project [7] in Europe has developed an IoT-based smart water management platform for ideal irrigation with a proactive approach on four pilots in Brazil and Europe. The SWAMP architecture, the platform, and the system deployed presented by the European people include a performance analysis of FIWARE components. They aim to reengineer some of its components to provide greater scalability by using less number of computational assets.

The amount of land irrigated in the US is approximately the same as their farmers used to irrigate ten years earlier, but the important thing is water they are using nowadays for this purpose is quite less than previously used. They are growing plenty of fruits, vegetables, nuts, and whole grains that fulfill their inhabitant's requirements whole year. Two types of irrigation traditional technologies have been used in the US since 2013 [8]. First one is used in the gravity systems; it makes up 35 to 42% of irrigation systems in the United States. It delivers water from its source to a crop area by flooding through land-forming measures, including canals, waterways, basins, and furrows. Examples are the furrow system controlled flooding systems and uncontrolled flooding systems. The second type of irrigation technology is used by the pressure systems. In pressure systems, tubing or pipes are used to pump water, and irrigation is done through an applicator such as a sprinkler or perforated pipe.

China's development has been affected by three major issues regarding agriculture, landscape, and farmers [9]. The solution to these glitches is agricultural transformation. Though this transformation is not so easy and quick, introducing the cloud computing with Internet of Things to their agriculture is going to help them in solving the issue. However, cloud computing, IoT, and SOA technologies, are helping in the they have built huge data involved agricultural harvesting. Cloud computing is linked to IoT, and both

collectively can enhance the agricultural production to solve the matters regarding agriculture, landscape, and farmers.

In India, different traditional methods are designed and applied regionally in India over the past decades to cope up the necessities of their people in a sustainable way. The three irrigation methods that exist in India are diversion channels, small-scale water bodies such as tanks to store rainwater, and wells to collect groundwater. These methods are for small-scale as well as large-scale applications. As the population of India is increased enough, the needs on the water increase for various drives such as irrigation, domestic, hydroelectricity, industrial, mining, and regeneration. However, India has the largest irrigated area in the whole world, and the irrigated area is only about 40% of the cropped area [10]. One of the main reasons for this low irrigated land is the major use of traditional irrigation methods, which leads to low water use efficiency of about 35–40% [11].

The use of traditional methods without the reach of cloud computing and edge computing causes the unstable watering system for the plants. Consequently, a well-organized and judicious watering system is the major intention. During some last decades, irrigation systems with the use of some sensor networks with different IoT approaches are initiated which basically provides the solution but still they need some improvements. Table 1 shows their water-saving percentages, techniques used by them, and the sensors used by them.

In 2008, Bernard used the rain sensor and estimated the eminence of pasture with and without the sensor. He tried to figure out irrigation water use. He experienced common Bermuda grass to achieve 34% water saving. Xiao et al. [13] self-designed the sensor network for the irrigation system, and they achieved water saving of about 65.22%. Dukes [20] described that water saving of about 40% to 70% can be achieved by using smart controllers but for real-world scenarios of bigger fields; this value can be lessened to 10% [19].

Gutiérrez et al. [14] designed and tried to implement a mechanized irrigation system to use water efficiently. They used a wireless network of some sensors to manage water saving of about 90% as compared to conventional irrigation methods. Similarly, Kumar et al. [5] presented a similar work in the same year and Parameswaran and Sivaprasath [6] and Rawal [16] latterly introduced a few similar sensor-based solutions. Nelson in 2015 used a few sensor data such as temperature and soil moisture and WSAN to automate the irrigation process with decreased water consumption. Saab et al. [17] tried and thrived an on-field survey of a smart phone irrigation setting up. He investigated and tested that application in Mediterranean environments achieved 25% of water saving. Recently, another input to these contributions was made by Saqib (2020), i.e., a network system for the HC12 module is intended to improve the communication range.

3. Architecture of the Proposed System

The anticipated irrigation system is entrenched with the potential smart decisions taking capability to water plants by considering the factors such as crop type, soil type, climate type, temperature, humidity, and soil moisture. Ontology is implanted to query about the decision for a particular plant type, climate type, and soil type, while remaining factors such as temperature, humidity, and soil moisture are sensed by our sensor network. Final decision for watering plants or not relies 50% on the ontology result, and the other 50% is based on our trained machine learning model. The smart architecture of our watering system is given in Figure 1.

Our proposed architecture of IoT has four layers, application layer, processing layer, transport layer, and the perception layer, rather than basic IoT architecture which consists of three layers (application layer, network layer, and perception layer). The perception layer is known as the physical layer, which means it has sensors for assembling data. It senses temperature, soil moisture, and humidity from air. The transport layer is the source of transferring sensed data collected previously to the processing layer through networks such as wireless, 2G, 3G, and LAN. The processing layer stores, scrutinizes, and processes huge amounts of data coming from the transport layer. It utilizes technologies such as databases, cloud computing, and edge computing. The application layer is for providing application-specific services to the end user. Our system deals with the sensors, GSM module, edge server + IoT server, and additionally an android application. These are the perception layer, transport layer, processing layer (cloud computing and intelligent computing), and the application layer, respectively.

3.1. Sensor Data. At first, data are gathered by the sensors as presented in Figure 2. Soil moisture, humidity, and temperature data are collected in this phase. The perception layer has all sensors, actuators, and the microcontroller. Rest is the part of remaining three layers. Transport and processing layers collectively provide schedule for watering crops, their supervision, and other suggestions. After gathering the data, the next stage is to accumulate data at data centers for analyses.

The detailed design inspection of the physical components used is presented in the figure. All the components are with no trouble available in the market and cheap also. So, the device to be deployed in the real environment can be made easily available. This implantable device has the layer of sensors used, i.e., humidity, light, and moisture sensors. The microcontroller fixed in the Arduino board receives the analog signals from these sensors, and after every 30 seconds, these values are transferred to the data center through GSM module SIM808. The final results from our decision-making process can be visualized by the user all the way through an android application, after which the user can direct our system's actuators, and finally, water is released from the valve or closed.

The next section briefs the working of our ML smart decision system deployed at the IoT server which speedily timetables the watering plan for plants. This setting up also evolves the soil type, climate type, and crop type. In our smart system, ontology inhabits in these parameters for better competence and precision. By means of these technologies, we have prepared our system to be fully functionally automatic. The subsequent section describes the semantic knowledge base for our smart irrigation system.

Table 1: Sensor-based solutions.

Work year	Sensor occupied	Technique/methodology	Water saved
Cardenas-Lailhacar et al. (2008) [12]	Rain sensor	Soil moisture sensor system	34%
Xiao et al. (2010) [13]	Self-designed wireless sensor	WSN	65.22%
Gutiérrez et al. (2014) [14]	Soil moisture sensor VH400 Temperature sensor DS1822 Temperature sensor (LM35)	WSN and GPRS	90%
Kumar et al. (2014) [5]	Humidity sensor (CLM53R) Soil pH sensor	WSN and XBee-based communication	No result
Nelson et al. (2015) [15]	Soil moisture sensor	WSAN with cloud platform	72%
Parameswaran and Sivaprasath (2016) [6]	Soil moisture sensor	Drip irrigation with IOT	No result
Rawal (2019) [16]	Soil moisture	Sprinklers with IOT	1000 m^3/ha
Saab et al. (2019) [17]	Blueleaf tool	DSS with IOT	25.7%
Saqib et al. (2020) [18]	Soil moisture sensor	WSNs	No results
Grady et al. (2019) [19]	Prototype/model	Edge computing	No results

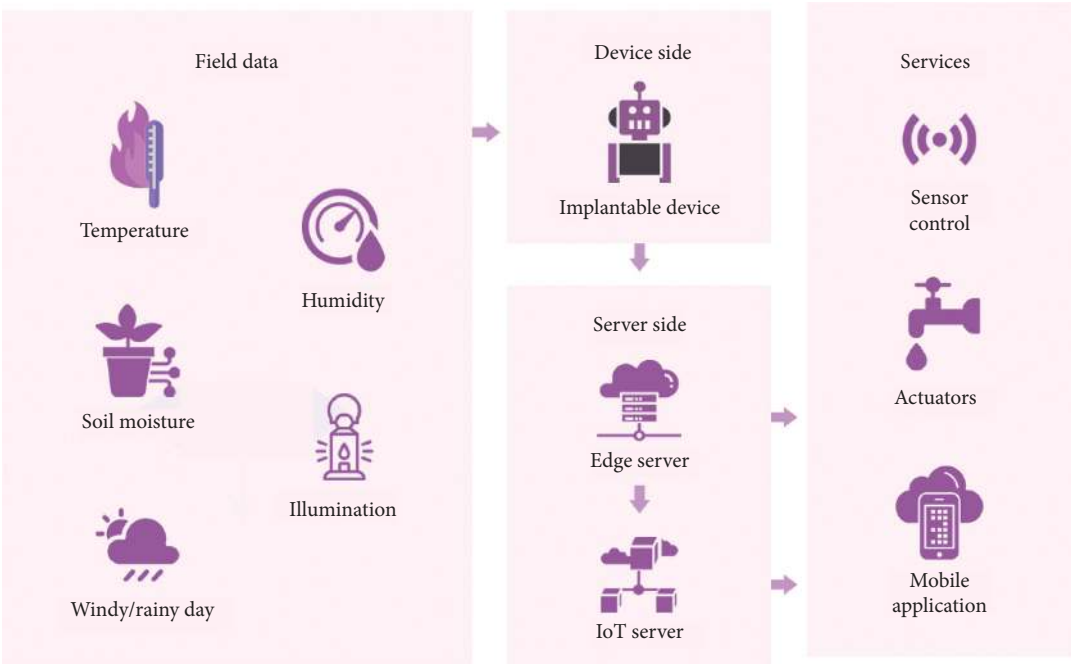


Figure 1: Proposed architecture for smart irrigation.

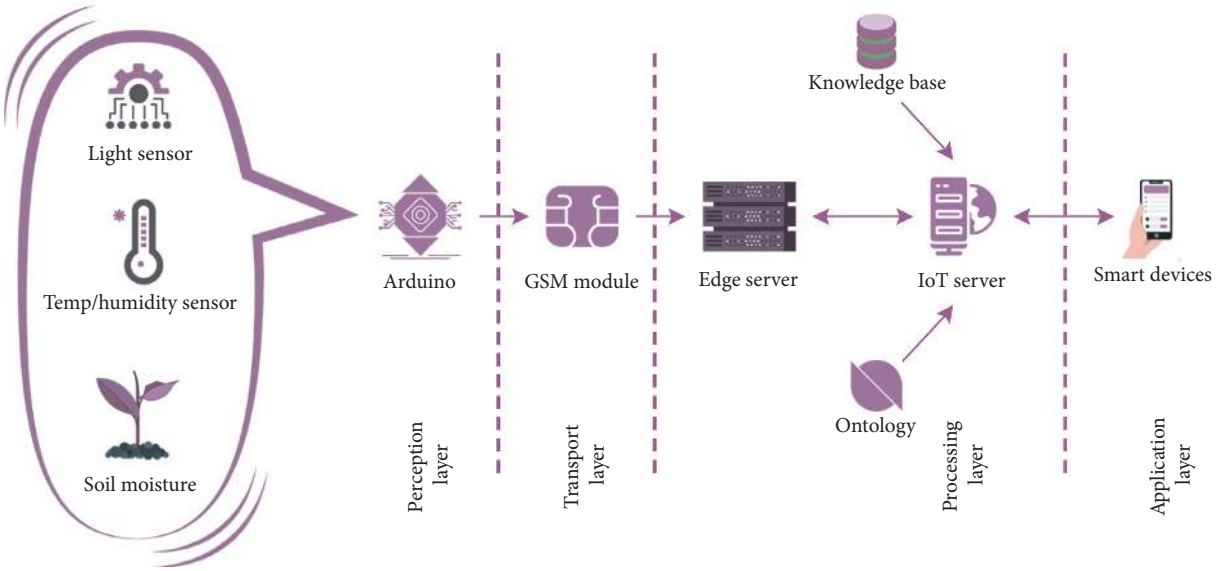


Figure 2: Hardware design for the integrated system.

3.2. Semantic Knowledge Base. The semantic data model (SDM) is designed for incorporating and handling of the real-world data. In the semantic data model, the logical levels are applied for the categorizing of concepts and evaluation of the information. On the basis of the results extracted from propositional logic systems set in the ontology, one can make a smart decision.

There are concepts in our ontology to make prediction of the level of water need on the basis of crop type, climate type, and soil type. These parameters collectively constitute the structured data that why we can query decision on their bases from the given ontology. The sensed data and the decision resulted in SPARQL (RDF query language) together comprise the full ground vital to make a watering system run. For instance, if, due to the climate type and soil strata type, a particular plant requires water, it would be contingent to water the plant. This action of watering crops is the consequence of the actuation that is performed on the valve. Likewise, it could be turned off as directed by the field specification.

Sensor data are pulled together at different levels of a large area. The observed properties or the sensed data such as temperature, humidity, and luminance are measured at the yard level, while soil moisture (superficial and deep) is measured at the quadrant level. These data in the form of RDF and the desired knowledge (crop species, climate types, and soil types) from ontology are sent to the control agent. The control agent also receives data about plant requirement for quantity of water in specific soil texture.

The ontology on which our system depends is vast and complex due to the wide range of factors/features engaged in taking decision for watering plants or not (Appendix A). An abstract view of ontology is shown in Figure 3. There are different climate zones of Pakistan, and they are distinguished into four different types such as highland zone, arid zone, lowland zone, and coastal zone. As the humidity level is diverse in diverse areas, irrigation in these climatic zones has wide-ranging water needs.

In addition to temperature and humidity, another feature, soil type, also influences the level of water need to be given. The clay which is known as well-drained like loamy soils is the excellent soil type for wheat [21]. There are four to five different types of soil considering their structure and texture. In the same way, each crop has its some specific water needs as some require more water such as sugarcane and rice than others such as wheat and cotton.

The architecture of our decision support system is shown in Figure 4. The information about crop types is giving the watering requirements of the crop by utilizing plant ontology. Then, data sensed from pasture/crop land, soil type, and climate type is used for depiction on the actual watering supplies for the field. Water instructions or suggestions will be shown as recommendations on the mobile phone via an android app, and as a resultant of a button click from the farmer's smart phone, actuations will be executed on the valves positioned in the field.

3.3. Used Analysis Technique. Water requirement level can be predicted by any machine learning approach such as random forest, decision trees, KNN, naive Bayes, and support vector machine as all of these are classification dilemma-handling algorithms. The modeling practice we are using lies underneath supervised machine learning, known as KNN (with $k = 5$). It uses the whole dataset to predict an unseen data instance. It searches through the whole dataset to find "k" number of neighbors which are the most close neighbors to that data instance. This is done by actually finding the correspondence between the instance data with the whole dataset, where "k" is the number of neighbors found closer to instant data. If the value of "k" is set to 3, then three most similar neighbors will take part in assigning class label to instant data. It then allocates the most common class label (among those k-training instances) to the test data. Shemim et al. utilized three feature selection algorithms, CBFS, FPRS, and KFRS, for the dataset, and then KNN is used to classify featured classes [21]. Bzdok et al. also discussed about supervised learning algorithms including KNN in 2018 [22].

3.3.1. Algorithm for KNN

Step 1: calculate the Euclidean distance between new data X (4 features involved to predict the resultant class, A, B, C, and D) and each existing point P_n in the input dataset S:

$$\text{Euclidean distance} = \sqrt{(XA - PA)^2 + (XB - PB)^2 + (XC - PC)^2 + (XD - PD)^2}. \quad (1)$$

Step 2: choose the value of "k," i.e., no of nearest neighbors to new data X:

$$k = 5. \quad (2)$$

Step 3: count the ballots of all the "k" neighbors to predict the class of test data X.

Step 4: assign that class to the test data X, which won more votes.

4. Application of the Proposed Architecture

Our system to be implemented uses an Arduino UNO (ATmega328P) controller. The data sensed by the sensors (perception layer) are received by the microcontroller, they are transferred to the edge server (1st processing layer) via GSM SIM808 (transport layer), in which basic scrutiny occurs, and just the immediate data required to predict the resultant water level are transferred to the main IoT server

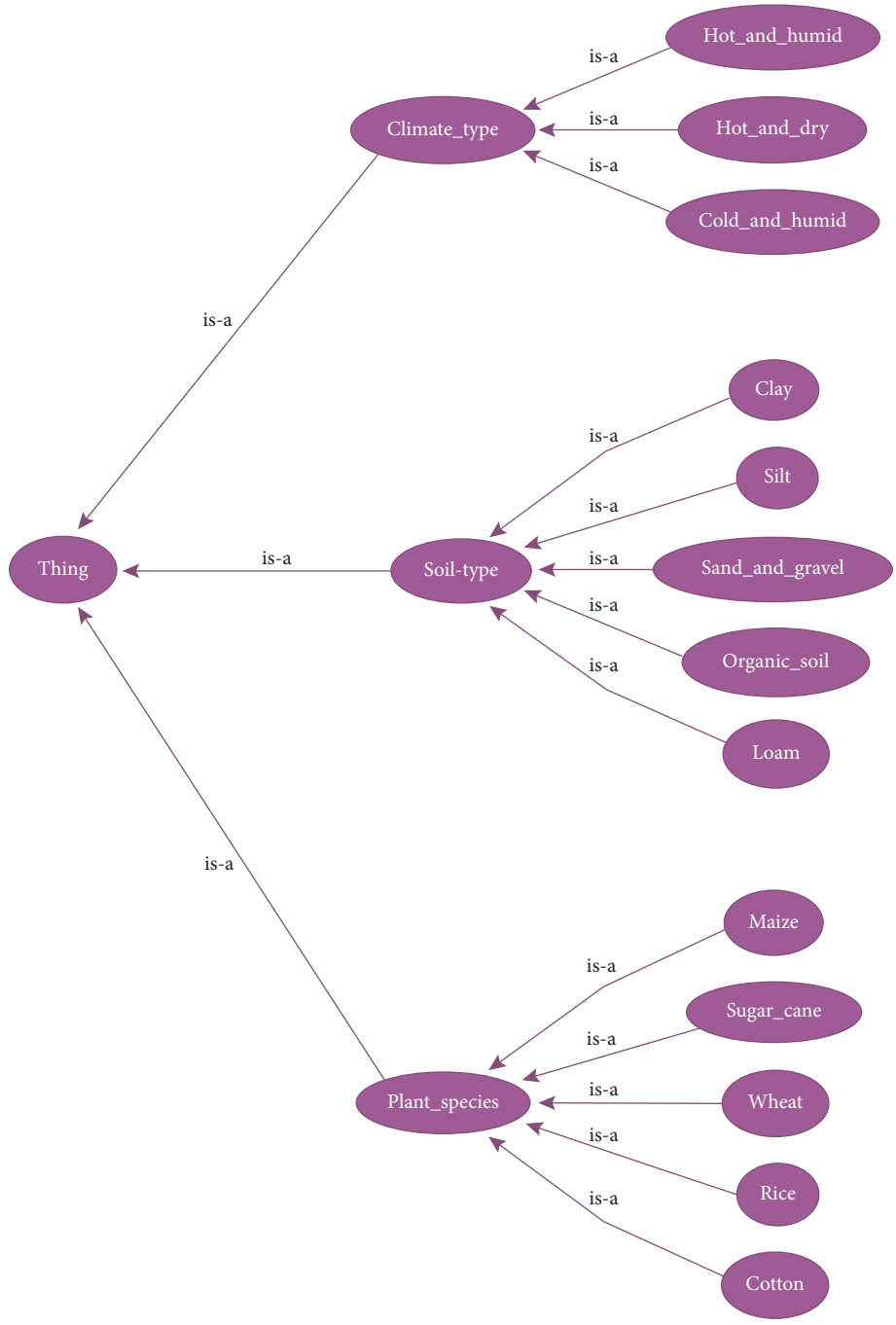


Figure 3: Ontology: an abstract view.

(2nd processing layer) where our trained machine learning model is deployed. This model, after detailed analysis, tells the rank of water need for a field. The following section elaborates the hardware setting.

4.1. Hardware Setting for the User. Embedded sensors used in the IoT-based system are the source of sensing inventively and cost-effectively, and they can record and analyze real-time data (Sarwar, Bajwa, Ramzan et al. 2018 and Munir, Bajwa and Cheema 2019) [23]. The proposed smart IoT system as shown in Figure 5 employs some sensors to gather data from the environment, and a GSM module SIM808 is used to transfer the values to the edge server. A data SIM card is inserted in it to get facilitated by the real-time data transportation. As we can see in Figure 6, a hygrometer sensor is used for soil moisture, while for the moisture from air, AM2302 DHT22 (temperature/humidity) sensor is used. Their details are described in the following.

4.1.1. HL-69 Soil Hygrometer Sensor. For the detection of the humidness of the soil, we used HL-69 soil hygrometer moisture sensor. The basic purpose for using the HL-69 soil

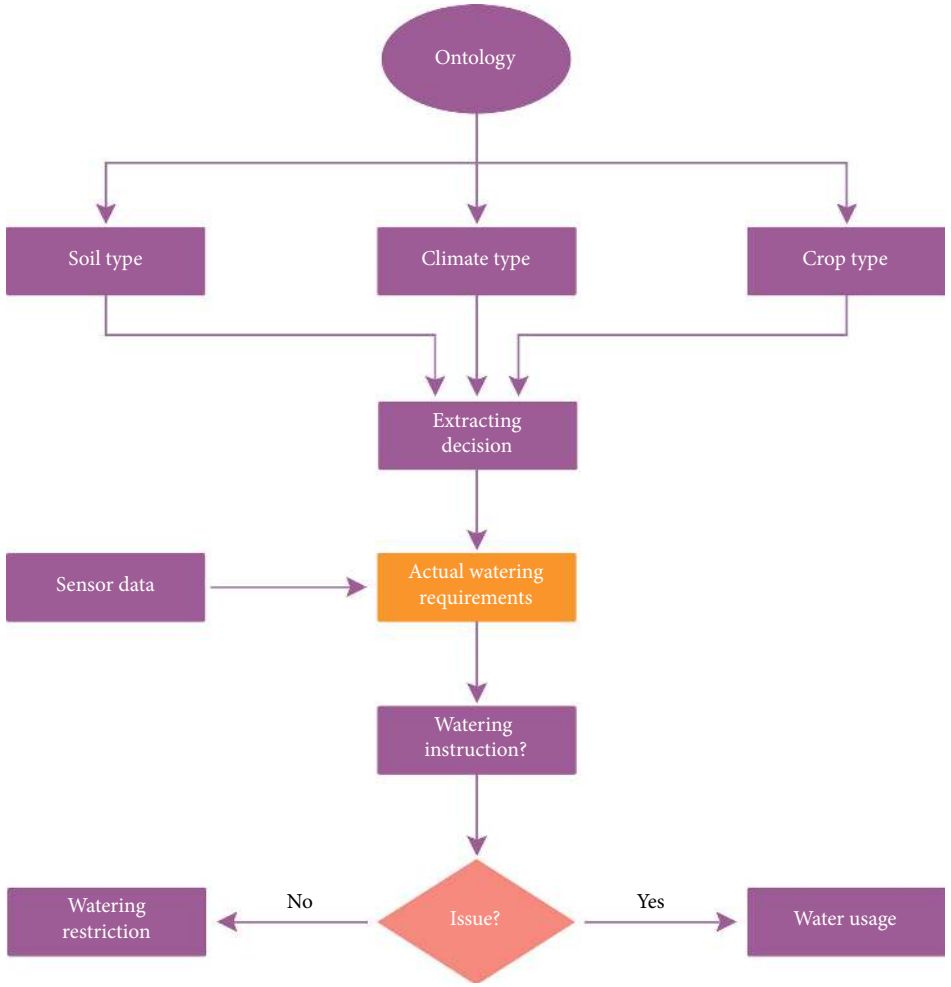


Figure 4: Inference rule schema.

hygrometer moisture sensor is to provide better reading than other soil moisture sensors. This sensor is used for real-time monitoring soil moisture of plants in a tunnel farm. The voltages of the sensor output change accordingly to the water content in the soil. There are some key factors of HL-69 soil hygrometer sensor. If soil moisture is greater, then the output voltage decreases, but if the soil is dry, then the output voltage increases. The hygrometer soil moisture sensor provides an analog signal as an output which has to be converted to digital by Arduino. This sensor includes two pieces: one is an electronic board and another one is two pads that detect the water content. LM393 comparator chip is located on the electronic board. The electronic board of the HL-69 soil hygrometer sensor has a fixed bolt hole used for easy installation. It contains two lights: red and green; red light shows the power indicator, and the green light shows the digital switching output indicator.

4.1.2. AM2302 DHT11 Sensor. The DHT22 sensor is a common temperature-humidity sensor that is used to determine temperature and humidness in air. The DHT22 sensors are made up of two parts: a humidity sensor and a thermistor. There are some key factors of the DHT22 sensor which are as follows: the cost of the DHT22 sensor is low. DHT22 sensor is good for 0–50% temperature readings with 2–5% accuracy and a humidity range from 20 to 80% with 5% accuracy. The I/O voltage for the DHT22 sensor is between 3 V and 5 V. While requesting data, the maximum current use during conversion is 2.5 mA. DHT22 sensor contains 4 pins with 0.1 spacing between them. The body size of the DHT22 sensor is approximately 15.1 mm *25 mm *7.7 mm.

4.1.3. BH1750 FVI Light Sensor. BH1750 is a common digital light sensor that can determine the light intensity. BH1750 is a calibrated digital light sensor, and it can measure even small traces of light and can convert it into a 16-digit numeric value. It is commonly used in mobile phones to exploit the screen brightness based on the environmental lighting. BH1750 measures the light intensity in the range of 0 to 65,535 lux (L). In the smart irrigation system for tunnel farming, we used the H-resolution mode of the BH1750 sensor. There are some key factors of BH1750 sensor which are as follows: the chip of the BH1750 sensor is BH1750FVI. The power supply of the BH1750 sensor is 3.3 V to 5 V. The BH1750 sensor is a built-in 16 bit AD converter that converts detection of light into a 16-digit numeric value.

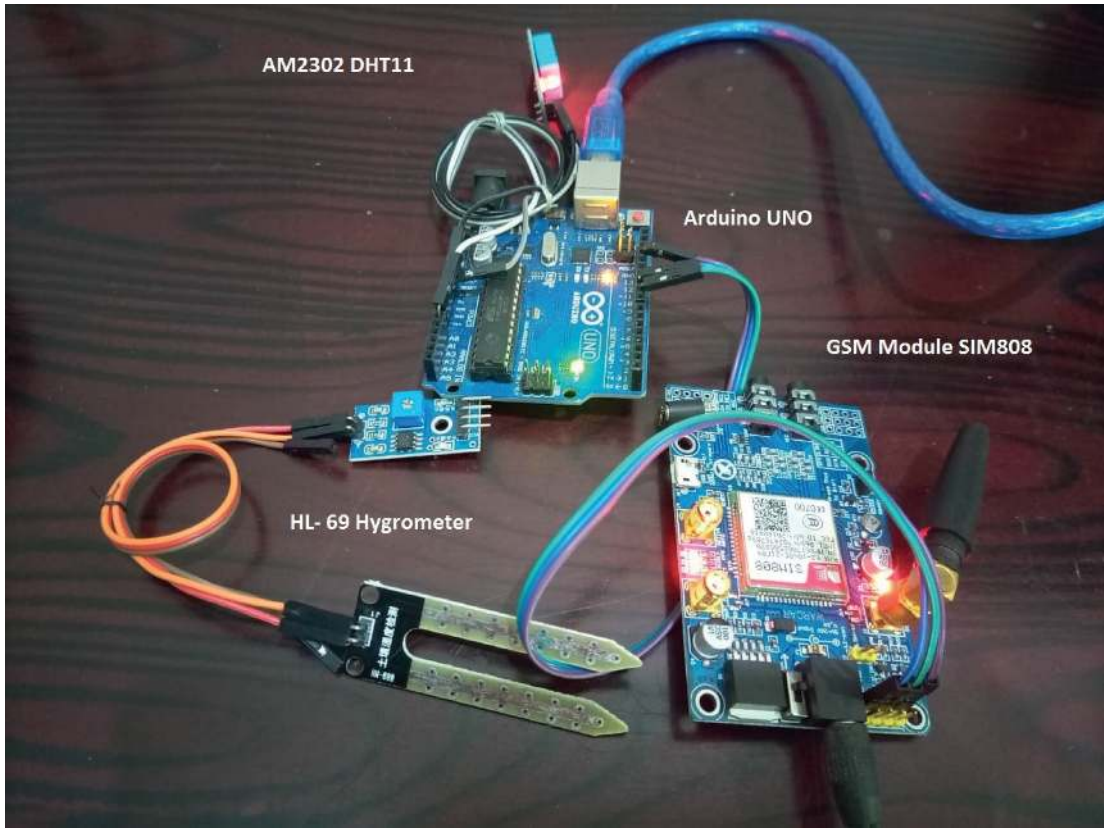
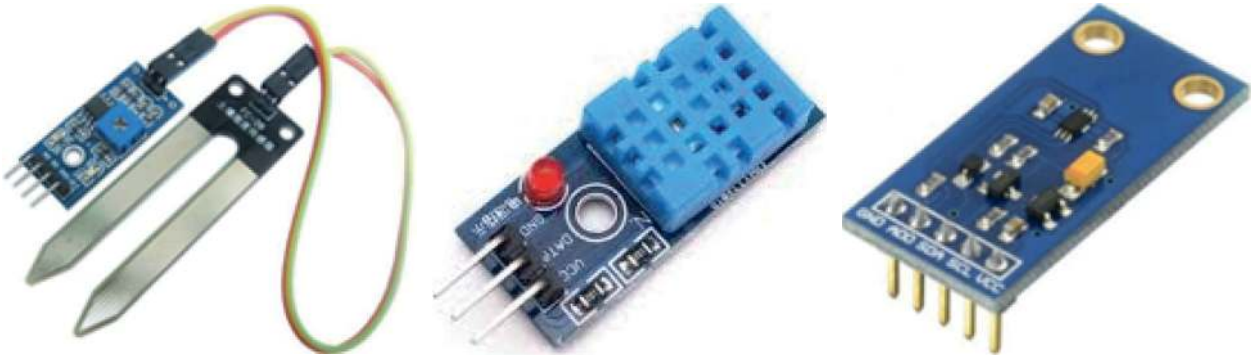


Figure 5: Hardware prototype.



HL-69 hygrometer soil moisture sensor

AM2302 DHT22 temperature-humidity sensor

BH1750 light sensor

Figure 6: Sensors used.

The range of light intensity in the BH1750 sensor is 0 to 65,535 lux. The size (L*W) of the BH1750 sensor is approximately 3.2 cm*1.5 cm.

5. Results and Discussion

The proposed watering system for tunnel farming is so smart that develops and employs the assistance of true decision-making capability of machine learning. The architecture and the hardware details of the system are given in the preceding section. All the sensors (temperature and humidity, light sensor, and the soil moisture sensors) were deployed to the actual field to analyze the reaction of the proposed system. The data transferred to the edge server through the GSM module and through an Android application whereas the results can also be seen by a farmer. A user can then perform some actuation to open or close the valve.

5.1. Preparing the Training Dataset. The system is completely automated as the sensor data receiving from the field are processed according to our trained model of machine learning, i.e., trained by the characteristic sensor values shown in the following. Table 2 shows five classes: highly

needed, needed, average, not needed, and highly not needed for various levels of soil moisture sensed by the HL-69 hygrometer sensor and temperature and humidity sensed by AM2302 DHT22. The output of a HL-69 hygrometer varies from 0 to 870, while the humidity level of the AM2302 DHT11 sensor varies in the air from 20 to 80%, and its temperature value ranges from 0 to 50.

Here is our sampled training dataset shown in Figure 7 based on our rules set in Table 2, which we have provided to our machine learning algorithm to predict water needs for the given crop types.

5.2. Training of the KNN Model. We have implemented the code in Anaconda, created for python programs, and have trained our model on the fact that, on a particular temperature, water requirements of different crops, which we are taking into consideration, can be given.

Rice > sugarcane > maize > cotton > wheat.

This general rule can be elaborated more specifically by identifying ranges for temperature, humidity, and soil moisture for all the given types of crops to recognize its class. Here is the rule summary in Table 3.

In Table 3, labels "HN," "N," "A," "NN," and "HNN" are second hand for classes highly needed, needed, average, not needed, and highly not needed correspondingly. Likewise, working rules for watering considering climate and soil are given in the following.

Sand and gravel > clay > silt > loam > organic soil.

Hot and dry > hot and humid > cold and humid.

5.3. Deploying the Trained Model. Our trained model has been developed using Scikit-learn. To make it available to production and to make it useful for end users so that they could extract real values from it, we have deployed it. In this regard, we need to have three components shown in Figure 8.

The developed and trained model for predicting the water level as a "model.pkl" file is ready, and model evaluation is provided in the results section. The web service we have used for this purpose is the Flask API. Lastly, we need cloud as a service provider, and Heroku server fulfilled our requirement in this proposed system.

5.4. Implementation Using Edge Computing. In the process of deploying the trained model through Flask API, we actually define routes to where an HTTP request handles. One route is for handling one HTTP request. The data are travelled in the system from one side (perception layer) to other (edge server).

The piece of code in Figure 9 is responsible for sending the sensor data from the sensor-Arduino side (i.e., perception layer) to edge server Firebase (i.e., processing layer). Second last line of the code is establishing a link to which data (temperature, humidity, soil moisture, and phone no.) are transferred. These data are received by our edge server by a route "/submit" defined in the application of Flask API as shown in Figure 10. Whenever data from sensors send to the established link of HTTP request, it triggers the following piece of code to run. In this piece of code, the sensor values and the SIM card no. (phone no.) are inserted in our database server (Firebase).

Table 2: Classes of sensor data.

Soil moisture (%)	Temperature (°C)	Humidity (%)	Class
<30	>45	<30	Highly needed
30–45	35–45	30–45	Needed
46–60	25–34	46–60	Average
61–80	20–24	61–80	Not needed
80–100	<20	>80	Highly not needed

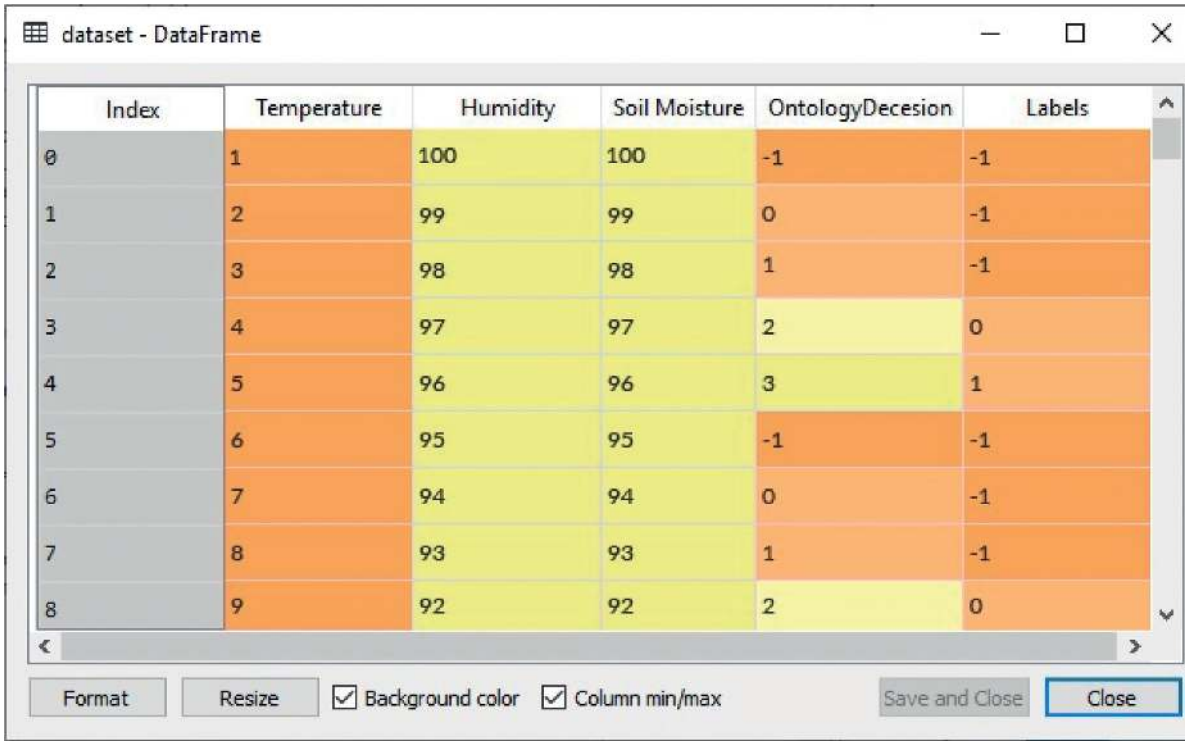
As we can see in Figure 11, each phone no. is representing a different device. Any data coming from a particular device are stored under the hierarchy of its phone no. Each record under a device has a key value associated with it, which actually contains the sensor data values. Whenever a particular record arrives at the edge server, its key value stores in the parameter "latestKey" under its phone no. When another entry of record happens, its key value replaces the previous one. In this way, our database is designed to have the record of most recent data entered in it.

Firebase can be omitted from the system, and data can directly be sent to the Heroku server (IoT server), i.e., cloud computing. That is really a bad practice due to overburden of the IoT server with useless data. Since sensors are sending each and every instant value to the IOT server and IOT server is responsible for scrutiny of data coming from a device (which means three sensors values every 30 seconds), and then applying model to predict value for water requirement. It will definitely affect the speed and efficiency of the system. This is the main concept of introducing edge computing.

The piece of code in Figure 12 is triggered by the smart mobile/tab when the user clicks to predict results for water requirements. It utilizes the trained model to predict the water requirements and return the value to the server. This value is sent to the user, and he/she can see the result on his/her phone via the app.

5.5. Android Application. An android platform is provided to the farmers. The input parameters (crop type, climate type, and soil type) are put on view in a dropdown, and users can select from these and can send the command to the device implanted to the field.

Codes for crop types, soil types, and climate types are transferred from the mobile app interface in Figure 13(a) to the server to which ontology is attached. Decision extracted from the ontology section along with the sensor values then reaches the main IoT server where our machine learning algorithm is installed. Our training dataset also contains the encoded values for labels for different classes which are converted to the text (class label) at the front end in the android app as in Figure 13(c). These codes are shown in Table 4.



Index	Temperature	Humidity	Soil Moisture	OntologyDecesion	Labels
0	1	100	100	-1	-1
1	2	99	99	0	-1
2	3	98	98	1	-1
3	4	97	97	2	0
4	5	96	96	3	1
5	6	95	95	-1	-1
6	7	94	94	0	-1
7	8	93	93	1	-1
8	9	92	92	2	0

Figure 7: Sampled training dataset.

Table 3: Working rules for different crops.

Rule no.	Temperature	Humidity	Soil moisture	Ontology decision	Class
1	T >50	H <20	SM <20	HN	HN
2	T >40	H <40	SM <40	HN	N
3	T >30	H <60	SM <60	HN	A
4	T >20	H <80	SM <80	HN	NN
5	T <20	H >80	SM >80	HN	HNN
6	T >57	H <20	SM <10	N	HN
7	T >40	H <30	SM <30	N	N
8	T >35	H <40	SM <40	N	A
9	T >30	H <60	SM <60	N	NN
10	T <30	H >60	SM >60	N	HNN
11	T >57	H <20	SM <10	A	HN
12	T >40	H <30	SM <30	A	N
13	T >35	H <40	SM <40	A	A
14	T >30	H <60	SM <60	A	NN
15	T <30	H >60	SM >60	A	HNN
16	T >50	H <30	SM <40	NN	HN
17	T >40	H <40	SM <60	NN	N
18	T >30	H <60	SM <80	NN	A
19	T >20	H <90	SM <100	NN	NN
20	T <20	H >90	SM >100	NN	HNN
21	T >50	H <30	SM <30	HNN	HN
22	T >40	H <50	SM <60	HNN	N
23	T >30	H <60	SM <80	HNN	A
24	T >20	H <80	SM <90	HNN	NN
25	T <20	H >80	SM >90	HNN	HNN

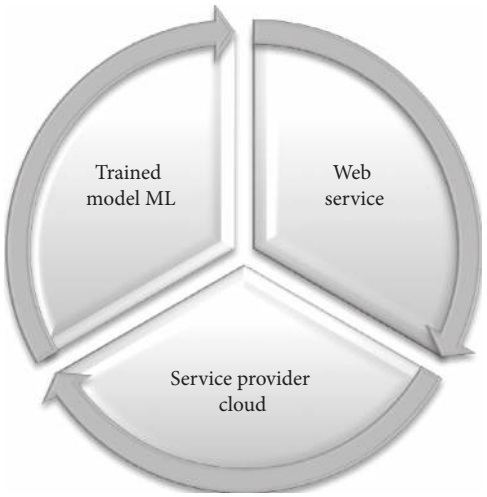


Figure 8: Components required in deployment.

SIM808_HTTP-POST | Arduino 1.8.9

File Edit Sketch Tools Help

Verify

SIM808_HTTP-POST §

```
String url = "https://irrigate-api.herokuapp.com/submit?";  //URL for HTTP-POST-REQUEST
String temp;
String humi;
String soilmoisture;
String phone;
 void gsm_sendhttp() {
  mySerial.println("AT+HTTPINIT");
  runsl();
  delay(100);
  mySerial.println("AT+HTTPPARA=CID,1");
  runsl();
  delay(100);
  mySerial.println("AT+HTTPPARA=URL," + url+"temp="+temp +"&humi="+ humi + "&soilmoisture=" + soilmoisture + "&phone=" + phone);
  runsl();
```

Figure 9: HTTP request sending to store sensor data.

```
@app.route('/submit', methods=['POST'])
def submit():
    dump(request)
    if request.values and len(request.values) == 4:
        data = {
            'temperature': request.values['temp'],
            'humidity': request.values['humi'],
            'soil_moisture': request.values['soilmoisture']
        }
        print('data ===> ', data)
        result = firebase.post('/sensor-data-collection-7d34e/data/' + request.values['phone'], data)
        firebase.put('/sensor-data-collection-7d34e/data/' + request.values['phone'], 'latestkey', result)
        return json.dumps({'status': 'true', 'message': 'Data Saved!'})
    else:
        return json.dumps({'status': 'false', 'message': 'Invalid request!'})
```

Figure 10: Route defined for inserting values to the database.

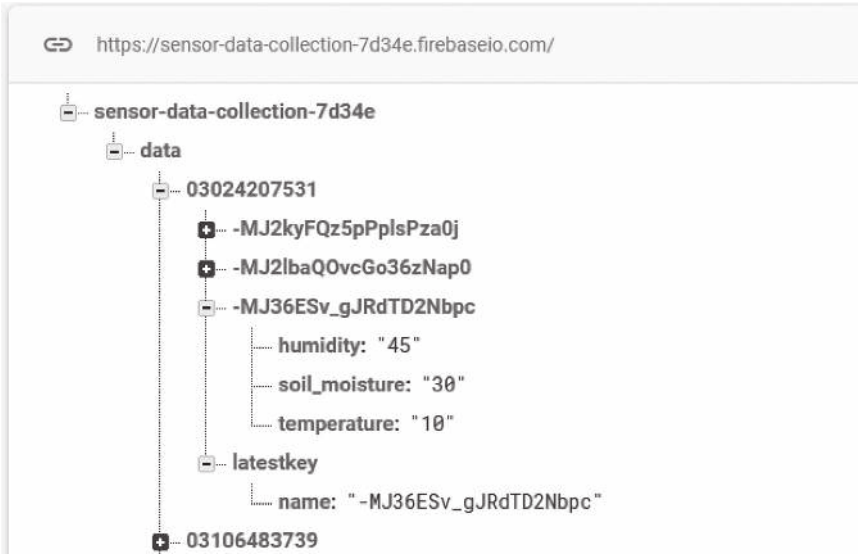


Figure 11: Edge server handling data from sensors via the HTTP request.

```
@app.route('/predict', methods=['POST'])
def predict():
    dump(request)
    if request.values and len(request.values) == 4:
        crop = request.values['crop']
        climate = request.values['climate']
        soil = request.values['soil']
        phone = request.values['phone']
        latestkey = firebase.get('/sensor-data-collection-7d34e/data/' + phone + '/latestkey', 'name')
        humidity_result = firebase.get(
            '/sensor-data-collection-7d34e/data/' + phone + '/' + latestkey,
            'humidity')
        soilmoisture_result = firebase.get(
            '/sensor-data-collection-7d34e/data/' + phone + '/' + latestkey,
            'soil_moisture')
        temperature_result = firebase.get(
            '/sensor-data-collection-7d34e/data/' + phone + '/' + latestkey,
            'temperature')
        model = pickle.load(open("./model/model.pkl", "rb"))
        prediction = model.predict(
            [[humidity_result, soilmoisture_result, temperature_result, crop, climate, soil]])
        print(prediction)
        return json.dumps({'status': 'true', 'message': str(prediction[0])})
    else:
        return json.dumps({'status': 'false', 'message': 'Invalid request.'})
```

Figure 12: Data transfer from edge server to IoT server with prediction results.

For example, we choose, from the dropdowns in the user input screen shown in Figure 13(a), sugarcane as a crop type, hot and dry as a climate type, and loam as a soil type. After clicking the button "Send" from the Figure 13(b) interface, the sensor readings come across to the server.

The values for humidity, temperature, and soil moisture and the encoded result for soil type, climate type, and crop type values are considered by our trained ML model to recognize the watering need for the specific crop. So, with the 50% result coming from ontology and the sensor values, our system foretells to water the crops and displays a note on the farmer's mobile screen as shown in Figure 13(c).

The highlighted text "Highly need water" is mainly the output of our machine learning algorithm already discussed in the previous section. As shown in Table 4, our training dataset holding the labels ranges from −1 to 3. These are effectively the degree of water necessity to a specific plant at specific time.

5.6. Performance Evaluation. We have performed tests on our sample data, which we have obtained randomly from about 500 instances. We provided these instances to train our KNN model for the proposed system to forecast class

(a) (b) (c)

Figure 13: Android app different interfaces (a, b, and c).

Table 4: Codes for class labels.

Class	Code
Highly needed	3
Needed	2
Average	1
Not needed	0
Highly not needed	−1

labels. We used two statistical measures to estimate the performance of our KNN model, i.e., precision and recall. Figure 14 shows the accuracy report of the results of the KNN model providing $k = 5$.

The accuracy report of the trained model is given in Figure 1 and also presented graphically. Predicted results for class label "Needed" are lacking in precision. This performance is dependent on the "k" value that is the no. of nearest neighbors involved in the predicting class. Figure 15 shows the confusion matrices without normalization and with normalization.

To increase accuracy, we should choose the "k" value precisely. As per general rules for the KNN algorithm, the value of "k" for the problem of two classes should be an odd value, and for more than two classes, the "k" value should not be the multiple of the number of the resultant classes. As in our case, we have five labeled classes to be predicted, so we will choose "k" accordingly. In order to choose a suitable value for "k", we have to plot "k value versus mean error" graph to identify the error trend. So, we plot it by using "matplotlib.pyplot" in Figure 16.

As we can see, the mean error initially increases up to 0.5 as the "k" value increases, but there is a sudden fall which occurs after that to the value of 0.3 when the "k" value reaches 10 to 11. After that, rise in error rate starts, and it continues to increase with the increase in the "k" value. It means that our "k" value should be "11" that is the maximum value of 'k' for which mean error remains lowest. So, for the value of $k = 11$, we again find the accuracy report to check if the performance of our model is getting better or not.

Figure 17 shows the significant improvement in the performance of the model as accuracy rate increases when we set the "k" value to 11. The precision value for class "Not Needed" is increased from 0.33 to 1.0 which means accuracy rate increases from 33% to 100%. Similarly, for class "Average," the precision value increases from 0.25 to 0.33 which means that accuracy rate increases from 25% to 33%. This is how tuning of the model can be possible. By tuning training data values and by adjusting the "k" value, we can have a better model for our system. This is the main reason why we choose KNN algorithm for our proposed system.

	Precision	Recall	F1-score
−1	0.8	1	0.89
0	0.33	0.2	0.25
1	0.25	0.33	0.29
2	0	0	0
3	1	1	1
Accuracy			0.45
Macro avg	0.48	0.51	0.48
Weighted avg	0..42	0.45	0.43

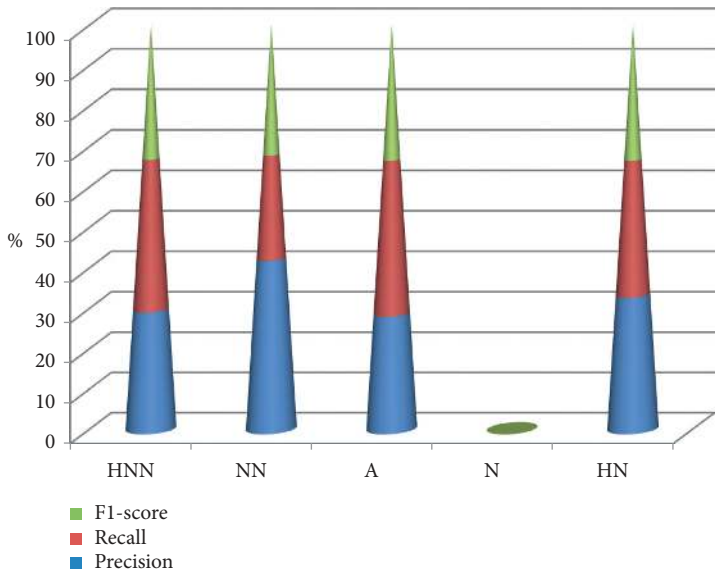


Figure 14: Accuracy report with "$k = 5$."

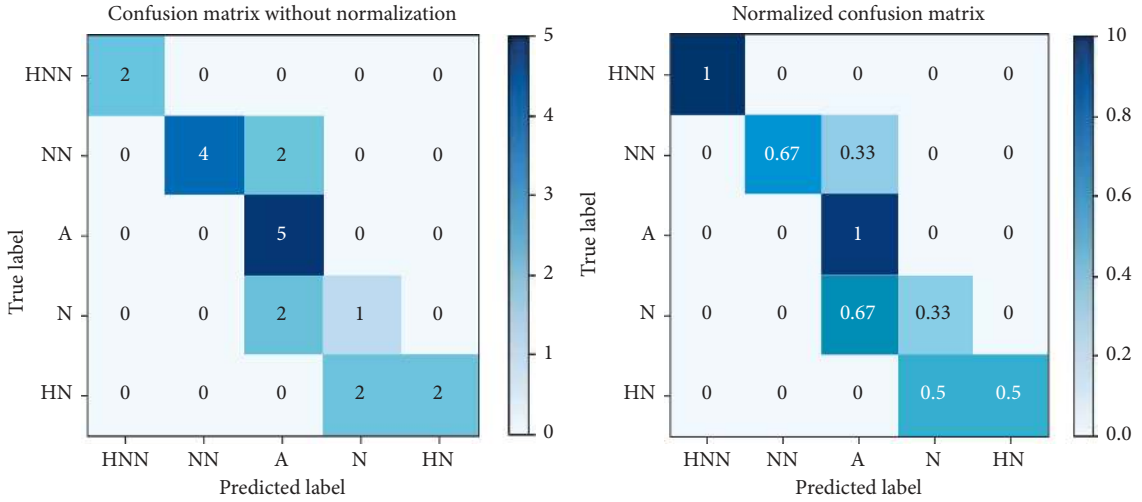


Figure 15: Confusion matrices without normalization for "$k = 5$."

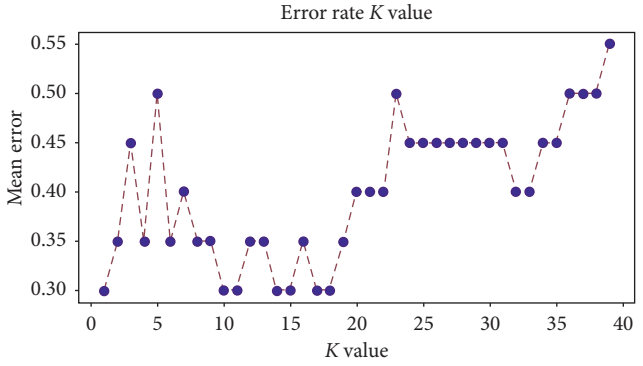


Figure 16: Error rate with respect to the k value.

	Precision	Recall	F1-score
−1	0.8	1	0.89
0	1	0.5	0.67
1	0.33	1.00	0.50
2	0	0	0
3	0.83	0.83	0.83
Accuracy			0.65
Macro avg	0.59	0.67	0.58
Weighted avg	0.64	0.65	0.61

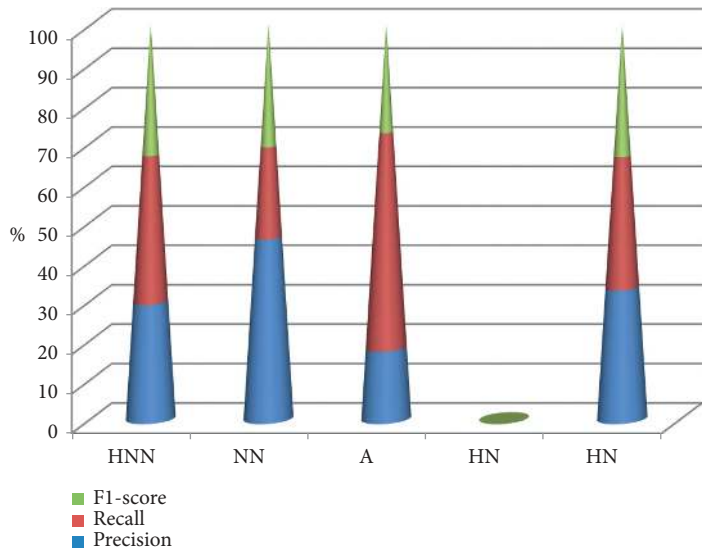


Figure 17: Accuracy report with "$k = 11$."

6. Conclusion and Future Work

Our proposed solution for smart irrigation constitutes three modules: first module is the sensor network, which is required to sense parameters influencing the water need. We have used sensors DHT22, light sensor BH1750, and HL-69 hygrometer to sense temperature, soil moisture, light, and humidity in air. In the third module, we use edge and main IoT servers to transfer and receive data via HTTP requests. In the second module, we applied KNN on the sample dataset to train the model and used it for efficient decision-making of water requirements. Our trained model classifies the input into five possible classes based on input values such as highly not required, not required, average, required, and highly required. We have fully implemented the proposed system in Anaconda.

Currently, our system employs KNN for decision-making, but other intelligent data-extracting techniques can also be used for decision-making. So, the presented irrigation system can reproduce in future by using other decision-making techniques such as random forest. Moreover, the edge computing architecture can be further improved by making the edge server responsible for processing data and depicting the result from the machine learning algorithm. In other words, the trained model for KNN can be deployed at the edge server so that nearby devices to a particular edge can get facilitated by that edge server. It will improve latency rate remarkably.

Data Availability

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

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] M. S. Munir, I. S. Bajwa, M. A. Naeem, and B. Ramzan, "Design and implementation of an IoT system for smart energy consumption and smart irrigation in tunnel farming," *Energies*, vol. 11, no. 12, p. 3427, 2018.

[2] H. Sattar, I. S. Bajwa, R. U. Amin et al., "An IoT-based intelligent wound monitoring system," *IEEE Access*, vol. 7, pp. 144500–144515, 2019.

[3] B. Sarwar, I. Bajwa, S. Ramzan, B. Ramzan, and M. Kausar, "Design and application of fuzzy logic based fire monitoring and warning systems for smart buildings," *Symmetry*, vol. 10, no. 11, p. 615, 2018.

[4] B. Sarwar, I. S. Bajwa, N. Jamil, S. Ramzan, and N. Sarwar, "An intelligent fire warning application using IoT and an adaptive neuro-fuzzy inference system," *Sensors*, vol. 19, no. 14, p. 3150, 2019.

[5] A. Kumar, K. Kamal, M. O. Arshad, S. Mathavan, and T. Vadamala, "Smart irrigation using low-cost moisture sensors and XBee-based communication," in *Proceedings of the Global Humanitarian Technology Conference (GHTC)*, San Jose, CA, USA, October 2014.

[6] G. Parameswaran and K. Sivaprasath, "Arduino based smart drip irrigation system using internet of things," *International Journal of Engineering Science*, vol. 6, p. 5518, 2016.

[7] C. Kamienski, J.-P. Soininen, M. Taumberger et al., "Smart water management platform: iot-based precision irrigation for agriculture," *Sensors*, vol. 19, no. 2, p. 276, 2019.

[8] M. Stubbs, *Irrigation in US Agriculture: On-Farm Technologies and Best Management Practices*, Congressional Research Service, Washington, DC, USA, 2016.

[9] F. TongKe, "Smart agriculture based on cloud computing and IOT," *Journal of Convergence Information Technology*, vol. 8, no. 2, 2013.

[10] A. Narayanamoorthy, "Economics of drip irrigation in sugarcane cultivation: case study of a farmer from Tamil Nadu," *Indian Journal of Agricultural Economics*, vol. 60, pp. 235–248, 2005.

[11] M. W. Rosegrant, X. Cai, and S. A. Cline, "Global water outlook to 2025: averting an impending crisis," International

Food Policy Research Institute, Washington, DC, USA, 572-2016-39087, 2002.

[12] B. Cardenas-Lailhacar, M. D. Dukes, and G. L. Miller, "Sensor-based automation of irrigation on Bermuda grass, during dry weather conditions," *Journal of Irrigation and Drainage Engineering*, vol. 134, pp. 184–193, 2008.

[13] K. Xiao, D. Xiao, and X. Luo, "Smart water-saving irrigation system in precision agriculture based on wireless sensor network," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 26, pp. 170–175, 2010.

[14] J. Gutiérrez, J. F. Villa-Medina, A. Nieto-Garibay, and M. A. Porta-Gandara, "Automated irrigation system using a wireless sensor network and GPRS module," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 1, pp. 166–176, 2014.

[15] N. Sales, O. Remédios, and A. Arsenio, "Wireless sensor and actuator system for smart irrigation on the cloud," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 693–698, IEEE, Milan, Italy, December 2015.

[16] S. Rawal, "IOT based smart irrigation system," *International Journal of Computer Applications*, vol. 159, no. 8, pp. 7–11, 2017.

[17] A. Saab, M. Therese, I. Jomaa, S. Skaf, S. Fahed, and M. Todorovic, "Assessment of a smartphone application for real-time irrigation scheduling in Mediterranean environments," *Water*, vol. 11, p. 252, 2019.

[18] M. Saqib, T. A. Almohamad, and R. M. Mehmood, "A low-cost information monitoring system for smart farming applications," *Sensors*, vol. 20, no. 8, p. 2367, 2020.

[19] M. J. OGrady, D. Langton, and G. M. P. O'Hare, "Edge computing: a tractable model for smart agriculture?," *Artificial Intelligence in Agriculture*, vol. 3, pp. 42–51, 2019.

[20] M. D. Dukes, "Water conservation potential of landscape irrigation smart controllers," *Transaction ASABE*, vol. 55, pp. 563–569, 2012.

[21] M. S. Munir, I. S. Bajwa, and S. M. Cheema, "An intelligent and secure smart watering system using fuzzy logic and blockchain," *Computers & Electrical Engineering*, vol. 77, pp. 109–119, 2019.

[22] D. Bzdok, M. Krzywinski, and N. Altman, "Machine learning: supervised methods," *Nature Methods*, vol. 15, pp. 5-6, 2018.

[23] M. Safdar Malik, I. Sarwar Bajwa, and S. Munawar, "An intelligent and secure IoT based smart watering system using fuzzy logic and blockchain," *Computers and Electrical Engineering*, vol. 77, no. 1, pp. 109–119, 2018.