BACHELOR OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

BRAC
UNIVERSITY

Inspiring Excellence

Machine Learning based data processing and latency reduction in the Internet of Things for agriculture

Authors
Sajeed Ur Rahman
Leonard Michael Gomes Dip
Manan Moin Shuddho
Kishwar Maheen

Supervisor
Amitabha Chakrabarty, Ph.D
Associate Professor
Department of CSE

**A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of
B.Sc. Engineering in CSE**

**Department of Computer Science and Engineering
BRAC University, Mohakhali, Dhaka - 1212, Bangladesh**

December 2018

Declaration

It is hereby declared that this thesis /project report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

Authors:

Sajeed Ur Rahman
Student ID: 14201054

Leonard Michael Gomes Dip
Student ID: 14201056

Manan Moin Shuddho
Student ID: 14201018

Kishwar Maheen
Student ID: 15101021

Supervisor:

Amitabha Chakrabarty, Ph.D
Associate Professor, Department of Computer Science and Engineering
BRAC University

December 2018

The thesis titled ***"Machine Learning based data processing and latency reduction in the Internet of Things for agriculture"***.

Submitted by:
Sajeed Ur Rahman Student ID: 14201054
Leonard Michael Gomes Dip Student ID: 14201056
Manan Moin Shuddho Student ID: 14201018
Kishwar Maheen Student ID: 15101021
of Academic Year 2018 has been found as satisfactory and accepted as partial fulfillment of the requirement for the Degree of **B.Sc. Computer Science and Engineering** .

1.	Amitabha Chakrabarty, Ph.D Associate Professor Department of Computer Science And Engineering	Supervisor
2.	Md. Abdul Mottalib, PhD Professor and Chairperson Department of Computer Science And Engineering	Chairman
3.	Dr. Md. Motaharul Islam Associate Professor Department of Computer Science and Engineering	Session Chair
4.	Dr. Md. Golam Rabiul Alam Assistant Professor Department of Computer Science and Engineering	Session Coordinator
5.	Sadia Hamid Kazi Assistant Professor Department of Computer Science and Engineering	Member

Acknowledgements

This research has been supervised by Dr. Amitabha Chakrabarty, Associate Professor, Department of Computer Science and Engineering, BRAC University. We would like to thank him for his relentless support and patience throughout the period. We also want to thank the BRAC IT for allocating a dedicated computer for us, our work might have been undone without that. Lastly, we are thankful to all the faculty members of BRAC University who have inspired and motivated us throughout the entire undergraduate program.

Abstract

The Internet of Things is best stated as a network of “things” that have the ability to generate and share information between themselves and interact with the environment according to the percepts from this environment. This network between these devices and humans generates a massive amount of data that often leads to problems such as clogged networks, unhandled data traffic and, thus, greater latency throughout the network. Machine learning, in the IoT sector, helps to manage the data in a more seamless way. Classification plays a significant role in predicting discrete actions based on the learnings from the extracted data of the “things”. We have used agricultural data in order to predict controls that lead to maintaining the optimal conditions for a select crop in our controlled environment. We have successfully used a classification algorithm in our proposed model after analyzing the metrics of five algorithms, namely Decision Tree, Logistic Regression, K-Nearest Neighbours, Gaussian Naïve Bayes and Support Vector Classification to decide which one would be most suitable for our scenario which is based on agriculture. Furthermore, using our proposed model, we have successfully reduced latency throughout the scenario by bringing data processing closer to the “things”. Our main purpose was to determine which classification algorithm was suitable for implementing data processing in our scenario by using accuracy, precision, recall, specificity and F1-score, as well as to reduce latency to make the IoT within our scenario more effective and efficient using our proposed model. Experimental results showed Decision Tree having the best performance with best scores of 100% in every metric and the latency being decreased to approximately a range of 0.002 to 0.003 seconds.

Keywords: Internet of Things, IoT, Machine Learning, Classification, Decision Tree, Latency, Accuracy, Sensors, Actuators.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The Internet of Things concept came a long time before the title was coined by Kevin Ashton in 1999 as a concept of objects and people being identified by computers, currently as "things", and connected to a network where these "things" could interact with each other with minimum fuss [2]. One of the first and most important Internet-connected appliances came in 1991 as a modified coffee pot known as "Trojan room coffee pot" in University of Cambridge, which could notify the researchers of the university through the use of just a web-cam connected to a network server whether the machine was loaded or not. This was one of the earliest example of "things" connected to a computer and within a network, where humans and that coffee pot as a "thing" were both involved [12]. Further research and a mesh of multiple technologies such as ubiquitous computing, real-time analytics, embedded systems and several others brought the Internet of Things to life [2]. Since there is no one widely accepted definition of the Internet of Things to the best of our knowledge, we have provided our own definition of the Internet of Things based on our own understanding and research: A network of Internet-connected objects or "things", which include sensors and actuators, and people, identified by the devices they use, in which these "things" can freely interact with each other and interpret the information received between each other.

In our scenario, the "things" are referred to as the sensors and actuators that measure crop conditions and actuators that control them. At this moment, there are only handful of sensors available to us for measuring factors affecting crops. These are temperature, light intensity, humidity, soil moisture and rainfall. However, there are several other factors, such as soil pH, fertilizer concentration, nutrient concentration for each type of nutrients, atmospheric carbon dioxide concentration etc. for which the sensors are not widely available. Also, the actuators available to manipulate such features are unavailable. Water pumps, water sprinklers and other such control devices are the only available ones to control only a select few features such as humidity, soil moisture and light intensity. As our main aim in this scenario is to increase or maximize yield all around the year, we tried to implement Machine Learning to keep the soil moisture and temperature under control in a controlled closed environment by training our proposed model using a dataset to restrict these features within threshold values that we assume should maximize the yield.

1.1 Motivation

While the Internet of Things is set to be the next revolutionary technology amongst individuals and businesses alike, the number of devices connected, estimated by analyst firm Gartner to reach 20.4 billion by 2020, is set to generate too much information for human beings to analyze and interpret [24]. This was highlighted by Ashton. He claimed during 2009 that computers are largely, if not completely, dependent on human beings for their information. He also stressed that people do not have enough time, concentration and accuracy to process all of that information. Not only will the data being generated by the vast network of devices be humongous, it will also be heterogeneous [2]. Different types of sensors will generate different types of data. For example, a temperature sensor will provide temperature, a pH sensor will provide the pH level, humidity sensor will provide humidity, and so on. Currently, these types of information are manually analyzed and interpreted by human beings, and several of these types of data will be generated which is when the limitations they have will come to light. The question is, how can we tackle the problem such that we can analyze and interpret any volume of heterogeneous data efficiently and quickly? Thus, our motivation for this research is to find out an effective and quick way which can help us analyze, interpret and tackle this huge pile of heterogeneous data generated by IoT systems.

1.2 Contribution

We implemented Machine Learning, more specifically the Decision Tree algorithm, in the top-most layer of our model that we have proposed in this report. By doing so, we have successfully been able to handle the data to control actuators based on the threshold values as well as decreased latency by bringing the data processing closer to the sensors and actuators.

1.3 Methodology

We are proposing a new model to transform the IoT experience into a more advanced and manageable one. The proposed model consists of multi-layered devices that are interconnected and controlled over the internet. To implement the framework we used a Raspberry Pi for IoT implementation, Thingspeak as a platform for the IoT data storage and a laptop for implementing the topmost layer for our model and for implementing machine learning there.

In order to achieve learning at the Domain level, we required a Machine Learning algorithm that would allow the devices at that level to be able to make decisions based on the data they receive. Decisions are sent and received in discrete forms of data and are derived from continuous data. Therefore, an algorithm was required to derive discrete values representing commands or feedback from continuous data.

In Supervised Machine Learning, there are two types of algorithms – regression and classification. Regression involved predicting continuous values from continuous data, whereas classification involved predicting discrete values from continuous data [14]. Hence, a classification algorithm was specifically suited to our cause, but we needed to know which classification algorithm was most suitable. Therefore, we chose a few classification algorithms to analyze their performances before comparing them to decide which one was most suitable for implementation on the Domain level of our Domain-Entity-Node framework, and ended up choosing Decision Tree based on our results analysis (see Chapter 6.1). Decision Tree is a model that can solve both regression and classification problems. It is basically a tree with nodes and edges. The nodes either resembled a condition used to narrow the problem down, or a solution to the problem itself. The edges led to other nodes. The condition asks a simple yes or no question which helps in narrowing down the problem [7]. Other algorithms used in analyzing performance were Logistic Regression, which uses best fit lines to fit continuous variables with binary variables [6], Support Vector Machine that uses a hyper-plane between value points to differentiate between discrete values [17], K-Nearest Neighbours that uses a certain number of neighbouring values to classify [21] and Gaussian Naïve Bayes which uses the Bayes' probability theorem to determine discrete values [16]. The results' analysis were concluded based on the confusion matrix and the metrics derived from the confusion matrix. It is a matrix that represents the relationship between actual and predicted values in the prediction test [23].

The confusion matrix consists of four values in a 2x2 matrix with actual values as columns and predicted values as rows. This is the case when a discrete target variable has only 2 possible values. How the matrix is viewed for discrete values of 1 and 0 is given in Table 1.1

Number of true positives are the number of predicted values of 1 with corresponding actual values also being 1. Number of false positives are the number of predicted values of 0 with corresponding actual values actually being 1. Number of false negatives are the number of predicted values of 1 with corresponding actual values actually being 0. Number of true negatives are the number of predicted values of 0 with corresponding actual values

	Actual (1):	**Actual (0)**
Predicted (1):	Number of true positives (index 0, 0)	Number of false positives (index 0, 1)
Predicted (0):	Number of false negatives (index 1, 0)	Number of true negatives (index 1, 1)

Table 1.1 Structure of a confusion matrix derived from a classification algorithm

also being 0 [23]. Accuracy is the fraction of correct predictions made by the model over all of its predictions. It is a fairly proper measure of performance for models in which positives and negatives are somewhat balanced. However, it is not a good measure when one massively outnumbers the other amongst actual values. It is given by the following formula of Accuracy 1.1.

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{1.1}$$

TP is the number of true positives, TN is the number of false negatives, FP is the number of false positives and FN is the number of false negatives [23].

Precision is the fraction of true positive predictions made over the number of total positive predictions made by the model. It tells us, in our case, whether the domain was right in turning on a relay (see Chapter 5.1).It is good to measure of minimization of false positives. Precision is given by the following formula of Precision 1.2 [23].

$$P = \frac{TP}{TP + FP} \tag{1.2}$$

Recall is the fraction of true positive predictions made over the sum of numbers of true positive and false negative values. It gives a measure of how many times the relay was turned on by the algorithm when it should have been turned on (see Chapter 5.1). It is given by the following formula of Recall 1.3 [23].

$$R = \frac{TP}{TP + FN} \tag{1.3}$$

Specificity is the fraction of true negative predictions made over the sum of numbers of true negative and false positive values. [2] It gives a measure of how many times the relay was turned off by the algorithm when it should have been turned off (see Chapter 5.1). It is given by the following formula of Specificity 1.4 [23].

$$S = \frac{TN}{TN + FP} \tag{1.4}$$

F1-Score is the harmonic mean between precision and recall of a model. It is used to provide a measure of both precision and recall at the same time. It is given by the following formula of F1-Score 1.5 [23].

$$F = \frac{2xPxR}{P+R} \tag{1.5}$$

These metrics were used to determine performance of the five model, that were compared in order to determine which one was most suitable to implement for the Domain level in our Domain-Entity-Node framework.

1.4 Thesis Outline

In **Chapter 1**, we basically discussed about our thesis motivation, methodology and introduction about our thesis topic.
In **Chapter 2**, we emphasized upon literature review. Here we tried to show a summary of related papers and knowledge we gathered from them.
In **Chapter 3**, we proposed our IoT-based framework. In this part we elaborately discussed about the architecture of our framework and certain scenarios.
In **Chapter 4**, we have given an overall view about our data and data preprocessing.
In **Chapter 5**, we implemented our proposed framework. We included both hardware implementation and machine learning implementation in this section.
In **Chapter 6**, we described and elaborated the analysis and findings of our proposed model as compared to conventional IoT.
In **Chapter 7**, we stated the limitations we faced in our thesis. Moreover, we added our future works here.
Lastly, in **Chapter 8**, our thesis report concludes.

Chapter 2

Literature Review

To conceptualize the topic “Internet of Things”, Ashton, K. described that computers identify objects and people as “things”, and these things are connected to a network where they could interact with each other without any big hassle [2]. From the point of view of Kamsin, A., Alansari, Z., Anuar, N.B, Soomro, S., Belgaum, M.R., Miraz, M. and Alshaer, J. The Internet of Things offers a chain of connected people, objects, applications, and data over the Internet for remote control, interactive, services integration and management [1]. However, there are some problems and challenges regarding IoT. Mohammeda, Z., Ahmedb, E. in their paper showed that Data volumes and Data interpretation are two of them [26]. Data interpretation is the challenge of generalizing the local context from the sensor data as accurately as possible. Data volume is the amount of the mass data that are generated by the live sensors embedded in the "Things" of the Internet of Things. It increases at an increasing rate when more and more devices are connected in network. The unbounded and unhandled heterogeneous data may lead to a massively clogged internet and proof to be very inefficient in terms of scalibility. Díaz, M., Martín, C. and Rubio, B. highlighted the importance of cloud computing in Internet of Things as well as conducted a survey of integration components based on cloud and Internet of Things. They also surveyed various integration proposals and data analytics techniques, and discussed about the challenges and research issues based on the Internet of Things. One of those challenges mentioned latency, of which they highlighted fog computing as a potential solution. However, they acknowledged that fog computing does not have the capability to handle complex analysis. We tried to solve this problem by implementing Machine Learning in one of the levels of our proposed model [4].

On the other hand, according to Wu, Q. et al. in their paper Cognitive Internet of Things (CIoT) is basically Internet of Things (IoT) infused with human cognition, which allows objects involved within IoT, which are already able to gather information by ‘seeing’, ‘hearing’ and ‘smelling’ the outside world, to ‘learn’, ‘think’ and understand data and information by themselves with minimized human intervention [25]. In the whitepaper published by IBM, Kelly, J. E. mentioned that cognitive computing involved three core concepts to it called learning, understanding and reasoning [11]. We initially wanted to involve cognitive computing on a basic scale just to research how our proposed model would work using those concepts. However, due to severe hardware limitations and lack of open-source cognitive libraries, we had to resort to Machine Learning instead to implement our proposed model for demonstrative purposes. Using Machine Learning, however, partially covers the learning concept of cognitive computing, which IBM confirmed as a crucial aspect of that concept. To make the conventional IoT system cognitive Mezghani, E., Exposito, E. Drira, K. discussed about implementing cognitive computing embedded into the IoT network using the five fundamental cognitive tasks of perception-action cycle, massive data analytics, semantic derivation and knowledge discovery, intelligent decision making and on demand service provisioning, In order to link the physical and social environments with smart decision making and service dispatch system [15]. Since introduction of cognitive computing in the Internet of Things is asserted to have human like cognition, it can be said that the data manageability of CIoT will be significantly better than that of the conventional IoT. The astounding amount of data generated will be cognitively handled, stored, organized and used to learn and refer from within the system itself. The use of cloud storage platforms, Internet of Things platform and various other Platforms as a Service will be more efficiently used. Besides that, data interpretation will also be affected by this newly introduced cognition. Hu, S., Wang, H., She, C. and Wang, J. proposed an ontology for the Internet of Things regarding agricultural data, called AgOnt, which is aimed at addressing semantic heterogeneity resulting from increased sensor networks in the agricultural field [8]. Since they have used the Internet of Things in the most conventional form, despite addressing the crucial aspect of handling heterogeneous data, the issue of high latency in the conventional network of IoT remains. Their future concerns involved having to manage agricultural ontology at real-time, which we have addressed to our utmost efforts by reducing the latency heavily using our proposed model.

By using various machine learning algorithms we can find out and predict the original condition of the context from data given by sensors. It can also handle the heterogeneity of the produced data. Stojkoska, B. L. R., Trivodaliev, K. V in their paper "A review of Internet of Things for smart home: Challenges and solutions" talked about a multi-layered framework called "Holistic Framework" to handle the heterogeneous data generated in the inter-connected smart home system. However, in their framework all the collected data eventually sent to cloud for processing. As a result, it does not solve the latency issue in IoT [22]. Hence, we have tried to address these specific problem domains and come up with probable solutions with the means of our proposed model (see chapter 3). Our main aim was to decentralize the data handling from the cloud to layers that are closer to the devices itself. Since the data processing is brought down closer to the devices, the dependency on the cloud can be significantly reduced together with the overwhelming network traffic generated by the heterogeneous data that IoT produce. Carvalho, H. F. and Kim, Y. worked on an implementation of the Internet of Things to control temperature and humidity in a home-style greenhouse manner. This implementation is similar that of our proposed model. However, human intervention is required to control the actuators in their implementation. In our proposed model, due to the use of Machine Learning, the actuators are controlled by the Internet of Things system in an autonomous manner. Therefore, no human intervention is required in our model. This reduces the risk of human error in the agricultural sector as well as removes the painstaking effort of a person having to control the system everyday [3].

Chapter 3

Proposed Model

We are proposing a new framework to transform the IoT experience into a more advanced and manageable one. The proposed framework consists of multi-layered devices that are interconnected and controlled over the Internet. The framework essentially has three very important layers to begin with. The layers are namely, the Domain layer, the Entity layer and the Nodes layer together forming the Domain-Entity-Node (DEN) framework. Here in our proposed system, the things will no longer be a cluster of ordinary analog or digital sensors sending continuous streams of mostly unfiltered data. The things will be controlled by a machine that has the ability to think, predict, act, react and respond according to changes in behaviour of the data that it processes and the experiences that it gathers.

3.1 Network Architecture: Domain-Entity-Node (DEN) Framework

The Domain-Entity-Node (DEN) network framework for IoT is proposed to take the era of IoT a step forward. The very first level that we are going to describe is the shallowest, consisting of the "Things" of the IoT infrastructure; a layer we are calling the Nodes layer, where each "Things" connected to the framework will be referred to as nodes. The second layer, called the Entity layer will consist of smart and powerful devices that acts intermediary to the nodes and the higher level. This layer begins to form the very first clusters within the entire framework. The highest layer within our framework is called the Domain layer. This layer will essentially act like a parent to all the layers below. A machine with learning capabilities will be the focus of this layer. The machine connects and controls several entities from the Entity layer. Thus, forming the second clusters within the framework. The nodes are basically the sensors and actuators that send and receives data to and from the entity layer.

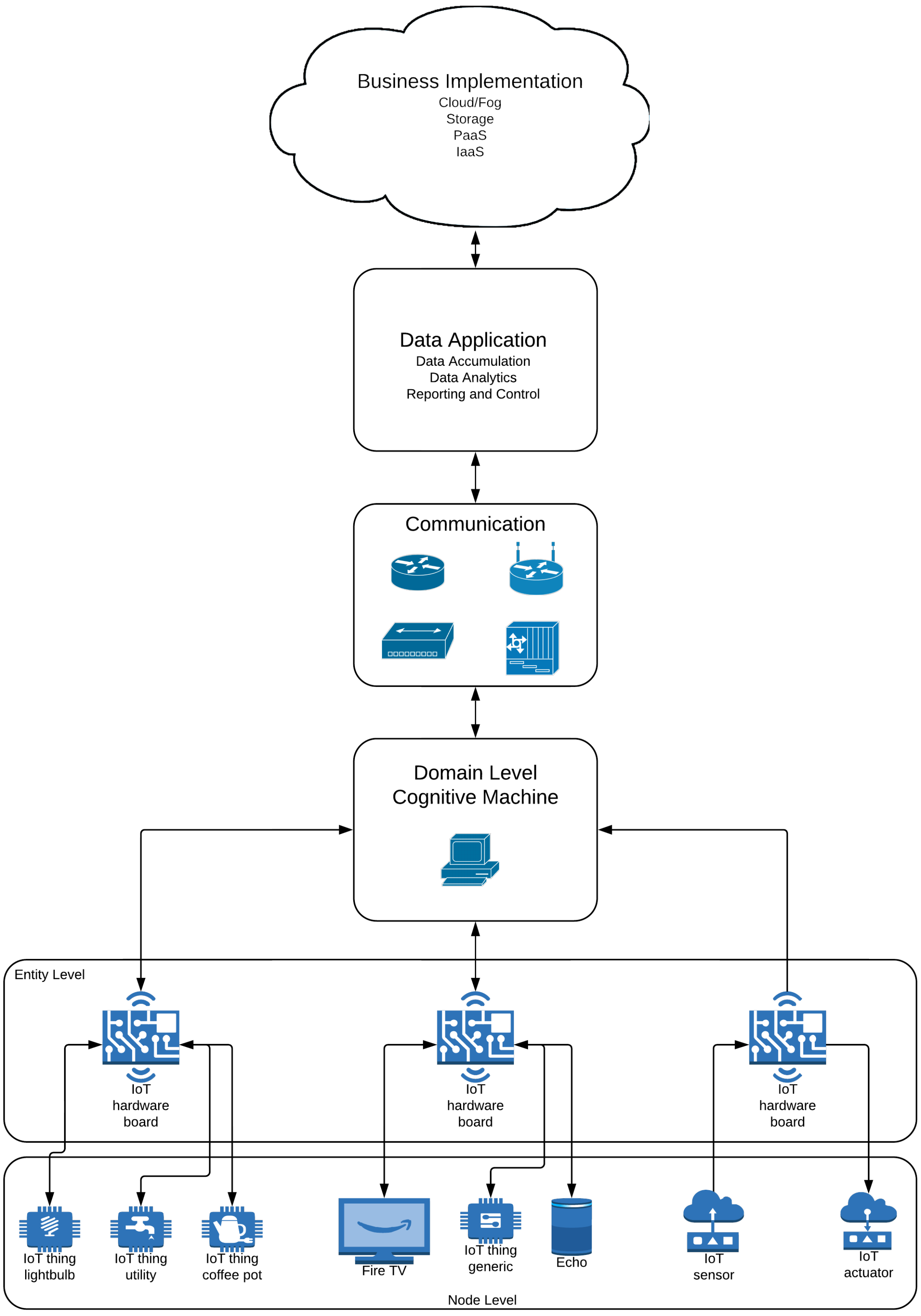

Fig. 3.1 The Domain-Entity-Node (DEN) framework

Domains: These are the largest and most powerful segment of the three. As shown in the figure 3.1, it primarily acts as the parent system to the clusters of entities under its control. The domains have two imperative features. Firstly, it is the only gateway to the immediate upper layer and adjacent domains. Secondly, the domains use machine learning to control the decision-making processes of the machines in the immediate lower level (entities). The Domain layer itself handles all the heterogeneous data that the entities are unable to handle or finds as exceptions. This way, all the data are filtered and sorted into homogeneity and processed within this domain system that comprehends the data, learns the possibilities of outcomes, makes rational decisions based on previous experiences and current demands of situation, performs an efficient and precise action and controls the entities and nodes below it to act accordingly. The Domain layer will consist of domain systems and machines that are interconnected within themselves as well as the lower entities and upper devices providing connectivity to the cloud. There will be inter-domain communication when necessary data required for a function that is in need and is being prompted, is missing from the domain's own collective experiences. The domain will principally consult with other domains and ask for suggestions to perform a task more efficiently when it does not have enough knowledge to take the decision alone itself. The domain layer will therefore transform into a chain of knowledge and experience sharing systems that learns from every point of access to data anywhere. The domain will then be able to perform any task that would previously require a human's intelligence to be performed. This system will substantially increase the preciseness of decision-making and data analysis of anyone who wishes to use this system.

Entities: These are the second layer in the DEN framework. These are essentially, a cluster of artificially intelligent systems that controls the nodes and are under control of the domain. The child systems shown in the second layer from the bottom of the figure: 3.1 are called the entities, have enough capabilities to organize, handle and interpret heterogeneous streams of data received from the nodes preliminarily. The entities learn from the data that they receive from the nodes, processes the data as intelligently as it can within its limitations, recognizes it as an event, situation, or a trigger and generates an intelligent response suited for the event, situation or trigger that has just been identified and activated. They also send the now organized and structured data that they received to their corresponding parent domains. Any new and unrecognized form of data is tried to be handled within the entity systems and learned by matching user information with already learned and experienced events. However, if it does not find a pattern or any further information regarding processing this data, it simply passes the entire stream of unhandled, unsorted and unfiltered data that it is unable to comprehended the parent domain system. Those data that are not possible to be handled and are sent to the parent system for broader investigation are then registered within the framework as a new unknown event, situation, or trigger.

Nodes: These are the lowest segment of them all. Comprising of the collections of sensors and actuator that forms the traditional Internet of Things, the nodes form a cluster themselves for every event and for every entity that they correspond to. Such several clusters of nodes will be connected to their corresponding entity, which are further connected to their corresponding domains. Sensors act as the devices that pick up signals and information from the surrounding environment of our physical world. The sensors receive real-world information, converts it into digital data and sends the data as-is in real-time to the entity that it is under control of. Actuators are devices that control the physical action that is required to happen in order to form a response to the sensor's received information. The actuators receive instructions in the form of electronic digital signals from the entity that controls it and performs a desired action as a response according to the control logic generated by the domain. These sensors and actuators are essentially the "Things" in the IoT that acts as the basic access and delivery points of information that communicates to and from the human world.

3.2 Real Life Scenario

An example of where we can apply the Domain-Entity-Node (DEN) framework is to implement a Smart Apartment System, where an apartment is the domain. An apartment may consist of different kinds of rooms, such as- living room, dining room and bed room. These rooms act as entities. Each room may have different sets of sensors and controllers which serve specific purposes, such as a combination of a temperature sensor and an AC controller. Each of these sets act as a node. Suppose, a node in the living room close to the entry door consists of a motion sensor, when a person enters the apartment the motion sensor senses movement and sends data to the intelligent device (living room entity) controlling the living room. The device commands the CCTV camera in an another node to view the person, which in return sends the person's real time footage. The device processes the person's footage and identifies the individual, as well as identifies any pattern in his features that it can use. Suppose the individual is a resident of the apartment and is moving towards the sofa, the device processes the footage it views from the CCTV camera, and then commands the television controller in another node to turn on the television, if not already turned on and switch to the channel according to the individual's preferences based on the time and individual's habits that the machine already learned from data. Similarly, while entering a dining room, the device may ask the oven controller to preheat the oven depending on inputs from another CCTV camera. Assuming all rooms have a node consisting of an AC controller and a temperature sensor, the temperature sensor in each room constantly provides temperature readings to the device controlling that room, which in turn commands the AC controller to control the AC accordingly and set the temperature of the room to a comfortable level. The parent device which belongs to the apartment as a whole, leads each device in each room to process data and to make decisions as per it's directions. It is also sends processed data to the upper layer for business level decision making after receiving them from its child devices.

3.3 Implementation Scenario

Based on our proposed framework and example of would be real life scenario, we implemented it for agricultural field. We have implemented it on a small scale with raspberry pi and some sensors. In our node level we gathered our data through five types of sensors. The main tasks of these sensors are to collect temperature, humidity, soil moisture, light intensity and rainfall data. After generating the data the nodes send it to domain via entities. In the domain level we implemented our machine learning part (which discussed thoroughly in the next chapter). For now, as we're setting up the machine for the first time we have to set up the threshold value manually. After analyzing the data, domain level machine cross check the found out metrics with the threshold value. Moreover, depending on the comparison domain send some commands to the nodes via entities to follow. For example- if we find out the soil moisture level is lower than the threshold value we send commands to the nodes to turn on the water pumps. In this case, water pumps are the actuators. Details of this scenario implementation and setup diagram 5.3 has been thoroughly discussed in chapter 5.1.3.

Chapter 4

Data Preprocessing

To implement Machine Learning in the Domain level of our Domain-Entity-Node framework, we required relevant data to train each algorithm used for the results analysis, as well as for training the algorithm that we ultimately used in the Domain level after comparing performance metrics of those algorithms. As there is no widely available sensor data for the agricultural sector of Bangladesh, we had to use experimental data taken from one of the research groups of our institute. The data originally contains 3862 entries of data [10].

The generated data from sensors, consisting of 3862 entries of data, that we are using for our purpose is shown below (see Table 4.1 and 4.2)

A sample of 50 rows of that data is shown in Table 4.1, 4.2, which have eight columns. The ‘serial’ column resembled the serial of data entries whereas the ‘time’ column refers to the exact time when each row of data was collected from sensors. Moreover, the ‘temp’ column resembled atmospheric temperature and the ‘hum’ column resembled atmospheric humidity. Furthermore, the ‘co2’ and ‘co’ columns referred to the levels of Carbon Dioxide and Carbon Monoxide in the atmosphere respectively, the ‘mos (soil)’ column referred to soil moisture and the ‘o2’ column referred to Oxygen levels in the atmosphere [10].

Due to unavailability of proper actuators required to control conditions suitable for crop yield optimization we set two sample actuators using relays. These relays are being controlled by using atmospheric temperature and atmospheric humidity values present in the dataset. Before we could use those values, however, we had to check the authenticity of the data we were working with. As shown in Table 4.1, 4.2, the atmospheric temperature showed anomalous values of 7777.77 degrees Celsius which is impossible at normal conditions. Similarly, the atmospheric humidity showed values of 6666.6% [10]. Furthermore, we had no column of discrete values in this dataset that we could use for classification algorithms which would be crucial for controlling our relays. Therefore, we were required to preprocess the

serial	time	temp	hum	co2	co	mos (soil)	o2
700	9/27/2018 2:48	7777.77	6666.66	337.03	1	8	23
701	9/27/2018 2:48	7777.77	6666.66	334.68	1	8	24
702	9/27/2018 2:48	7777.77	6666.66	334.68	1	8	23
703	9/27/2018 2:48	7777.77	6666.66	334.68	1	8	23
704	9/27/2018 2:49	7777.77	6666.66	334.68	1	8	23
705	9/27/2018 2:49	7777.77	6666.66	334.68	1	8	23
706	9/27/2018 2:49	7777.77	6666.66	334.68	1	8	23
707	9/27/2018 2:49	7777.77	6666.66	334.68	1	8	24
708	9/27/2018 2:49	7777.77	6666.66	332.35	1	8	24
709	9/27/2018 2:49	7777.77	6666.66	332.35	1	8	24
710	9/27/2018 2:49	7777.77	6666.66	332.35	1	8	24
711	9/27/2018 2:49	7777.77	6666.66	320.83	1	8	24
712	9/27/2018 2:49	7777.77	6666.66	290.07	1	9	24
713	9/27/2018 2:49	7777.77	6666.66	283.76	1	8	24
714	9/27/2018 2:49	7777.77	6666.66	415.73	1	7	23
715	9/27/2018 2:49	7777.77	6666.66	440.81	1	7	23
716	9/27/2018 2:49	30.2	71.4	440.81	1	7	23
717	9/27/2018 2:49	7777.77	6666.66	469.9	1	7	23
718	9/27/2018 2:49	7777.77	6666.66	399.59	1	8	23
719	9/27/2018 2:49	7777.77	6666.66	426.75	1	7	23
720	9/27/2018 2:49	7777.77	6666.66	368.63	1	7	23
721	9/27/2018 2:49	7777.77	6666.66	368.63	1	7	23
722	9/27/2018 2:49	7777.77	6666.66	366.13	1	7	23
723	9/27/2018 2:50	7777.77	6666.66	371.14	1	7	23
724	9/27/2018 2:50	7777.77	6666.66	371.14	1	7	23
725	9/27/2018 2:50	7777.77	6666.66	376.21	1	7	23

Table 4.1 A sample of the relevant data before preprocessing(continued) [10]

726	9/27/2018 2:50	7777.77	6666.66	378.76	1	7	23
727	9/27/2018 2:50	7777.77	6666.66	378.76	1	7	23
728	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
729	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
730	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
731	9/27/2018 2:51	7777.77	6666.66	376.21	1	7	23
732	9/27/2018 2:51	7777.77	6666.66	376.21	1	7	23
733	9/27/2018 2:51	7777.77	6666.66	378.76	1	8	23
734	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
735	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
736	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
737	9/27/2018 2:51	7777.77	6666.66	378.76	1	8	23
738	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
739	9/27/2018 2:51	7777.77	6666.66	378.76	1	7	23
740	9/27/2018 2:51	7777.77	6666.66	378.76	1	8	24
741	9/27/2018 2:51	7777.77	6666.66	381.32	1	8	23
742	9/27/2018 2:51	30.1	62.9	376.21	1	8	23
743	9/27/2018 2:51	27.8	66.4	376.21	1	8	24
744	9/27/2018 2:51	27.7	64.5	373.67	1	8	24
745	9/27/2018 2:51	27.7	64.5	373.67	1	8	24
746	9/27/2018 2:51	27.6	64.6	376.21	1	8	24
747	9/27/2018 2:51	27.5	65	376.21	1	8	24
748	9/27/2018 2:52	27.5	69.1	378.76	1	8	24
749	9/27/2018 2:52	27.4	65.4	376.21	1	8	23

Table 4.2 A sample of the relevant data before preprocessing [10]

data in order to get it ready for implementing Machine Learning, specifically classification algorithms, for our purpose.

Data preprocessing is simply a process of getting data ready before training a Machine Learning algorithm. This is very important as not doing so might cause the algorithm to be over-fitted or under-fitted and may cause unnecessary noise in the algorithm that would hamper its ability to predict values correctly. We preprocessed the data that we used for our purpose in two simple steps.

4.1 Data cleaning

Before we could use the atmospheric temperature and atmospheric humidity values required to control our relay switches, we required to remove the anomalous values mentioned previously. Therefore, we replaced those anomalous values with NaN (not-a-number) values which are treated as null values and then simply dropped the rows with those NaN values, which is easier to do than dropping rows with anomalous values directly.

A sample of 50 rows of the clean dataset is shown in Table 4.3, 4.4. It clearly suggests no anomalous or extremely abnormal values amongst both the atmospheric temperature and the atmospheric humidity values. The maximum value after cleaning the dataset for both atmospheric temperature and atmospheric humidity is 38.8 degrees Celsius and 86.2% respectively.

4.2 Data Binarization

Despite the fact that we got rid of the anomalous values, we still do not have discrete values of columns required to control the relay switches. However, we did have clean atmospheric temperature and atmospheric humidity values that we could use to create our own discrete columns which would later be required to train the algorithm to give it the knowledge of turning each relay switch on or off. Therefore, we binarized the columns that stored atmospheric temperature and atmospheric humidity, and saved the results in two separate columns. The base value for atmospheric temperature was set to 30 degrees Celsius above which the value should be 1. Otherwise, the value should be 0. Similarly, the base value for atmospheric humidity was set to 60% below which the value should be 1. Otherwise, the value should be 0. The value 1 refers to a relay switch being turned on, and the value 0 refers to a relay switch being turned off.

Serial	Time	temp	hum	co2	co	mos (soil)	o2
0	9/27/2018 2:49	30.2	71.4	440.81	1	7	23
1	9/27/2018 2:51	30.1	62.9	376.21	1	8	23
2	9/27/2018 2:51	27.8	66.4	376.21	1	8	24
3	9/27/2018 2:51	27.7	64.5	373.67	1	8	24
4	9/27/2018 2:51	27.7	64.5	373.67	1	8	24
5	9/27/2018 2:51	27.6	64.6	376.21	1	8	24
6	9/27/2018 2:51	27.5	65	376.21	1	8	24
7	9/27/2018 2:52	27.5	69.1	378.76	1	8	24
8	9/27/2018 2:52	27.4	65.4	376.21	1	8	23
9	9/27/2018 2:52	27.4	69.6	376.21	1	8	24
10	9/27/2018 2:52	27.4	66.6	373.67	1	8	24
11	9/27/2018 2:52	27.3	66.6	376.21	1	8	24
12	9/27/2018 2:52	27.2	65.3	376.21	1	8	24
13	9/27/2018 2:52	27.4	68.1	378.76	1	8	24
14	9/27/2018 2:52	27	65.7	373.67	1	8	24
15	9/27/2018 2:52	26.9	68.1	371.14	1	7	24
16	9/27/2018 2:52	26.8	66.3	373.67	1	8	24
17	9/27/2018 2:52	26.9	67.3	368.63	1	8	24
18	9/27/2018 2:52	26.6	65.9	366.13	1	8	24
19	9/27/2018 2:52	26.6	72.3	366.13	1	7	24
20	9/27/2018 2:52	26.5	69.9	368.63	1	8	24
21	9/27/2018 2:52	26.5	68.2	368.63	1	8	24
22	9/27/2018 2:52	26.5	74.6	363.64	1	8	24
23	9/27/2018 2:52	26.4	75.9	366.13	1	8	24
24	9/27/2018 2:52	26.3	72.8	358.7	1	8	24
25	9/27/2018 2:52	26.2	74.9	358.7	1	8	24

Table 4.3 A sample of the relevant data after cleaning (continued)

26	9/27/2018 2:52	26.2	75.5	353.8	1	8	24
27	9/27/2018 2:52	26.4	74.2	353.8	1	8	24
28	9/27/2018 2:52	26.1	74.4	356.24	1	8	24
29	9/27/2018 2:52	25.9	75.8	351.37	1	8	24
30	9/27/2018 2:52	25.9	76.8	353.8	1	8	24
31	9/27/2018 2:52	25.9	74.1	353.8	1	8	24
32	9/27/2018 2:52	25.8	73.8	356.24	1	8	24
33	9/27/2018 2:52	25.7	70.2	469.9	1	7	22
34	9/27/2018 2:52	26.1	75.8	368.63	1	7	23
35	9/28/2018 12:10	31.1	74.6	452.28	1	7	24
36	9/28/2018 12:11	31.1	74.6	455.19	1	8	24
37	9/28/2018 12:11	31.1	74.6	458.1	1	8	24
38	9/28/2018 12:11	31.1	74.5	455.19	1	8	24
39	9/28/2018 12:11	31.1	74.6	455.19	1	8	24
40	9/28/2018 12:11	31.1	74.6	455.19	1	8	24
41	9/28/2018 12:11	31.1	74.6	455.19	1	8	24
42	9/28/2018 12:11	31.1	74.6	455.19	1	8	24
43	9/28/2018 12:11	31.1	74.6	458.1	1	8	24
44	9/28/2018 12:11	31.1	74.6	458.1	1	8	24
45	9/28/2018 12:11	31.2	74.7	455.19	1	8	24
46	9/28/2018 12:12	31.1	74.6	458.1	1	8	24
47	9/28/2018 12:12	31.1	74.6	449.4	1	8	25
48	9/28/2018 12:12	31.2	74.7	446.52	1	8	25
49	9/28/2018 12:12	31.2	74.7	551.85	2	8	23

Table 4.4 A sample of the relevant data after cleaning

Serial	time	temp	hum	co2	co	mos (soil)	o2	relay-channel1	relay-channel2
0	9/27/2018 2:49	30.2	71.4	440.81	1	7	23	0	1
1	9/27/2018 2:51	30.1	62.9	376.21	1	8	23	0	1
2	9/27/2018 2:51	27.8	66.4	376.21	1	8	24	0	0
3	9/27/2018 2:51	27.7	64.5	373.67	1	8	24	0	0
4	9/27/2018 2:51	27.7	64.5	373.67	1	8	24	0	0
5	9/27/2018 2:51	27.6	64.6	376.21	1	8	24	0	0
6	9/27/2018 2:51	27.5	65	376.21	1	8	24	0	0
7	9/27/2018 2:52	27.5	69.1	378.76	1	8	24	0	0
8	9/27/2018 2:52	27.4	65.4	376.21	1	8	23	0	0
9	9/27/2018 2:52	27.4	69.6	376.21	1	8	24	0	0
10	9/27/2018 2:52	27.4	66.6	373.67	1	8	24	0	0
11	9/27/2018 2:52	27.3	66.6	376.21	1	8	24	0	0
12	9/27/2018 2:52	27.2	65.3	376.21	1	8	24	0	0
13	9/27/2018 2:52	27.4	68.1	378.76	1	8	24	0	0
14	9/27/2018 2:52	27	65.7	373.67	1	8	24	0	0
15	9/27/2018 2:52	26.9	68.1	371.1	1	7	24	0	0
16	9/27/2018 2:52	26.8	66.3	373.67	1	8	24	0	0
17	9/27/2018 2:52	26.9	67.3	368.63	1	8	24	0	0
18	9/27/2018 2:52	26.6	65.9	366.13	1	8	24	0	0
19	9/27/2018 2:52	26.6	72.3	366.13	1	7	24	0	0
20	9/27/2018 2:52	26.5	69.9	368.63	1	8	24	0	0
21	9/27/2018 2:52	26.5	68.2	368.63	1	8	24	0	0
22	9/27/2018 2:52	26.5	74.6	363.64	1	8	24	0	0
23	9/27/2018 2:52	26.4	75.9	366.13	1	8	24	0	0
24	9/27/2018 2:52	26.3	72.8	358.7	1	8	24	0	0
25	9/27/2018 2:52	26.2	74.9	358.7	1	8	24	0	0

Table 4.5 A sample of the relevant data after binarization (continued)

26	9/27/2018 2:52	26.2	75.5	353.8	1	8	24	0	0
27	9/27/2018 2:52	26.4	74.2	353.8	1	8	24	0	0
28	9/27/2018 2:52	26.1	74.4	356.24	1	8	24	0	0
29	9/27/2018 2:52	25.9	75.8	351.37	1	8	24	0	0
30	9/27/2018 2:52	25.9	76.8	353.8	1	8	24	0	0
31	9/27/2018 2:52	25.9	74.1	353.8	1	8	24	0	0
32	9/27/2018 2:52	25.8	73.8	356.24	1	8	24	0	0
33	9/27/2018 2:52	25.7	70.2	469.9	1	7	22	0	0
34	9/27/2018 2:52	26.1	75.8	368.63	1	7	23	0	0
35	9/28/2018 12:10	31.1	74.6	452.28	1	7	24	0	1
36	9/28/2018 12:11	31.1	74.6	455.19	1	8	24	0	1
37	9/28/2018 12:11	31.1	74.6	458.1	1	8	24	0	1
38	9/28/2018 12:11	31.1	74.5	455.19	1	8	24	0	1
39	9/28/2018 12:11	31.1	74.6	455.19	1	8	24	0	1
40	9/28/2018 12:11	31.1	74.6	455.19	1	8	24	0	1
41	9/28/2018 12:11	31.1	74.6	455.19	1	8	24	0	1
42	9/28/2018 12:11	31.1	74.6	455.1	1	8	24	0	1
43	9/28/2018 12:11	31.1	74.6	458.1	1	8	24	0	1
44	9/28/2018 12:11	31.1	74.6	458.1	1	8	24	0	1
45	9/28/2018 12:11	31.2	74.7	455.19	1	8	24	0	1
46	9/28/2018 12:12	31.1	74.6	458.1	1	8	24	0	1
47	9/28/2018 12:12	31.1	74.6	449.4	1	8	25	0	1
48	9/28/2018 12:12	31.2	74.7	446.52	1	8	25	0	1
49	9/28/2018 12:12	31.2	74.7	551.85	2	8	23	0	1

Table 4.6 A sample of the relevant data after binarization

According to a sample shown in Table 4.5, and 4.6, the binarized dataset contained 2 new columns, ‘relay-channel1’ and ‘relay-channel2’, which were related to the atmospheric humidity column and the atmospheric temperature column respectively. All the other columns remain unchanged. The resulting dataset after preprocessing was suitable enough to implement Machine Learning in our Domain level for the Domain-Entity-Node framework, as well as to perform results analysis for each algorithm to select which would be the most suitable one to implement in the Domain level.

Therefore, the dataset that we have taken from one of the research groups from our institute would now be ready to be used for Machine Learning, more specifically classification algorithms. This is crucial both to implement our proposed model, which has eventually been proved to decrease latency, and to decide which classification algorithm to use by performing results analysis with the preprocessed dataset.

Chapter 5

System Implementation

5.1 Hardware Implementation

5.1.1 Components used

- Raspberry Pi 3 Model B
- Raspberry Pi Zero W
- DHT11 Temperature and Humidity Sensor Module
- YL-83 Rain Detector Board + Control Board
- YL-69 Soil Hygrometer Probe + Control Board
- ADS1115 16-bit I2C Analog-to-Digital Converter (ADC)
- Light Dependent Resistor
- 1 kΩ Resistor
- 5v 2-Channel Relay Interface Module Board
- Breadboard
- Jumper Wires

5.1.2 Tech Stack used

- Python for Data science (Sci-kit learn, Numpy, Pandas)

- Python for Raspberry Pi hardware programming and interfacing (DHT, ADS1x15 and other libraries including dependencies).
- Raspian OS
- Python for Socket Programming (Socket)
- Thingspeak as a platform for IoT data storage
- RESTful APIs for reading and writing data to Thingspeak

5.1.3 Hardware setup implementation

According to the proposed Domain-Entity-Node (DEN) framework 3.1and the implementation scenario [section 3.3], a simplistic prototype [see figure: 5.1, 5.2 and 5.3] have been developed which even though, is not as sizable yet is very closely resembling to the practical implementation of the Internet of Things into the real world. The prototype mainly consists of a computer connected with three Raspberry Pi machines through a router forming an intranet setup. The Domain computer that empowers the entire framework with predictive and machine learning capabilities is set as server. The Raspberry Pi machines have been designated as the clients that connect to the Domain and becomes the Edges in framework. Furthermore, the set of four sensors namely DHT11 Temperature and Humidity sensor, YL-83 rainfall sensor, YL-69 soil hygrometer and LDR all acts as the Nodes in the prototype's infrastructure.

Two different Raspberry Pi machines have been used in the framework to demonstrate the versatility of the framework across computers of different specifications. The Raspberry Pi 3 Model B is a third-generation board under the Raspberry Pi flagship powered by the ARM Cortex-A53 Quad Core 1.2GHz Broadcom BCM2837 64-bit CPU and 1GB RAM. Some of the best features of the board include on-board Bluetooth Low Energy (BLE) and wireless LAN, 40-pin extended General Purpose Input/Output (GPIO) [13]. On the other hand, the Raspberry Pi Zero W is single-core 1GHz Boardcom BCM2835 CPU with 512MB RAM and an on-board Bluetooth Low Energy (BLE) and a wireless LAN, together with 40-pin extended General Purpose Input/Output (GPIO) pins just like the Raspberry Pi 3 Model B [13]. The framework's capabilities may not be limited to the Raspberry Pi System-on-Chip (SoC) only and may be implemented using other SoCs and/or micro-processors and micro-controllers like ATMega328, ESP32s and WeMos boards.

Moving forward, the DHT11 temperature and humidity sensor that is used generates a calibrated digital signal that connects to a high performance 8-bit micro-controller that outputs a digital signal that is simple to read on most microprocessors. It has a measurable humidity range of 20-90 %RH ±5%RH and 0-50° C ±2% °C for temperature [18]. There are other temperature and humidity sensors like DHT22, DS18B20 for temperature and off-the-shelf hygrometers may be used in place of the DHT11. Up next, the YL-83 or its alternative FC-37 rain sensor is a sensor composed of two parts, the control board and the collector/detection board. The collection/detection board measures the relative resistance based on the amount of water that collects of the board, more water results in low resistance and less or no water results in high resistance [19]. The electronic control board reads the resistance, adjusts the sensitivity according to the on-board potentiometer and generates a digital as well as an analog signal that can be read by any micro-processor or micro-controller with analog inputs or via an analog-to-digital converter. The YL-69 or its alternative HL-69 soil hygrometer is very similar to the rain sensor, being composed of two components. The detection probe when placed into soil measures the water content within the soil column, where wetness generates a high resistance and dryness generates a low resistance [20]. The LDR is used with a 1kΩ resistor in series to generate a varied resistance corresponding to the intensity of light falling on the LDR. According to the necessity of the software side algorithm that the Domain-Entity-Node (DEN) framework has been implemented on, it is absolutely essential to be able to read the continuous analog signal generated by the array of sensors. Furthermore, Raspberry Pi being incapable of directly reading the analog signals that the sensors generate have raised a necessity for an analog-to-digital(ADC) being interfaced within the prototype's structure. To serve the purpose, ADS1115 16-bit I2C ADC is being used. The ADS1115 is a very compact and low power chip that is designed to have 4 single input channels and is compatible with the I2C [9]. The Inter-Integrated Circuit or I2C is a high speed communication protocol between not only 2 ICs on the same board but also components that are compatible. It features a true master-slave relationship with the devices interconnected over the two I2C buses, Serial Data (SDA) and Serial Clock (SCL). The analog output of the rainfall sensor, soil moisture sensor and LDR are connected on three separate input channels of the ADS1115 ADC which is then connected using the SDA and SCL buses to the Raspberry Pi.All the components are powered using the two 5v power pins and several Ground(GND) pins of the Raspberry Pi spread across a breadboard. The DHT11 sensor's data output is connected to one of the GPIO of the Raspberry Pi.

Sensors are just half the life of an Internet of Things setup. Without devices that cause a change in the environment, IoT would only be able to sense and perceive the surroundings and be limited to that only. Effectors or actuators come into play here completing the IoT picture. For our scenario's demonstrative purpose, a two channel 5v relay module is used that can control and switch devices within a range of 110v to 240v and can handle currents

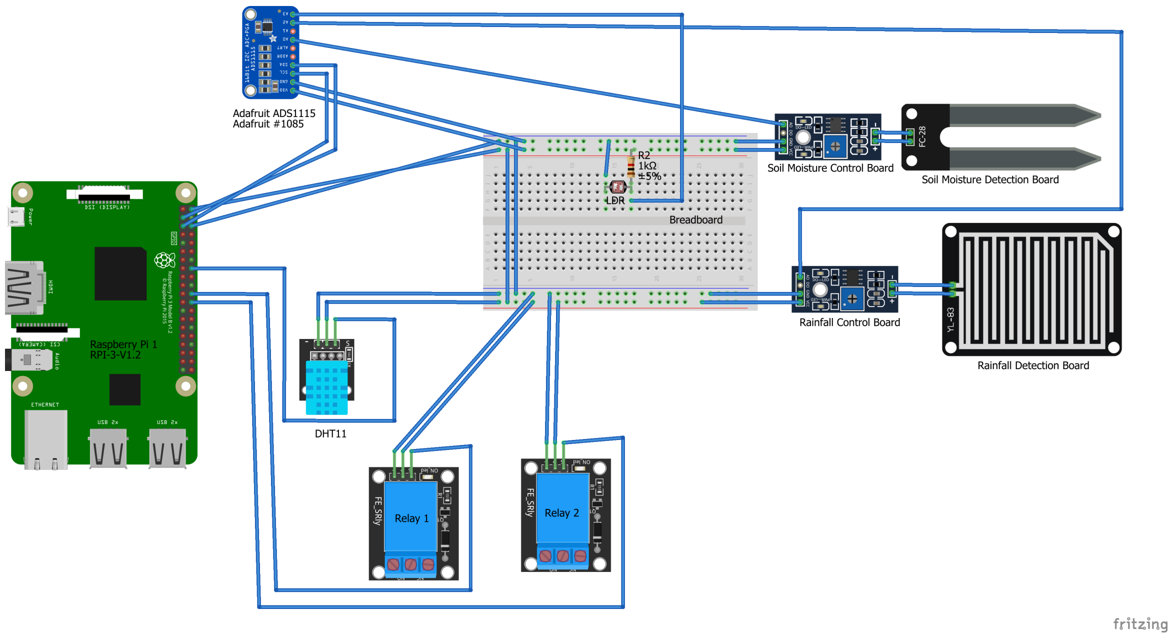

Fig. 5.1 The circuit diagram of an Edge and its corresponding Nodes

up to 10A. The two separate channel inputs are connected to the Raspberry Pi's two GPIO pins that are programmed to perform as digital output pins. With an abundance of free GPIO pins in the Raspberry Pi, the use of a two channel relay is not limited to that amount only. It can quite easily accommodate a sixteen channel relay for a single Raspberry Pi. Therefore, a single Edge can, theoretically, easily be able to handle, control and switch up to sixteen different electronic devices each connected to a single relay input. A demonstrative circuit diagram featuring all the hardware components and their corresponding wired connections is included.

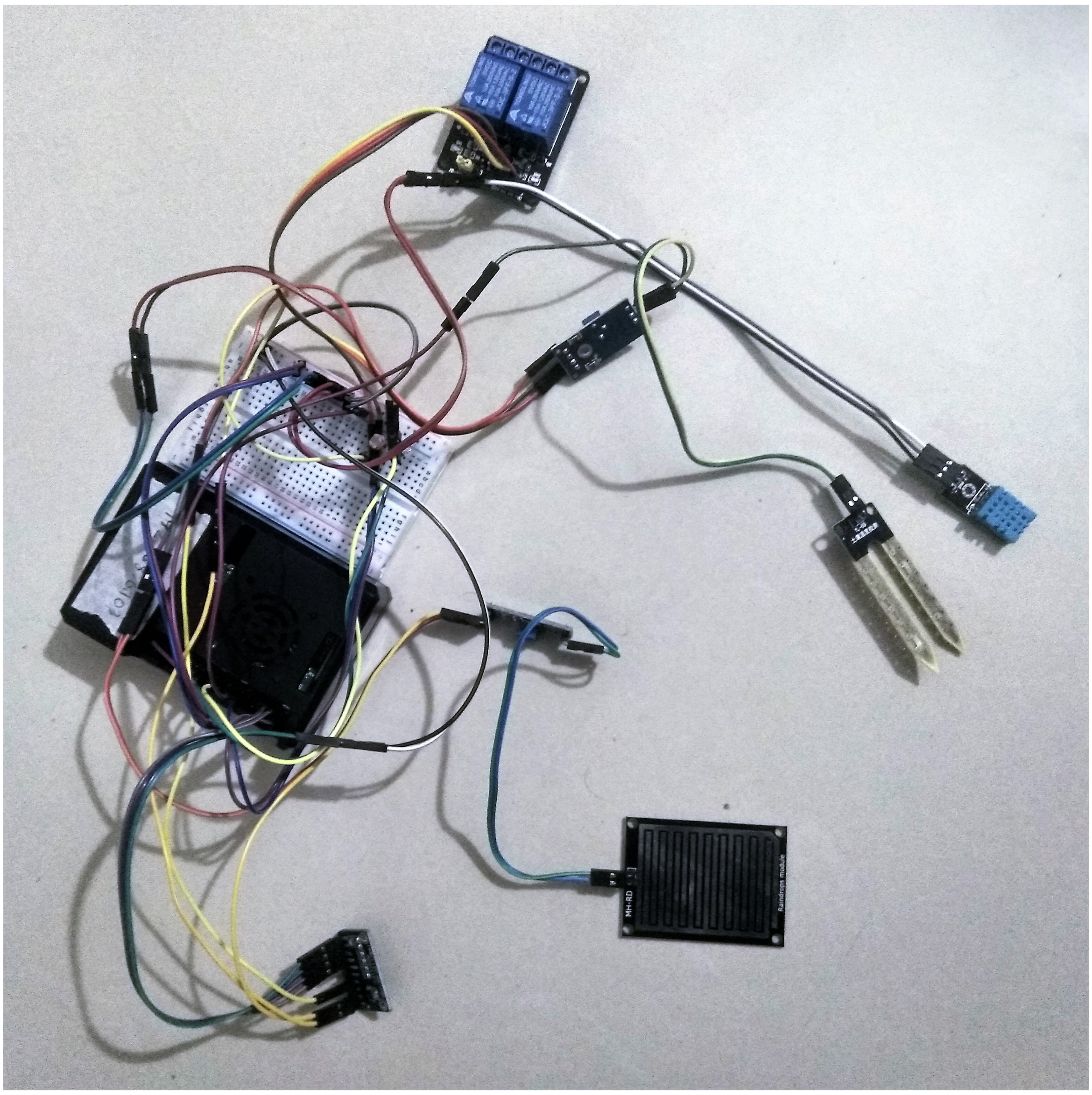

Fig. 5.2 Single Entity-Node Prototype

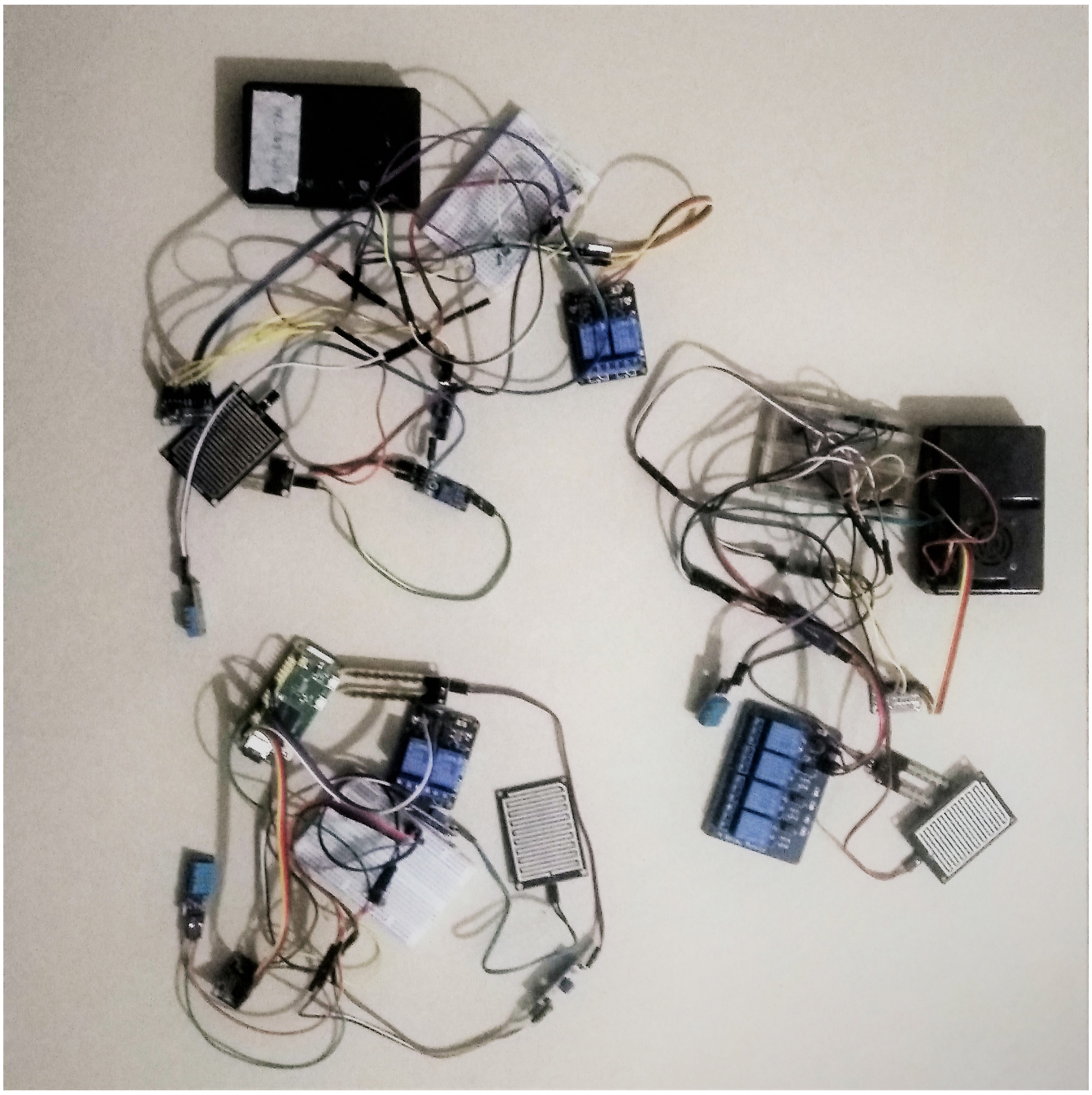

Fig. 5.3 A Cluster of Multiple Entity-Node Prototypes

5.1.4 Hardware Interfacing and Programming

The Domain-Entity-Node (DEN)framework requires all the machines within the framework to be connected in a Local Area Network(LAN). Specifically, the Domain machine requires to be connected with the Edges in a LAN and be able to establish stable wireless communication between each other. One way of establishing stable wireless communication between two or more devices is by employing socket programming, which in our case, suits due to the intranet connectivity. The Domain is programmed as a socket server that is capable of serving multiple clients' requests and deliver their corresponding responses. While, the Edges are programmed to act as light-weight socket clients that each connects to the Domain's server. The socket program communicates using the Transmission Control Protocol (TCP) for a consistent and lossless transmission of data. The Domain machine runs the machine learning algorithms and predicts the corresponding best response for the clients' requests. When the Domain server is run, it waits for the client to send the live data which serves as a request. The Edge client, when run, reads live data from the array of sensors, stores the data in a local Comma Separated Values (CSV) file, as well as send the data to the domain server using the established socket connection. Upon receiving the data from the client Edges, the Domain server predicts the best response based on the received data and sends the response back to the corresponding clients. The response being predicted are binary values each corresponding to the relay channels that control the physical devices, Nodes, that causes a change in the environment. The response is then used by the client to switch on or off the relay channels that the responses relatively corresponds to.

5.2 Machine Learning Implementation

Our Domain-Entity-Node framework works in a way such that the Entity level, which in our case consists of each Raspberry Pi in our implementation, takes in data from sensors at the Node level and sends that data to the Domain level, which in our case is the laptop we used to implement our framework. In the Domain level, data is analyzed and, consequently, commands in discrete form are sent back to the Entity level, in which those discrete values are used to control actuators at the Node level. In order to be able to send commands, the Domain level must have a device that knows the appropriate thing to do at a certain condition. For example, if the temperature is too high, it must be able to decide that the air conditioner must be turned on to lower the temperature before it actually relays that information to the entity. For the device to be able to know, it must be trained with facts and the answers that must come with these facts, so that when a certain time comes during which we decide that it has been trained enough, it will be able to do the right thing at a certain condition. Therefore,

	ON	OFF
Relay Switch 1:	1	0
Relay Switch 2:	1	0

Table 5.1 Discrete values resembling commands used for relay switches.

we trained it using Machine Learning to be able to do just that. This is crucial to our bid in achieving the learning phase in the Internet of Things for agriculture since enabling the device within the Domain to make decisions of its own based on the data it is receiving allows the system to be automated. As we specified before, the commands come in discrete values. These discrete values are derived from continuous values within the sensor data that the devices in the Entity level send to the device in the Domain level. Therefore, we used a classification algorithm to train the device in our Domain level which derives discrete data from continuous data, as opposed to regression algorithms which derive continuous data from other continuous data. The classification algorithm we have specifically chosen to train the device within the Domain level is the Decision Tree, after analyzing performance metrics of five classification algorithms including the algorithm we have chosen for our model (see Chapter 6.1). The device in the Domain level is trained using the preprocessed dataset (see Chapter 4) and the Decision Tree algorithm. The device then uses that algorithm to predict when the relay switches should be turned on and when they should be turned off according to the atmospheric temperature and humidity values that the devices from the Entity level sends to it. The values resembling commands for each switch are given in Table 5.1 below.

Chapter 6

Analysis and Findings

6.1 Results Analysis

6.1.1 Confusion Matrix

As previously specified, two confusion matrices for each model have been calculated (see Chapter 1.3). The first confusion matrix of each model is derived from the prediction of discrete values of 1 and 0 used for the relay switch that relies on atmospheric humidity (see Figures 6.1, 6.3, 6.5, 6.7 and 6.9). The second confusion matrix of each model is derived from the prediction of discrete values of 1 and 0 used for the relay switch that relies on atmospheric temperature ((see Figures 6.2, 6.4, 6.6, 6.8 and 6.10). The confusion matrix that we desired for our purpose must have no false positive or false negative values, no matter the number of true positive or true negative values there is, since any of those false positive or false negative values would result in the relay switch (see Chapter 5) being turned off when it should be actually be turned on and vice versa (see Chapter 1.3).

According to Figure 6.1 Support Vector Classification has 506 true positive values and 594 true negative values without having any false positive or false negative values. This suggests that the model will turn on the relay switch, which relies on atmospheric humidity (see Chapter 5), only when it should be turned on, and will turn off the relay switch only when it should be turned off, as per our desired confusion matrix (see Chapter 1.3).

However, according to Figure 6.2 Support Vector Classification has 7 true positive values and 1092 true negative values, with 1 false positive value and no false negative values. The false positive value suggests that the relay switch, which is dependent on atmospheric temperature, would be turned off at a temperature reading for which it actually should be turned on. This is detrimental to our purpose as any false negative or false positive values

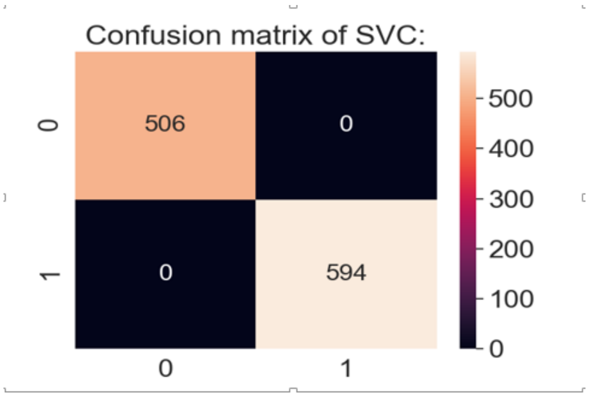

Fig. 6.1 Confusion matrix of Support Vector Classification regarding the relay switch which depends on atmospheric humidity

would mislead the relay switch into turning it on when it should be turned off, and vice versa (see Chapter 1.3).

According to Figure 6.3, Logistic Regression ended up with 493 true positive values and 594 true negative values, with 0 false negative values. However, it also came with 13 false positive values. This suggests that the relay, which is dependent on atmospheric humidity, would be turned off for the 13 humidity readings corresponding to these false positive values, during which the relay should actually be turned on. This is, as specified before, not suitable for our cause due to any false positive or false negative values leading the switch to be turned on or off at undesirable times (see Chapter 1.3).

Similarly, Figure 6.4 shows that Logistic Regression ended up with 8 false positive values and 1092 true negative values without any true positive or false negative values. This suggests that the relay switch, which depends on atmospheric temperature readings, would be turned off for the readings corresponding to those 8 false positive values when it should actually be turned on, which is undesirable (see Chapter 1.3).

The K-Nearest Neighbours algorithm, as shown in Figure 6.5, does not have any false positive or false negative values along with the 506 true positive and 594 true negative values. This, as specified before, is desirable for our purpose en it is required to be turned on, and would be turned off only if required to be turned off (see Chapter 1.3).

However, according to Figure 6.6, 1 false positive value is shown to appear for K-Nearest Neighbours along with 7 true positive values, 1092 true negative values and no false negative value. This false positive value would cause the relay switch, dependent on atmospheric temperature, to be turned off for a temperature reading during which it should actually be turned on, which is undesirable. As previously stated, any false positive or false negative

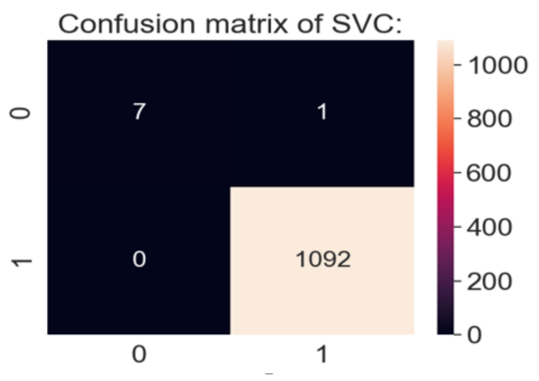

Fig. 6.2 Confusion matrix of Support Vector Classification regarding the relay switch which depends on atmospheric temperature

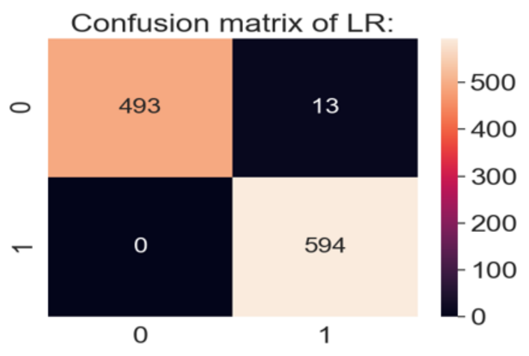

Fig. 6.3 Confusion matrix of Logistic Regression regarding the relay switch which depends on atmospheric humidity

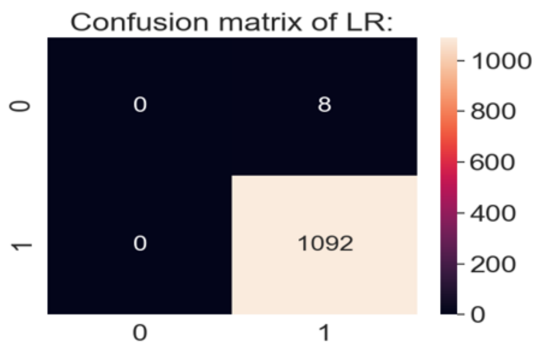

Fig. 6.4 Confusion matrix of Logistic Regression regarding the relay switch which depends on atmospheric temperature

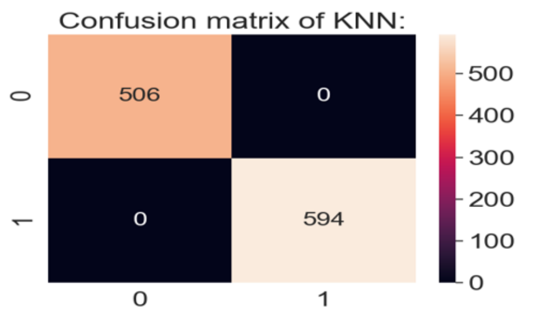

Fig. 6.5 Confusion matrix of K-Nearest Neighbours regarding the relay switch which depends on atmospheric humidity

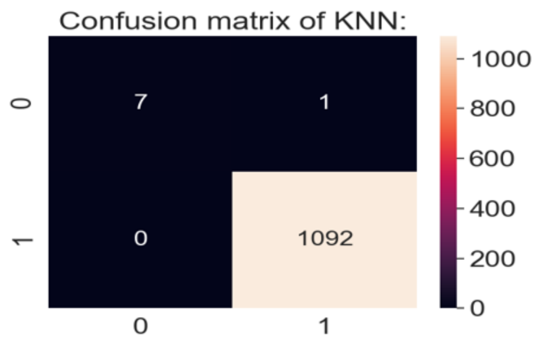

Fig. 6.6 Confusion matrix of K-Nearest Neighbours regarding the relay switch which depends on atmospheric temperature

values would mislead the relay switch into being turned off when it should be turned on, and vice versa (see Chapter 1.3).

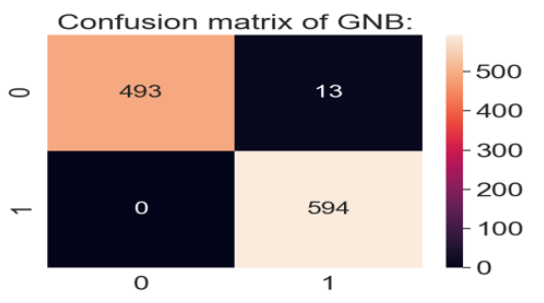

Fig. 6.7 Confusion matrix of Gaussian Naïve Bayes regarding the relay switch which depends on atmospheric humidity

Figure 6.7 suggests that Gaussian Naïve Bayes resulted in 13 false positive values along with 493 true positive values, 594 false positive values and no false negative values. These 13 false positive values would mislead the relay switch, dependent on atmospheric humidity, into turning off when it should be turned on (see Chapter 1.3). Hence, this is undesirable.

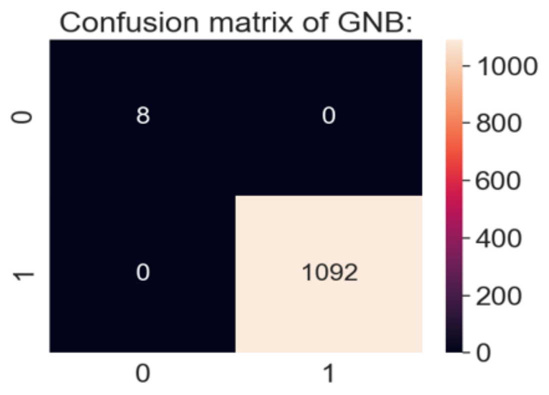

Fig. 6.8 Confusion matrix of Gaussian Naïve Bayes regarding the relay switch which depends on atmospheric temperature

However, according to Figure 6.8, there are no false positive or true positive values seen for this confusion matrix of Gaussian Naïve Bayes. This suggests that the relay switch which is dependent on atmospheric temperature would be turned on exactly when it should be turned on, and vice versa (see Chapter 1.3).

According to Figure 6.9, Decision Tree shows no false positive or false negative values with the 506 true positive values and 594 true negative values for the relay switch which is dependent on atmospheric humidity. This suggests that the relay switch would be turned on only when it should be turned on and would be turned off only when it should be turned off (see Chapter 1.3).

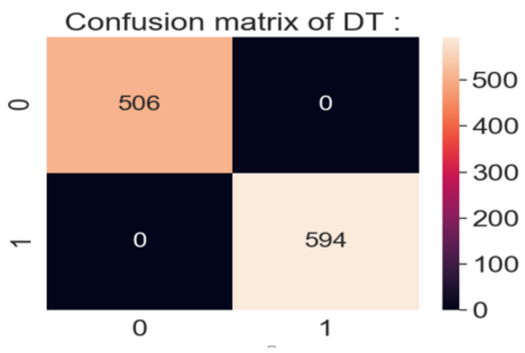

Fig. 6.9 Confusion matrix of Decision Tree regarding the relay switch which depends on atmospheric humidity

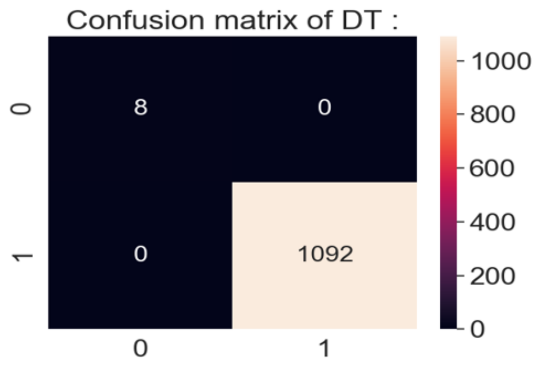

Fig. 6.10 Confusion matrix of Decision Tree regarding the relay switch which depends on atmospheric temperature

	Accuracy	Precision	Recall	Specificity	F1-Score
K-Nearest Neighbours	100%	100%	100%	100%	100%
Gaussian Naïve Bayes	98.8%	97.6%	100%	97.8%	98.4%
Logistic Regression	98.8%	97.6%	100%	97.8%	98.4%
Support Vector Classification	100%	100%	100%	100%	100%
Decision Tree	100%	100%	100%	100%	100%

Table 6.1 Performance metrics of the five classification algorithms used regarding the relay switch that is dependent on atmospheric humidity

Furthermore, according to Figure 6.10, the confusion matrix of Decision Tree for the relay switch which is dependent on atmospheric temperature also has no false positive or false negative values along with the 8 true positive values and 1092 true negative values. This implies that this relay switch is also turned on only when it should be turned on and turned off only when it should be turned off (see Chapter 1.3).

Conclusively, the results suggest that each model has no false negative values associated with each of their confusion matrices. However, each model, other than Decision Tree, has false positive values associated with one or both of their confusion matrices. Due to Decision Tree having no false positive or false negative values in either of its matrices, it is the ideal algorithm for our purpose. This is strongly justified by our explanation of the confusion matrix (see Chapter 1.3), and the performance metrics derived from each of the confusion matrices shown in the next section (see Chapter 6.1.2).

6.1.2 Metrics Derived From Confusion Matrix

From the two confusion matrices for each model (see Chapter 6.1.1), we derived two sets of metrics of each model aimed at measuring the performances of the models and comparing them to see which ones perform as ideally as possible for our purpose. Each set of metrics has the following features: accuracy, precision, recall, specificity and F1-score (see Chapter 1.3). A completely ideal classification algorithm would have scores of 100% for all of these performance metrics. This is because the algorithm must be able to predict each value correctly, must be able to predict all positive values only when the values are positive and all negative values only when the values are negative. As specified previously, a positive value would cause the relay to turn on and a negative value would do the opposite (see Chapter 1.3).

Table 6.1 shows performance metrics for each algorithm regarding the relay switch that is controlled by using atmospheric humidity readings. According to that, K-Nearest Neighbours, Support Vector Classification and Decision Tree all have 100% scores in accuracy, precision,

	Accuracy	Precision	Recall	Specificity	F1-Score
K-Nearest Neighbours	99.9%	87.5%	100%	99.9%	93.3%
Gaussian Naïve Bayes	100%	100%	100%	100%	100%
Logistic Regression	99.2%	0%	100%	99.2%	0%
Support Vector Classification	99.9%	87.5%	100%	99.9%	93.3%
Decision Tree	100%	100%	100%	100%	100%

Table 6.2 Performance metrics of the five classification algorithms used regarding the relay switch that is dependent on atmospheric humidity

recall, specificity and F1-score. This suggests that these algorithms made decisions on the relay switch based on atmospheric humidity at correct humidity readings, made the decision to turn on the switch only when it should have turned on the switch and made the decision to turn off the switch only when it should have been turned off. The other algorithms fell short on their accuracy and precision metrics, which suggest that at one or more points, they made the decision to turn on the switch when it should have been turned off (see Chapter 1.3).

However, Table 6.2, which shows performance metrics for each algorithm based on the relay switch which is controlled using atmospheric temperature readings, suggest different outcomes for each algorithm except the Decision Tree, which retains its scores of 100% in all five metrics. Logistic Regression failed in predicting any instances when it should have turned on the switch, whereas Support Vector Classification and K-Nearest Neighbours algorithms fell short on precision, causing one or more decisions where the switch was turned off when it should have been turned on. Gaussian Naïve Bayes and Decision Tree successfully predicted all correct decisions, made the correct decision to turn on the switch when it should have been turned on and the decision to turn it off when it should have been turned off (see Chapter 1.3).

Conclusively, Decision Tree has performed ideally in both situations, as it scored 100% in all metrics for both the relay switch that depends on atmospheric humidity, and for the relay switch that depends on atmospheric temperature. This justifies our decision in choosing the Decision Tree to be implemented on the Domain level of the Domain-Entity-Node framework that we proposed as a learning phase in the Internet of Things for agriculture in this case. As specified before, Machine Learning, especially classification algorithms, are crucial in our bid to achieve the learning phase in the Internet of Things, where data can be handled between devices automatically as well as effectively. By implementing the correct classification algorithm to the Domain level of our Domain-Entity-Node framework, we can automate decision making and feedback generation from the Domain level to the devices within the Entity level so that they can control their actuators automatically. Therefore, no human intervention is required, which is not the case in the traditional Internet of Things.

Furthermore, our framework is not limited to just our case, where we used agricultural data to illustrate how our framework could be used in that scenario. For any case, the system would first decide which algorithm to implement by analyzing and comparing performance metrics of several classification algorithms used for data relevant to that case before implementing the correct algorithm in our Domain-Entity-Node (DEN) Framework.

6.2 Model Performance

6.2.1 Performance of the Domain-Entity-Node (DEN) Framework

In order to judge the performance of the proposed model of the Domain-Entity-Node (DEN) framework, two test cases were established to generate comparable performance data. In the first case, the traditional IoT concept of sending and receiving data from/to sensors and effectors respectively through a cloud IoT platform have been programmed and the time taken or latency of the response being put into action is measured within the programme. In the second test case, the prototype of the proposed model of the Domain-Entity-Node (DEN) framework have been timed within the programme and the time taken from the sensors to generate the data to the time taken for the effector to get the response back is measured.

Test Case 1:
The data is read from the sensors by the Raspberry Pi and then written to a Thingspeak channel. Another computer reads data from that channel and performs a prediction on the aforementioned data, the predicted values are then written to another Thingspeak channel. The Raspberry pi then reads the predicted values from the channel and operates the effectors accordingly (see figure: 6.11).

Test Case 2:
The data is read from the sensors by the Edge Raspberry Pi and then sent to the Domain computer via TCP communication. The Domain socket server receives data from the socket client and performs a prediction on the aforementioned data, the predicted values are then sent back to the Edge on the client programme. The Edge then generates a control logic based on the received predicted values from the channel and operates the effectors accordingly (see Figure: 6.12).

Performance Comparison: (see table: 6.3)
As per US patented methods and apparatus for measuring a network performance, one of the aforementioned methods is the usage of the programme 'ping' which employs the difference of time measured in between transmission of a controlled set of data communication traffic,

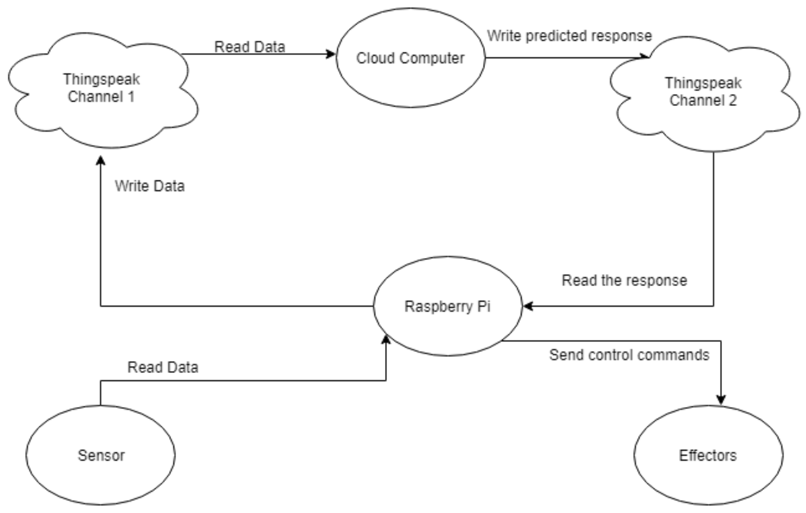

Fig. 6.11 Data-flow diagram of the test case 1

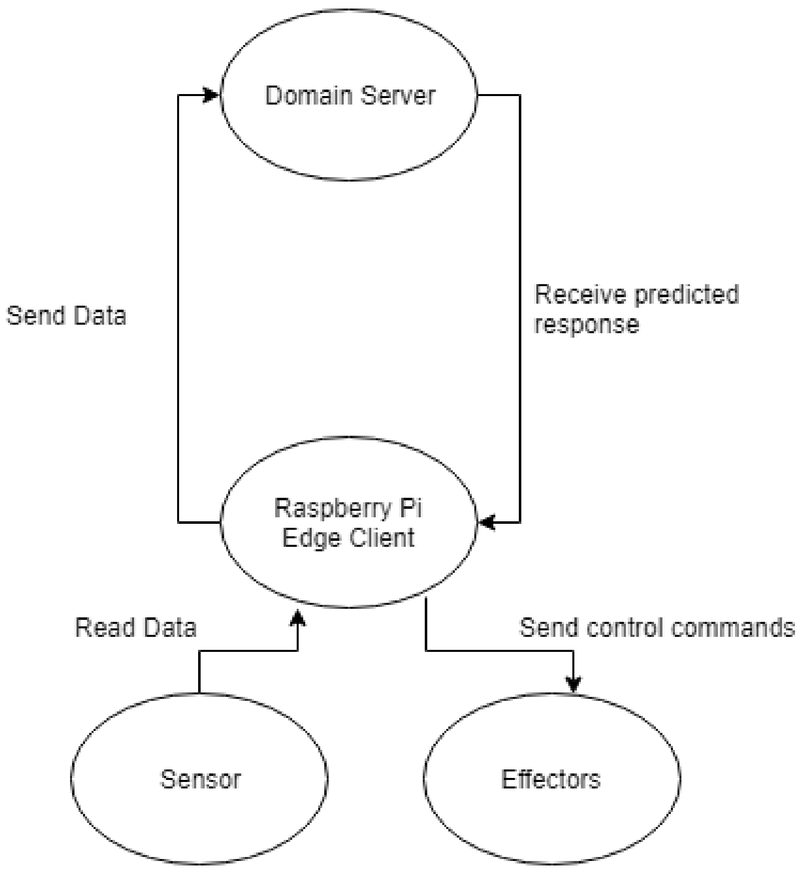

Fig. 6.12 Data-flow diagram of the test case 2

Trip no.	Average Latency (in seconds)	
	Test Case 1	Test Case 2
1	2.714	0.073
2	2.686	0.011
3	2.564	0.010
4	2.660	0.012
5	2.617	0.010
6	3.555	0.010
7	2.629	0.016
8	2.648	0.012
9	2.626	0.017

Table 6.3 Average latency comparison between the two test cases

for example Internet Control Message Protocol (ICMP) ping packets, from a source computer to a destination computer resulting in the time taken for the entire round-trip [5]. Thus, concluding with network latency or the time taken for a set of data to be sent and received as a measure for network performance, our test cases generated the following average latency report shown in table 6.3.

Chapter 7

Limitations and Future Works

7.1 Limitations

During our research and implementation of the Domain-Entity-Node framework for the Internet of Things, we faced various difficulties and limitations. Firstly, due to lack of sensors, sensor data, data regarding ideal crop conditions and yield data, there were a lack of features used to derive decisions of the domain. The Raspberry PI's hardware components had outdated libraries which affected the optimization of edge level components. Initially, we wanted to program cognitive computing into our proposed model during our research phase. However, lack of open-source cognitive libraries forced us to resort to implementing Machine Learning instead. Moreover, there is no practical cognitive computing done in the non-commercial level. There is also no generalized or universal acceptance of the Internet of Things which makes it very difficult to compare research results of our proposed model with a fairly justified one.

7.2 Future Works

According to our proposed model, the Domain-Entity-Node Framework, the lowest level of a simplistic conventional model of the Internet of Things (IoT) which consisted solely of the "Things" that connect individually to the internet have been split into three different layers that solve the various problems that surfaced in the design of the conventional IoT. However, an array of unsolved problems still remain in the designs of the higher levels constituted in the structure of an IoT network. Coming from the bottom up and as a future endeavour, the communication layer, data application layer and the business analysis layers can be further developed to pave way for the arrival and the thriving of a self sustaining cognitive system (see figure 3.1). The possible development of more advanced, secured, reliable and faster IoT communication protocols will prove to lower the latencies of larger network structures and provide a more dynamic and widespread scalability. Resulting in more manageable network structures, the future improvements in the communication layer of the Internet of Things (IoT) will thus make our proposed model even faster and may even possibly make the framework work in true real-time.

We have seen that the problem of generating exponentially increasing mass data as more devices are connected to the Internet of Things (IoT) network, have emerged in the development of the mainstream IoT and that our proposed Domain-Entity-Nodes (DEN) framework has been able to address and solve the problem of processing and handling of the mass data generated by the "Things" even before it could reach the Internet by decentralizing the data accumulation and data analysis within the closer layers of the "Things". As a result, the internet traffic can now be less of an handling issue and more focus can be given in the security of the data as well as further in-detail and deeper analysis of the data en masse. Larger and more robust analysis can be done further up in the Data Application layer of the Domain-Entity-Nodes (DEN) framework (see figure 3.1). Furthermore, the results of the Data Application layer can be used further up in the Business Implementation layer to help the real world in economy, technology, health, marketing and retail, agriculture, and more. The handled data as well as the advanced analysis results can be used in the further development of specially designed Platform as a Service (PaaS) for the handling of the Internet of Things. If and when, cognition is reached in the Internet of Things, the storage of the mass data will also be needed to be handled with utmost efficiency and possibly lead to the further research of a cognitive cloud storage system.

If better and a wider range of sensors and actuators are developed in the future, we look forward to using them to develop a system that better serves the purpose of the Domain-Entity-Node (DEN) framework. If any non-commercial or open-source cognitive programming is available in the near future, we hope to test it out for our domain level in our purpose to achieve a robust and successful Internet of Things that empowers and automates the processes involved in agriculture and crop production. Development of quantum computers or powerful processors will be instrumental in developing a more robust system for the Internet of Things in the domain of agriculture and crop production.

Chapter 8

Conclusion

The Internet of Things is currently a giant that is lying dormant. To awaken its true self, we must empower these “things” with learning abilities to tackle the large sea of information generated within the network. In this paper we proposed and implemented our framework to transform the IoT experience into a more advanced and manageable one. Moreover, we took help from the powers of machine learning to empower and transform simple things connected in a mass network into an intelligent web of controlled things that would prove to reduce network latency and create a faster communication between the "things" and the machines that are involved in learning and making decisions. We hope to conduct further research on better ways to improve the speed with which intelligent systems can learn so that IoT systems can be awakened faster, and to improve security measures for these systems.

References

[1] Alansari, Z., Anuar, N. B., Kamsin, A., Soomro, S., Belgaum, M. R., Miraz, M. H., and Alshaer, J. (2018). Challenges of internet of things and big data integration. In *International Conference for Emerging Technologies in Computing*, pages 47–55. Springer.

[2] Ashton, K. et al. (2009). That 'internet of things' thing. *RFID journal*, 22(7):97–114.

[3] Carvalho, H. F. and Kim, Y.-t. (2018). Implementation of iot (internet of things) and automation systems to control temperature and humidity in home-style greenhouse. , 2018:306–306.

[4] Díaz, M., Martín, C., and Rubio, B. (2016). State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. *Journal of Network and Computer applications*, 67:99–117.

[5] Guo, Q. (2008). Methods and apparatus for measuring network performance. US Patent 7,457,868.

[6] Harrell, F. E. (2015). Ordinal logistic regression. In *Regression modeling strategies*, pages 311–325. Springer.

[7] Holmes, G., Donkin, A., and Witten, I. H. (1994). Weka: A machine learning workbench. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361. IEEE.

[8] Hu, S., Wang, H., She, C., and Wang, J. (2010). Agont: ontology for agriculture internet of things. In *International Conference on Computer and Computing Technologies in Agriculture*, pages 131–137. Springer.

[9] Instruments, T. (2016). Ads111x ultra-small, low-power, i 2c-compatible, 860-sps, 16-bit adcs with internal reference, oscillator, and programmable comparator. *ADS1113, ADS1114, ADS1115 Datasheet, Texas Instruments*.

[10] Kamruzzaman, S. M., Pavel, M. I., Hoque, M. A., and Sabuj, S. R. (2018). *Promoting Greenness with IoT based Plant Growth System*. EAI/Springer Innovations.

[11] Kelly, J. E. (2015). Computing, cognition and the future of knowing. *Whitepaper, IBM Reseach*, page 2.

[12] Kesby, R. (2012). How the world's first webcam made a coffee pot famous'. *BBC News*, 22.

[13] McManus, S. and Cook, M. (2017). *Raspberry Pi for dummies*. John Wiley & Sons.

[14] MEYER, B. (2015). Case studies. In *Researching Translation and Interpreting*, pages 195–202. Routledge.

[15] Mezghani, E., Exposito, E., and Drira, K. (2017). A model-driven methodology for the design of autonomic and cognitive iot-based systems: application to healthcare. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(3):224–234.

[16] Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

[17] Ray, S., Jain, K., Blog, G., and Saraswat, M. (2016). Understanding support vector machine algorithm from examples (along with code). *Analytics Vidhya.*

[18] Robotics, D. (2010). Dht11 humidity & temperature sensor.

[19] Santos, S. (2018). Guide for rain sensor fc-37 or yl-83 with arduino.

[20] Sara Santos, R. S. (2018). Guide for soil moisture sensor yl-69 or hl-69 with the arduino.

[21] Srivastava, T. (2014). Introduction to k-nearest neighbors: Simplified. *Analytics Vidhya*, 10.

[22] Stojkoska, B. L. R. and Trivodaliev, K. V. (2017). A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454–1464.

[23] Sunasra, M. (2017). Performance metrics for classification problems in machine learning. *medium.com.* [online] https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b.

[24] Tung, L. (2017). Iot devices will outnumber the world's population this year for the first time.

[25] Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., and Long, K. (2014). Cognitive internet of things: a new paradigm beyond connection. *IEEE Internet of Things Journal*, 1(2):129–143.

[26] Zeinab, K. A. M. and Elmustafa, S. A. A. (2017). Internet of things applications, challenges and related future technologies. *World Scientific News*, 2(67):126–148.