2017

A Model for Anomalies Detection in Internet of Things (IoT) Using Inverse Weight Clustering and Decision Tree

Ahod Alghuried
Technological University Dublin

A Model for Anomalies Detection in Internet of Things (IoT) Using Inverse Weight Clustering and Decision Tree

Ahod Alghuried

D15123616

A dissertation submitted in partial fulfilment of the requirements of

Dublin Institute of Technology for the degree of

MSc. in Computing (Security and Forensics)

2017

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Security and Forensics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: ahod

Ahod Alghuried

Date: 03rd January 2017

ABSTRACT

Internet of Things (IoT) is one of the fast growing technologies today. It is a technology by which billions of smart objects or devices known as “Things” can use several types of sensors to collect various types of data about themselves and/or the surrounding environment. They can then share this with authorized parties to serve several purposes such as controlling and monitoring industrial facilities or improving business service or functions.

There are currently 3 billion devices connected to the Internet. The number will increase to 20 billion by 2020. While these devices make our life easier, safer and healthier, they are expanding the number of attack targets available to hackers. Therefore, protecting these devices from adversaries and unauthorized access and modification is very important.

This research proposes a novel approach for anomalies detection in IoT systems based on a combination of two robust machine learning algorithms; inverse weight clustering (IWC) and C4.5 decision tree algorithm. IWC is an enhanced version of k-means algorithm that can be used to effectively cluster data into groups based on similarities between this data. C4.5 is a decision tree algorithm that can be used to build decision tree for classifying data. The proposed model was tested and evaluated, and the result demonstrates that the model is very accurate in detecting anomalies in IoT data.

Keywords: Internet of Things (IOT), Anomalies detection. Machine Learning (ML), Inverse Weight Clustering (IW), Decision Tree (DT), C4.5.

ACKNOWLEDGMENT

First and foremost, I would like to thank my family; my mother, my father, my brothers and my sisters for the unlimited encouragement and support during my master's study at DIT in general and during producing this work.

Secondly I would like to thank all teachers and staff at DIT for help and support during my study. The year I spent at DIT was fantastic, full of knowledge, challenges and joy. It will never be forgotten.

Thirdly, I would like to thank my supervisor Dr. Basel Magableh for his continuous and unlimited support, feedback and adjustment of the research. Without your supervision, this research may have not been completed.

Finally, I would like to thank all my friends for their best wishes for me.

Contents

Figures

Tables

Chapter 1: Introduction

1.1. Background

Since the 1960s, the Internet, as a massive network of Local Area Networks (LAN) has been playing a vital role in connecting people, businesses and organizations together. The Internet has broken down geographical barriers between people and has given them a robust, efficient and cost effective means of communications.

Now, it seems that things are about to change in the world of the Internet as a result of the appearance of smart objects that have the capability of generating and communicating data over the Internet in a similar way to humans. Internet of Things (IoT) is the latest technology and systems that holds the potential to change our way of life. One can think of IoT as a technology made of two components; "Things" and "Internet". The "Things" refers to any object or device that has the ability to perceive or collect data about itself or the surrounding environment. Depending on what type, smartness and capabilities of this object, the object may be able to analyze and act smartly with other objects using "Internet" as a network for communications.

The communication over the Internet in IoT is not limited to communication between IoT objects only. It goes beyond this and approaches humans in a way that can make life easier, healthier, and much better. For example, there are so much research around how IoT can help improve humans' health through monitoring their health remotely thus eliminating the need to visit the hospital quite often.

One of these projects on IoT for health is being conducted by the University of Edinburgh in Scotland. The university has created the Centre for Speckled Computing to investigate how IoT can be utilized to improve the health of people in Scotland. They have created tiny computing devices about the size of a thumb. These devices can be attached to people on their chests, monitor and collect respiratory data and then transmit it wirelessly to doctors who are following their cases remotely (Nerney, 2012).

IoT is pervasive and used in almost all aspects of our life. IoT is used by governments around the world to collect data from different sectors and to provide better services in health, transposition, security and development. IoT is used by businesses in order to provide better services to the customers' or to enhance safety and security to workers in the workplace.

IoT is also used by people to better enjoy and manage their life as in the case with Amazon Echo; a smart IoT devices that has the capability of interacting with people using voice. Echo can be asked to provide advice with regards to the weather, schedule alarms, play music, or obtain new feeds from various resources from the internet.

1.2. Research Problem

As IoT becomes more and more pervasive every day, attacks against it increase. According to Gartner (2015), the number of devices connected to the internet by the end of 2016 will be 3 billion. It will increase by 2020 to hit 20 billion. Having such a massive increase in the number of devices by 2020, underlines the question of what sort of sensory data these devices will be able to communicate. Such data can be medical, financial, social, environmental, or any other type of data that has various structures and importance. With this in mind, it is very important to think of how to secure such data and devices.

According to Alexander and Finch (2013), information security is a component of computer science that is concerned with protecting information systems from adversaries who aim to take down the availability, integrity and confidentiality of the data and services provided by an information system. Security of an information system can be maintained using physical and logical controls. The physical controls are used to prevent adversaries from physically reaching the hardware devices that are used to run a given information system. Physical controls can be guards, CCTV, electronic budgets, finger print scanners and iris scanners. Logical controls are software safeguards that can be used to ensure that data and information systems are available only to authorized entities – individuals and process, according to business policies. Logical controls including passwords, access control and intrusion detection and prevention systems.

Intrusion detection and prevention systems are systems that can be used to detect known attacks or anomalies in information systems. The key difference between an intrusion detection system (IDS) and intrusion prevention system (IPS) is that IPS is a proactive system. Therefore, upon detecting or matching attack signature or anomalies, the IPS takes predefined and systematic steps to stop the attack or anomalies. The IDS, on the other hand, is limited to detecting and sending alerts in case of detecting attacks or anomalies (Gordon, 2015).

IDS and IPS can be network-based or host-based systems. They can be used to protect a network or end-user device. The network could be an IoT network and end-user device can be an IoT device such as sensing devices. Furthermore, IDS and IPS can be hardware and software in one bundle or software (Gordon, 2015).

Additionally, there are two methods of detections; signature based detection and anomaly based detection. Signature based detection methods are effective in detecting known attacks by inspecting network traffic or data in systems memory for specific patterns. Anomaly based detection is used in detecting unknown attacks by monitoring the behavior of the whole system, objects or traffic and compare it against predefined assumed to be normal behavior. Any division from normal behavior is treated as a potential attack (Prabha and Sree, 2016).

According to Prabha and Sree (2016), IDS and IPS used both types of detections to stop known and unknown attacks. However, anomaly based detection IDS and IPS have become more popular due to the constant increase in the complexity of attacks against assets. Attackers have become capable of producing malwares that have the ability to change their structure (i.e. polymorphism) on the fly to evade signature based IPS and IDS. Furthermore, once an attack is detected, it takes time to create a signature and apply it to IDS or IPS.

Anomaly based IDS and IPS relies on artificial intelligent (AI) and machine learning (ML) to detect anomalies (Buczak and Guven, 2015). The idea behind AI and ML is to make a machine capable of learning by itself and distinguish between normal and abnormal behavior on the system. The process of teaching a machine takes different forms; supervised, unsupervised and reinforcement learning.

Regardless of the means of teaching, a machine needs to be trained in order to be able to predict. Several ML algorithms have been used in intrusion detections. K-means is one of the most popular algorithms for clustering data into groups. Decision tree algorithms such as C4.5 and C5.0 are among the most popular classification algorithms used to distinguish between normal and abnormal behavior or patterns in data. Most of the intrusion detection systems use such a combination of algorithms to cluster sample data into groups, label them, and then use a classifier to train the intrusion detection systems to distinguish between these groups (Haq et. al, 2015).

According to Haq. et al. (2015), anomaly based intrusion detection systems suffer from false positive, which is the case when IDS or IPS identify normal activity as abnormal. Additionally, anomalies based on intrusion detection systems are said to be computing intrusive system. It requires a lot of processing power and memory to work fast specially if the IDS or IPS is a real time intrusion detection system.

Zhao and Ge (2013) examine the security in IoT. Using anomaly based detection in IoT is challenging and harder than using it with non-IoT for several reasons. Given the large number of IoT devices, attackers have more targets to attack than any time before. Furthermore, these targets are relatively easier to attack than traditional computers since their hardware capabilities in terms of processing and memory is very limited in a way that can render it difficult to use host-based intrusion detection systems. Additionally, IoT devices produce data of different structures and formats and communicate it over various types of networks including, the Internet, wireless sensory network (WSN), radio frequency identification (RFID), Bluetooth and many more.

With this in mind, the question posed could ask whether anomaly based detection techniques be used in detecting anomalies in IoT. The answer to this is yes and a lot of research has been conducted in this area. Pajouh et al. (2016), Hodo et al. (2016), Liu and Wu (2014), and Fu et al. (2011) used several ML learning techniques to detect anomalies in IoT networks. Most of these techniques are designed to detect anomalies of network attacks such as fake IoT nodes, malicious routing in IoT. Minimum attention has been paid to detecting anomalies in data generated by IoT devices themselves.

Therefore, the question that this research is trying to answer is:

How effective is a combination of inverse weight clustering (IWC) and decision tree (DT) classification algorithms in detecting anomalies in IoT?

As mentioned earlier, DT algorithms are generally used for classification in IDS and IPS. It is, however, used with K-means algorithm to produce IDS and IPS models. According to Ashour and Fyfe (2007), K-means has limitations; one of which is the initial selection of data before starting clustering. Ashour and Fyfe (2007) proposed inverse weight clustering (IWC) which is an enhanced version of K-means. This research will answer the above question through exploring a IWC and DT combination instead of K-means and DT combination.

1.3.Research Objectives

This research aims to develop an intrusion detection system for detecting application layer attacks against IoT device using a combination of inverse weight clustering (IWC) and decision tree (DT) algorithms. Usually an IoT device collects sensing data and sends it for further processing via gateways to a central system. Anomalies in data sent by IoT devices might be detected at an application layer in central systems. The objectives of this research are:

1. To deeply understand IoT, and how it works.
2. To deeply understand anomalies detection techniques; IWC and DT.
3. To develop a model for intrusion detection based on IWC and DT and apply it to IoT.
4. To evaluate the model.
5. To offer recommendations on improving the model.

1.4. Research Methodology

This research is set to produce a new intrusion detection model for IoT using a combination of two algorithms; inverse weight clustering algorithm and decision tree classification algorithm. Just like any other research, there is a need for data collection in order to be able to build and test the system. Collecting data might be costly, therefore, it is possible to use data that is already collected by others to conduct the research.

One of the freely available IoT data is that published by Madden (2004). It is collected from Intel Lab; which is a lab consisting of 54 sensors speared in a way shown in figure 1.1. Each sensor collects 4 values; temperature, humidity, light and voltage. The sensors collect data every 31 seconds and sends it to the central collection points in the lab. The data can be downloaded for free from: http://db.csail.mit.edu/labdata/data.txt.gz.

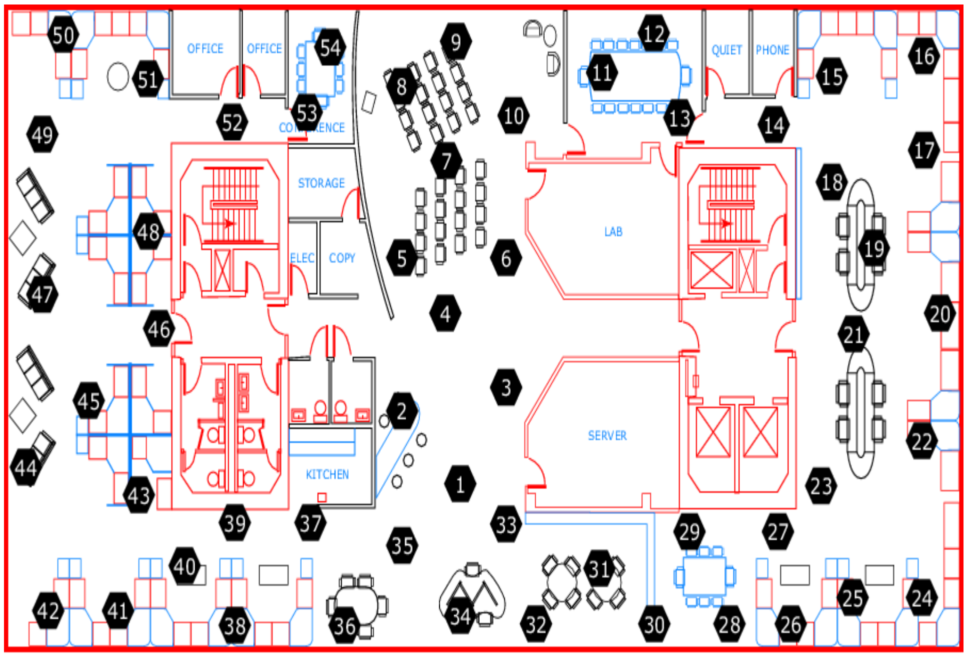


Figure 1.1: Map of sensors in Intel Lab

Intel Lab is an IoT data that is not intended use in intrusion detection. The data was collected for general purpose. In contrast to this, there are many intrusion detection datasets such as KDD 99 available for free and is widely used in intrusion detection research. However, this data is not IoT data and cannot be used in this research. Fu et al. (2011) acknowledge the difficulties in obtaining IoT dataset for intrusion detection. Therefore, they used Intel Lab dataset in their research and they had to manually add some abnormal records in the dataset to test their theory. This approach is followed in this research.

The modified version of Intel Lab IoT dataset will be used as an input to the IWC. IWC is supposed to cluster data in the dataset into two groups based on similarities between data records. After clustering is completed, the groups are labelled as “normal” and “abnormal”. The input data is split into two parts; training data (70%) and testing data (30%). The training data is used to build the classification tree; decision tree, and the testing data is used to test the system in predications. MATLAB is used to implement the algorithms and test the system.

The confusion matrix is used to evaluate the system in terms of true positive, true negative, false positive and false negative as shown in table 1.1. Then the results will be compared with work from other researchers to find the accuracy of the proposed intrusion detection model.

Table 1.1: Confusion Matrix

Actual	Predicted Normal	Predicted Attack
Normal	True negative (TN)	False positive (FP)
Attacks	False negative (FN)	True positive (TP)

1.5. Scope and Limitation

This research is set to provide an intrusion detection model of data communication by IoT devices only. The data that this model is designed to work on is numerical data or real values such as temperature and humidity. The model will not be able to deal with textual data.

Furthermore, the capabilities of the model in terms of intrusion detection is limited to detecting anomalies in data at application level not network layer. Therefore, any attack that could change hardware or software addresses of the IoT devices will not be detected by this model.

Finally, the proposed model for intrusion detection in IoT is designed to be logical control (i.e. software), but not physical control.

1.6. Document Outline

The rest of this dissertation is designed as follows: chapter 2 contains the literature review and related work. It describes in detail the definition of IoT, IoT systems architecture, IoT types and characteristics. It also describes machine learning types, algorithms, and usage in intrusion detection. The chapter furthermore contains related work of other researchers in the subject area.

Chapter 3 is the chapter of methodology. It describes the dataset used in training and testing the system. It describes how the data is prepared before it is fed into the detection model. It also describes IWC and DT algorithms and how they work. Additionally, it provides details of the evaluation method that is used to evaluate the proposed model.

Chapter 4 is the chapter of implementation and result. It discusses the implementation of the model using MATLAB. It shows how the logic of the system works, and provides the results found from testing the system.

Chapter 5 is the chapter of analysis and discussion. Chapter 6 is the chapter of conclusion and future work. It contains reflection, discussion of limitations and recommendations.

Chapter 2: Literature Review and Related Work

The Internet as we know it is changing rapidly as a result of the introduction of smart devices that have the ability to communicate with humans and with each other's use over the Internet. In a recent published research by Gartner – the technology research giant, the number of "Things" connected to the Internet by 2016 is expected to be 6.4 Billion (Van Der Meulen, 2015). According to the same source too, this number is expected to reach 20.8 billion by 2020.

One question immediately posed would be where such a number could come from. The answer is simple; an end user may have more than one device connected to the Internet. The list of devices includes but not limited to iPhone, iPad, iWatch, Smart TVs and many more. Each device has the ability to generate data and share it over the Internet, and this is what constitutes the Internet of Things (IoT).

2.1. Internet of Things (IoT)

IoT is an umbrella term that covers technologies, design principles, and systems associated with the ever-growing phenomenon of Internet-connected devices – "Things". According to Ning (2013), IoT as a phrase is not new. It appeared for the first time in 1999 at Massachusetts Institute of Technology (MIT) Auto-ID Centre, and was used to refer to building an Internet based network that cover all things in the world to realize automatic identification of things through information sharing. The "Things" use related technologies such as Radio Frequency Identification (RFID) to enable communication and realization.

2.1.1. IoT Definition

In 2005, the International Telecommunication Union (ITU) – a UN agency concerned with information and communication technology, augmented the concept of IoT and suggested four technologies to realize IoT; RFID technology, intelligent embedded technology, nanotechnology, and sensor technology (Ning, 2013).

According to Jung, Cho and Kang (2014), IoT is still in early stages and research groups agree that a common solution to realize IoT has not been developed yet. As a result, one can find many definitions to IoT. For instance, Jung, Cho and Kang (2014) define IoT as a computing environment where various physical RFID embedded things interact and harmonize together to deliver smart IT services.

The European Research Cluster on the IoT (IERC) defines IoT as:

"a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network". (IERC, 2014).

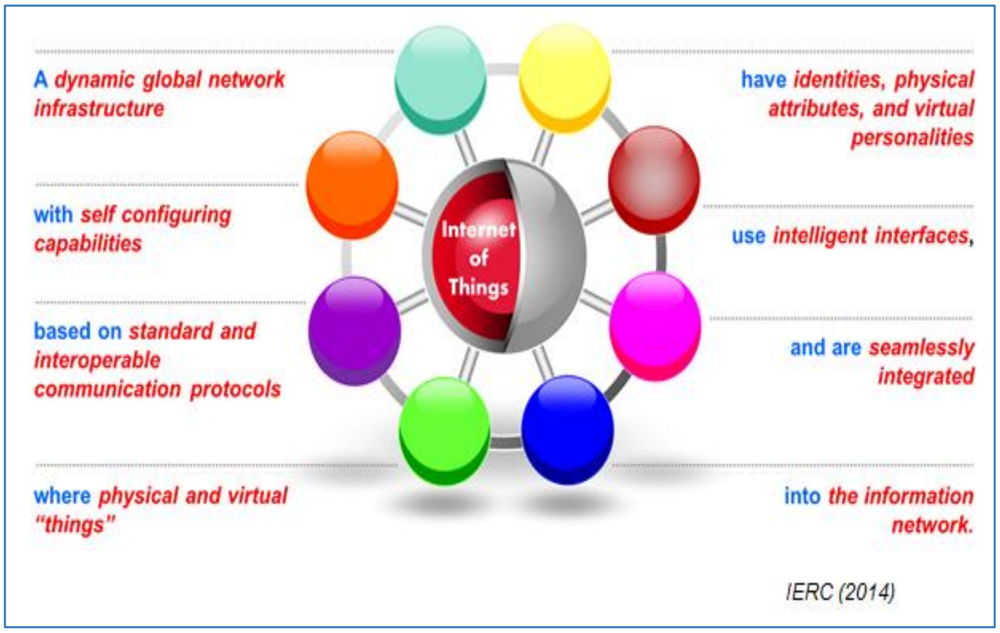

A dynamic global network infrastructure
with self configuring capabilities
based on standard and interoperable communication protocols
where physical and virtual "things"
Internet of Things
have identities, physical attributes, and virtual personalities
use intelligent interfaces,
and are seamlessly integrated
into the information network.
IERC (2014)

Figure 2.1: IERC definition of IoT.

Figure 2.1 breaks down the IERC definition of IoT. A "dynamic global network infrastructure" is a network that has the ability to automatically provide and allocate resources to meet the demands of the existing and forthcoming IoT devices. This is something that can be utilized using cloud computing (CC) and software defined networks (SDN). Together CC and SDN can allow for "self-configuring capabilities" based on "standard communication protocols" such as TCP/IP and UDP. Such communication protocols need to be embedded into devices with "intelligent interfaces" that have the ability to connect to the Internet, exchange data and information in a "seamlessly integrated" manner in the same way current personal computers do.

According to Holler et al. (2014), there are two reasons behind the revaluation of IoT; the decreasing cost of semiconductors and the introduction of Internet Protocol version 6 (IPv6). Advances in semiconductors as well as the cost of acquiring the raw material to produce such semiconductors have decreased the cost electronic devices manufacturing. This in turn has led to ubiquitous computing everywhere.

The IPv4 has been in use for a while now and is totally exhausted. IPv6 is the new version of IPv4 and introduces a significantly bigger IP address space. While the length of IPv4 address is 32 bits, the length of IPv6 address is 128 bits. This means that the address space in IPv6 is 2 to the power 128 (i.e. 2^{128}) IPv6 addresses. This number of IP addresses offers more than enough to accommodate any number of "Things" to be connected to the Internet (Jung, 2015).

2.1.2. Classification of "Things" in IoT

The IERC definition of IoT refers to "physical and virtual things" that have identities, attributes and characteristics (IERC, 2014). Ning (2013) argues the realization of the IoT by some researchers as a network for addressing physical objects in terms of connection, control, and ubiquity. Ning (2013) suggests that the mapping between "things" in the physical and cyber world is the core of IoT, and therefore, "things" can be classified into two types: (1) Physical things, (2) Cyber things.

.

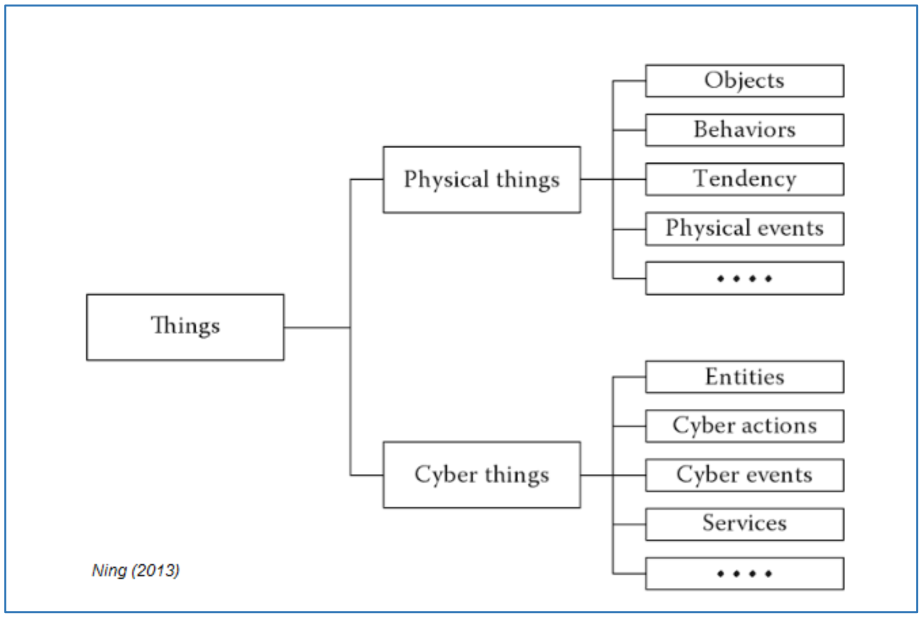


Figure 2.2: Things classification.

Figure 2.2 shows both types of “things” in IoT. The physical things cover:

1. Objects: which are concrete things with measurable bodies. Vehicles, persons, tablets, and birds can be examples of objects.
2. Behaviors: which are the movements of objects because of certain motivations. Driving, running, monitoring, and eating can be examples of objects behaviors.
3. Tendency: which are the trends in things of themselves or as a result of external environment such as traffic becoming congested or weather being rainy.
4. Physical events: which are a collection of all the above collaborating to describe what is going on as triggered by certain circumstances in the physical world.

The cyber things cover:

1. Entities: which indicate the abstract things such as code and data.
2. Actions: which indicate the processing of data such as data transmission from one entity to another.
3. Events: which indicate the causation of actions taken by an entity such as an entity reports data being transmitted.
4. Services: which indicate the functions offered by a thing or the functions that are being offered to a thing to deliver a specific goal.

One can think of physical and cyber things as Google’s car. This car represents an object that has behaviors such as driving, self-maintenance, and parking. The car can have tendency as when it discovers that weather has started to rain, it is expected from the car to take physical actions such as activating wiper blades. Cyber things bring semantics and context to the physical things. The code that is deployed to the wiper blades represents the abstractions that share and process data based on services available to take actions and report events.

2.1.3. IoT Architecture

According to Vermesan and Friess (2014), ITU proposed a layered architecture of IoT. There are four layers in this architecture; device layer, network layer, service and application support layer, and application layer – the layers are ordered in a bottom-up approach. In addition to these layers, ITU proposed two additional layers; management capabilities and security capabilities.

Figure 2.3 shows all the layers in the proposed architecture for IoT by ITU. The main four layers stacked horizontally are functional layers. In other words, these layers are used to collect, transmit, and process data in IoT networks. The other two layers depicted vertically take a part of functionality of each layer for the purpose of management and security.

At the far bottom of the stack there is the "Device layer" which is also known as the "perception layer". In this layer there are IoT devices such as smart electricity meters and sensors. This layer also contains gateways, which are a very critical component in IoT systems (Vermesan and Friess, 2014). The IoT gateway as the name implies are devices that receive and collect sensors data within their reception range, translate sensors protocols, process sensors data, and routes sensors data either inward or outward.

In an environment full of sensors, the gateways play a vital role. Think of a building with hundreds of sensing objects that have different functions and capabilities, and produced by different manufactures. These objects are expected to sense a wide range of tendencies – e.g. light, temperature, humidity, noise, people, and fire, and exchange that data with the gateway using different communication protocols such as serial ports (e.g. RS-232), MQTT, Bluetooth, Wi-Fi, Ethernet ZigBee, USB, and others. If the gateway does not hold the capability to understand the data that comes from sensing objects using different communication protocols, there would be no effective communication and control over that environment (Familiar, 2015).

Additionally, sensing objects are expected to act either autonomously or as instructed by other objects in case of an event happening. This requires generating additional data and then pushes it to relevant objects to undertake physical events. In doing this the gateway might need to connect to an out of band entity through the cloud to obtain analytics and insights. This augments the vitality of gateways at device layer in IoT architecture. Thus, protection and fault tolerance are required to avoid turning gateways to a single point of failure (Familiar, 2015).

The network layer in IoT architecture provides that ability to transport data from sensing objects and gateways to the cloud using the Internet. Gateways as well as some sensing objects have the capability of supporting internet protocols such as IPv4, IPv6, and 6LoWPAN (Familiar, 2015).

The services and application support layer is somehow similar to transport and session layer in the Open Systems Interconnection (OSI) combined. It controls the type of transmission, quality and destination port on the other side of connections. In other words, it helps routing sensing data to the right application in the application layer so that it can be handled (Familiar, 2015).

The application layer, however, is where applications such as those depicted at the top of figure 2.3 reside. IoT applications list includes but is not limited to smart transport, smart buildings, smart living and others (Vermesan and Friess, 2014).

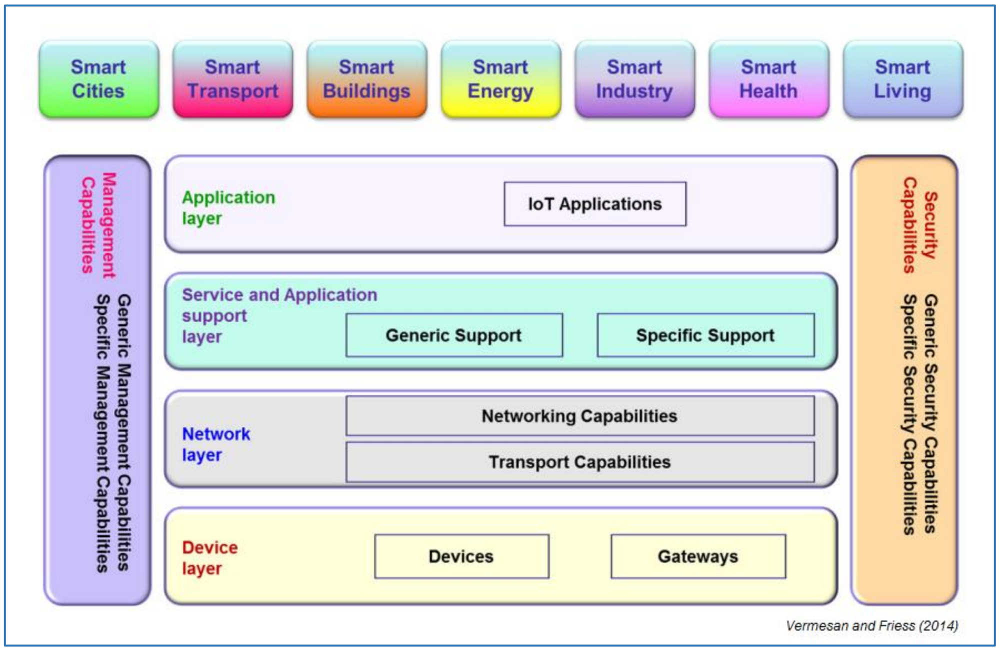


Figure 2.3: IoT architecture as developed by IUT

Management capabilities in IoT architecture refers to the tools that can be used to monitor and control the objects remotely. Remote management utilizes remote connection protocols such as SSH and VPNs to reach far objects and conduct management tasks such as firmware update, patching and configurations (Fortino & Trunfio, 2016).

Management capabilities also include assets management whereas each object in the network has an ID to associate that object's attributes with it. An object's attributes include object ID, manufacture, date of purchase, date of put in operation, capabilities, supporting protocols and others (Fortino & Trunfio, 2016).

Security capabilities refer to the tools and measures that are in place to protect sensing objects as well as the data they generate while it is moving from one end to another. It covers several types of controls some of it physical and most of it logical. The physical controls can be IoT objects themselves such as CCTV, while logical controls can be access control, identify management, encryption that put in place to prevent unauthorized access or modification to objects and data (Fortino & Trunfio, 2016).

2.1.4. IoT intrinsic Characters

Ning (2013) provides three unique characteristics of IoT, which make IoT the future of the Internet. These characteristics are:

1. Ubiquitous sensing – IoT has the ability to provide sensing capabilities to cover the physical and cyber world beyond human sensing ability (e.g. hearing, vision, touch, and smell). The variation and intensity of sensing devices can bridge the gap between human and machine and lead to unifying the physical and cyber world to solve human issues. Guardian Angels for a Smarter life is a project funded by the EU and aims at creating autonomous and intelligent personal companions that can assist and protect humans from infancy to old age (FET, 2012). Sensing objects with zero power consumption can be used to realize and monitor the physical statues of humans during the whole day and share that data with the supervising doctors. Environmental sensing objects will be used to grasp the surrounding environment and alert humans in case of hazardous conditions. Emotional sensing objects will be used to grasp human emotion and react to them. Such a project shows how IoT will assist in realizing a world in which can help humans while remaining tiny and unseen, and without consuming power.

2. Network of networks – IoT involves several and various kinds of networks. In other words, it is a heterogeneous network that covers GSM, CDMA, WCDMA, and IP networks.
3. Intelligent processing – IoT objects are deigned to be intelligent just like humans. The difference however, is the time and the speed at which data can be processed. IoT objects will be able to process a huge amount of sensing data in a far faster mode than humans can do. This could free humans from spending time in exploiting the information and focus on qualitative thinking.

2.1.5 IoT Challenges

IoT is a promising technology, yet it is still at a very early stage and faces many challenges. Jung, Cho and Kang (2014) identity some of challenges in IoT. Firstly, there is no standard architecture for IoT systems. Since the IoT is still at early stage, vendors are not keen to produce objects that comply with other vendors' objects to achieve financial gain and push consumers toward vendors' lock-in.

Secondly, networks in IoT are heterogeneous which make connecting, operating, managing and securing IoT network hard task. IoT objects use different communication protocols to communication over different types of network (e.g. GSM, WAN, WSN, Blutooth).

Finally, security and privacy. Given the number and various types of IoT objects as well as their limited hardware capabilities, it is almost impossible to deploy host-based security measures to secure the devices. This pushes towards centralized and non-centralized network based protections as well as anomaly detection and protection mechanisms.

2.2. Machine Learning for Anomaly Detection

Machine Learning (ML) is one of the top 10 emerging technologies in 2015 (ISACA, 2015). Figure 2.4 shows ML is among the top four technologies that can deliver big value to most users. It is marked with a red circle to show that ML is not yet embraced by most CIOs – Chief Information officers since it requires resources of high cost – marked with "H" in figure 2.4.

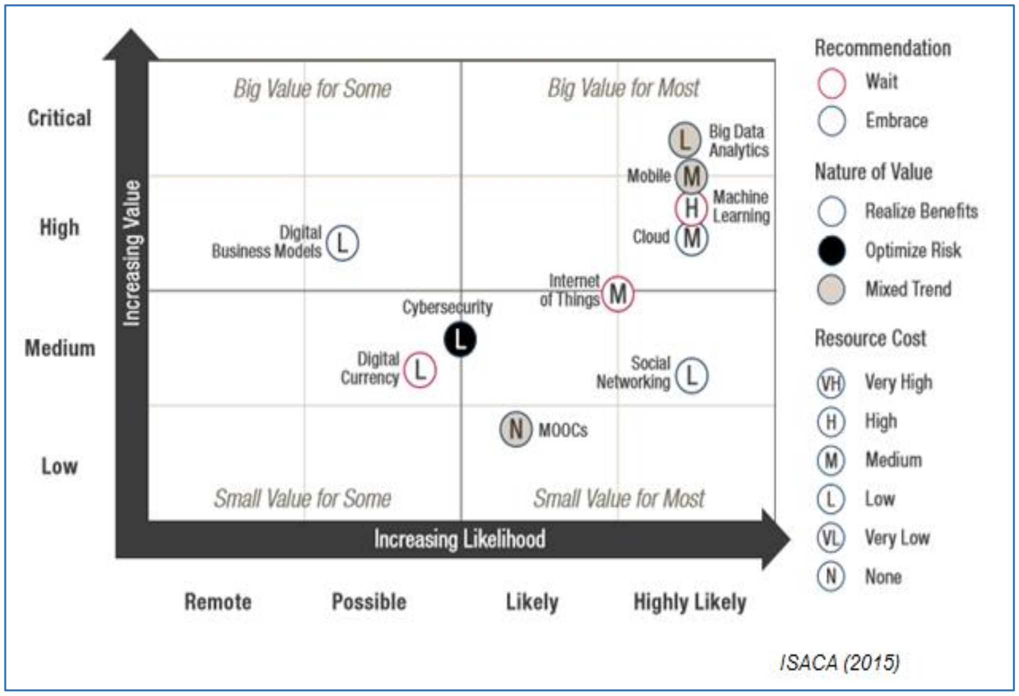


Figure 2.4: Machine Learning is an emerging technology.

2.2.1. Definition

ML is a field of computer science that is concerned with making a computer machine capable of learning without direct programming or intervention from humans. As a field, ML learning has been here since 1959 and maybe even before this date. According to Bell (2016), ML was originally defined by an engineer at IBM, named Arthur Samuel in 1959 as:

"Field of study that gives computers the ability to learn without being explicitly programmed"

In 1997, Tom Mitchell – the Chair of ML at Carnegie Mellon University, provided another definition of ML which is widely used nowadays. The definition is:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance P, if its performance at tasks in T, as measured by P, improves with the experience E"

In Mitchell definition of ML, three objects can be observed;

1. Task (T) – can be one or more
2. Performance (P)
3. Experience (E)

It can be said that a computer program that runs a set of tasks is learning if the performance of the tasks being conducted by this program improves based on the experience. Therefore, the relation between experience and performance can be described as linear relation. Therefore, more experience means improved performance.

Therefore, a system that has the ability to accept previous data as an input, harness and analyze this data to systematically to find patterns, and then undertake predictions is a system that is capable to learn. With continuous learning and tuning, the predictions that the system can produce become more fine and accurate.

2.2.2. Leaning Styles in ML

Although "learning" is very important in ML, it is not the goal. According to Shalev-Shwartz and Ben-David (2014), the main goal of machine learning is to produce a system that can automatically and correctly detect meaningful patterns in data.

According to Shalev-Shwartz and Ben-David (2014), "learning" in ML can be:

1. **Supervised learning:** Labeled data is used in supervised learning. The teacher provides the system with labeled data in order to help the system to build a knowledge base that upon which the system will attempt to recognize unlabeled data when presented. The teacher, before starting to teach the system, should label the data by giving it a meaningful label, class, or tag. For example, assume that there is a dataset with millions of pictures. The teacher

can label each picture by providing a brief textual description such as “Dog”, “Cow”. Once labeled, the dataset is given to the system to build its knowledge base. When presented with unlabeled data such as a picture of a “Lion”, the system should be able to provide an answer to such questions as what animal is in this picture?

2. **Unsupervised learning:** unlabeled data is used in unsupervised learning. Unlike supervised learning where the teacher knows the labels or classes of each item in input data, in unsupervised learning the teacher does not have knowledge about the labels or classes of the items in input data. The teacher, however, relies on the system itself to find out these labels or classes. This requires the machine to conduct systematical processing and analysis to break down data into groups with similar features based on certain attributes chosen previously either by the system or by the teacher. Once the data is broken into groups, the teacher can label the groups to the system, so that any prediction of new unlabeled data when presented becomes possible. In order to better understand this type of learning, let us assume there is a dataset of millions of transactional records. The teacher may ask the machine to break down this dataset into groups based on the IP addresses used in each transaction. Another teacher may ask the machine to break down the dataset into groups based on IP address and time at which the transaction took place. The teacher can then label the groups geographically and mine them to find patterns of the shopping attitudes of each group of people based on their location. Such a model will then be able to provide suggestions to people based on their location simply, just like Amazon does when recommending goods to people visiting its sites (Flach, 2012).

3. **Reinforcement Learning:** labeled and unlabeled data can be used in reinforcement learning to form basic knowledge. Reinforcement leaning works by having the teacher rewarding the system for each correct or incorrect prediction. The reward acts as feedback that the system can rely on while making next predictions. In time, the system will be able to have access to a knowledge base full of execution paths based on the input used to be given to

the system. When a new input is given to the system, the system will try to find the best execution path or combine more than one execution path to produce predictions and wait for the reward. If the reward happens to be better than the previous rewards regarding the same input, then this path becomes favorable. Reinforcement learning is used in online games such as Chess. When the machine plays chess against humans, it stores the steps it takes with regard to each movement the human opponent took and waits for the teacher for reward. All correct steps are rewarded and used by the machine to form a new and wider range of tactics that at some point qualify it to beat its human opponents. What makes such types of learning unique is the output that the model produces. In the case of supervised and unsupervised learning, the output of the ML model in terms of predictions is very specific on a probability scale. The model produces predictions with a certainty level that ranges from 0 to 1. The closer the prediction to 1 the more certain the machine is about it. In reinforcement learning, the machine produces predictions at certainty level of 1 and waits for feedback from the teacher to find out whether the predictions were accurate or not.

Back to Mitchell definition of ML, E or experience represents input data for training. This data can be labeled or unlabeled. The unlabeled data is raw data without any tags which is significantly easy to obtain compared to labeled data. However, unlabeled data requires more time and resources to be processed by the ML model and used to produce predictions. On the other hand, labeled data is not raw data. It is processed data to include description tags. Labeled data is much harder to obtain when compared to unlabeled data. It is usually not free as humans participate in judging the data and produce labels or classes (Flach, 2012).

2.2.3. Algorithms of ML

An algorithm is a set of rules to be followed in a certain order in order to solve a problem (Basu, 2013). In ML, there are plenty of algorithms that may have similar functions. ML algorithms can be split into the following groups (Brownlee, 2013):

1. **Clustering Algorithms:** These algorithms are concerned with identifying the structures or patterns in data in order to organize it into groups. Each item in the group would be similar to each other in terms of structure to the maximum level. The following are the most popular clustering algorithms:

   a. K-Means

   b. K-Medians

   c. Hierarchical clustering

   d. Expectation Maximization (EM)

2. **Regression Algorithms:** These algorithms are concerned with the statistical analysis of data in order to estimate the relationship between variables. These algorithms focus on the relationship between one dependent variable and one or more independent variables. The following are the most popular regression algorithms:

   a. Linear regression

   b. Logistic regression

   c. Ordinary lest squares regression (OLSR)

   d. Stepwise regression

3. **Decision Tree Algorithms:** These algorithms are concerned with constructing decision making models based on actual values of the attributes in data. The way these algorithms work is by training the system on data used in classification and regression. The algorithms search the tree structure (i.e. fork) until decisions are made. The following are the most popular decision tree algorithms:

   a. C4.5 and C5.0

   b. M5

   c. Classification and Regression Tree (CART)

d. Decision Stump

4. **Bayesian Algorithms:** These algorithms are directly concerned with applying Bayes' theorem in classification and regression. In short, Bayes theorem studies the conditional probability of each possible cause for a given outcome. The following are the most popular Bayesian algorithms:

    a. Naïve Bayes

    b. Multinomial Naïve Bayes

    c. Gaussian Naïve Bayes

    d. Bayesian Network

5. **Artificial Neural Network Algorithms:** These algorithms are concerned with applying regression and classification algorithms in a way similar to the human biological neural networks. The following are examples of artificial neural network algorithms:

    a. Back-Propagation

    b. Perceptron

    c. Radial Basis Function Network

    d. Hopfield Network

There are furthermore ML algorithms such as deep learning, dimensionality reduction, ensemble algorithms, association rule learning, regularization, instance-based, and features selection algorithms – to name just a few.

2.2.4. ML Breaks out

While ML has been known since 1959, the emergence of cloud computing as a model of commodity of resources that can be used on demand has helped ML to breakout again in the recent years (ISACA, 2015). The majority of ML algorithms are said to be computing power intrusive algorithms. They harness the data of several types and structures in real time for training and prediction and require access to data centers capabilities, which is something cloud computing can now provide to most end users.

The end users can be governments, businesses, or clients. According to Morgan (2015), governments nowadays use ML for public safety through collecting various data from several parts in a country and use it – with the help of ML, to spot fraud, thefts, and terrorism. Tech giants such as Facebook use ML to analyze and predict the content of videos and images uploaded by users on Facebook (Candela, 2016). Facebook ML system can automatically generate tags and caption based on the result of prediction.

2.3. Related Work

This part aims to critically review previous relevant work in the field of Intrusion Detection Systems in the context of Internet and specifically in the context of Internet of Things. We analyse the different techniques used for the main results achieved and we show how intrusion detection for Internet of Things has its own specific challenges due to the heterogeneity of the devices, the limited computational capabilities and the massive number of connected devices that makes the embedding of intrusion detection systems harder.

Intrusion Detection Systems (IDS) are used to protect data that is being exchanged between end points and processes within an information system, from unauthorized access and modification. According to Golmah (2014), an IDS is designed to distinguish between normal behaviours from abnormal behaviours based on effective classification model built inside that IDS.

2.3.1. Intrusion Detection Techniques

The problem of building effective classification models for an IDS is generally approached using several data clustering and classification algorithms. A recent work by Golman (2014) represents a classical hybrid approach closer to what we implement in this thesis work.

Goldman suggests a hybrid intrusion detection system based on Support Vector Machine (SVM) and C5.0. The author claims that using such a combination of algorithms would improve the accuracy of intrusion detection when compared to using SVM and C5.0 separately.

Golman (2014) model suggests using SVM for classifying data into classes. Given training data, SVM has the ability to map data into a high dimensional features space through using nonlinear mapping as depicted in figure 2.5. By computing the separating hyperplane of a given training data set, support vectors can be also computed and used to create a theoretical margin that SVM can use to predict which class or category a new sample of data falls into.

Golman (2014) suggests combining SVM with C5.0; which is a widely used implementation of the decision tree. C5.0 builds a tree with one root node and multiple non-root nodes. Each node aims to provide an answer with regards to specific attributes or tuple in the input record. Nodes report to the root node which uses information theory to make judgements.

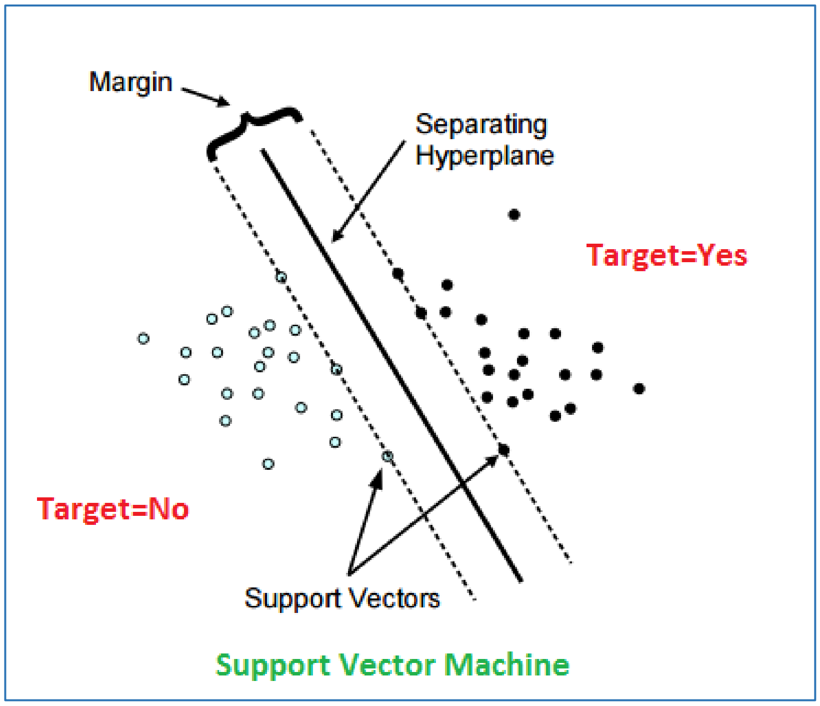


Figure 2.5: SVM basic concepts.

Golman (2014) uses KDD Cup 1999 to test the proposed intrusion detection technique. KDD Cup 1999 data set was split into training (75%) and testing (i.e. 25%) data. The Golman model passes training data to C5.0 first. C5.0 adds node information to the original data set and creates a new version of training data. The new data set is passed to SVM for evaluation.

When comparing the hybrid intrusion detection model to SVM, Golman (2014) found that the average preciseness of detection using C5.0 and SVM together is 99.96, while SVM on itself reported 99.78% detection preciseness.

The recent work by Hajare (2016) stresses the fact that the continuous increase in attacks against critical infrastructures and the failure of existing intrusion detection systems in detecting such attacks are due to the fact that these attacks are simply "unknown". Therefore, a system such as the one Golman (2014) proposed might not work since it relies on labelled data fed into C5.0 during the training stage.

Hajare (2016) proposes a novel intrusion detection model. The model uses MapReduce to process large structured and unstructured data set into key/value pairs. The way MapReduce produces key/value pairs for intrusion detection relies on using a combination of Fuzzy C-Means (FCM) clustering and SVM for classification. FCM uses features to cluster data into groups. The advantage of FCM over non-fuzzy CM is that FCM allow data points to belong to multiple clusters while non-fuzzy CM does not. In non-fuzzy CM, the data point can only belong to one cluster. Hajare (2016) has not published the results of testing the proposed model, therefore, no results can be given with regard to the accuracy of such model.

Tanpure et al. (2016) propose a hybrid intrusion detection system using two data mining techniques; K-means and Naïve Bayes for clustering and classifying data. The attacks that the Tanpure et al. (2016) model is designed to detect are; DoS, Probe, U2R and R2L. These classes refer to the types of attacks that exist in KDD cup 99 data set used by Tanpure et al. (2016) to evaluate their work. The classes of attacks are explained as follow:

1. DoS – Denial of Service; which is an attack that takes a system down by utilizing the resources in that system including processing, storage, and memory resource.
2. R2L– Remote to local; which is an attack that aims to gain access to a local machine remotely exploiting any existing vulnerability in the system.
3. U2R – User to Root; which is an attack that aims to escalate access to a system from normal user privilege to administrator privilege.
4. Probe – which refers to scanning activities that attack would carry out to collect information about a target before attacking it.

Initially, Tanpure et al. (2016) use K-means to cluster data into three clusters; DoS, Probe, and others. The "others" cluster would contain R2L, U2R and normal data. While it is easy to distinguish DoS and probe attacks from each other's and from other attacks, it is very difficult to distinguish between R2L, U2R and normal data. They are quite similar. Therefore, Tanpure et al. (2016) suggest using Naïve Bayes classifier to classify between attacks classes and normal traffic. Naïve Bayes classifier is being described as an effective learning algorithm that can establish classification criteria based on the relationship between independent and dependent variables. Tanpure et al. (2016) have not published the results of testing the proposed model, therefore, no results can be given with regard to the accuracy of such a model.

2.3.2. Intrusion Detection for IoT

On observing intrusion detection models proposed by Golman (2014), Hajare (2016) and Tanpure et al. (2016), it can be concluded that machine learning techniques in clustering and classification of data for network-based attack detection are commonly used and widely accepted. With this in mind, Sherasiya and Upadhyay (2016) point out that IoT objects are also exposed to such types of attacks. Furthermore, Sherasiya and Upadhyay (2016) point out that the data that IoT objects exchange are of the same value and importance – or occasionally more important than a non-IoT counterpart. However, when comparing IoT objects with non-IoT objects, IoT objects are easier to be attached due to the limited processing capabilities they have got, the thing that makes it very hard to deploy security solutions inside them.

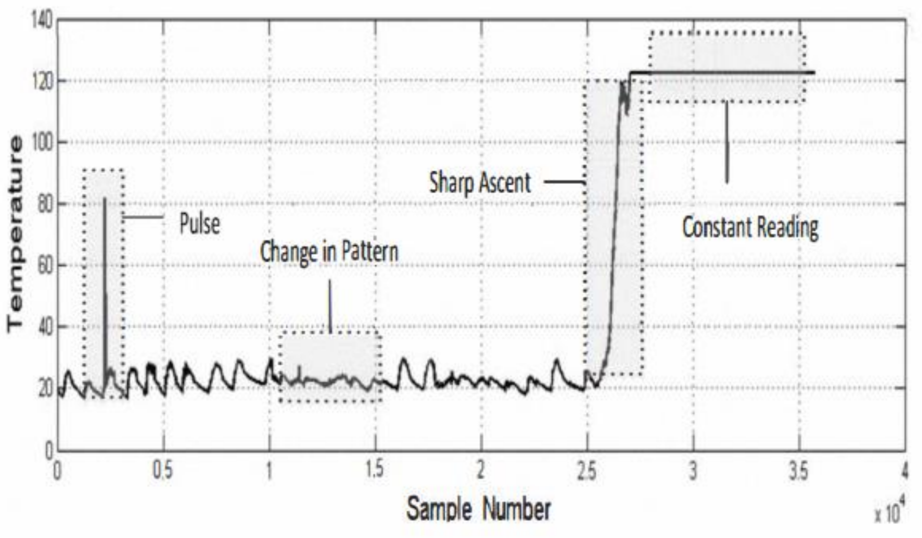
Sherasiya and Upadhyay (2016) propose a lightweight intrusion detection system as opposed to heavy weight intrusion detection systems – the IDS that use machine learning techniques. The model aims at *detecting nodes* within the IoT environment that have multiple identities. The model Sherasiya and Upadhyay (2016) offer is valid in the case of wireless networks since it relies on the strength of the wireless signal to discover genuine and adverse nodes, and on secret key and ID to identify nodes with multiple identities when they attempt communication with IPv6 border router.

It can be seen that the Sherasiya and Upadhyay (2016) proposal is more a patrolling model rather than an anomalies detection model. The model does not provide any means to detect *anomalies in payloads* that nodes may send to the control centre.

Fu et al. (2011) propose a new intrusion detection scheme for IoT based on anomaly mining. Fu et al. (2011) point to the key challenges in IoT environment that render typical intrusion detection schemes used with non-IoT systems non effective in the context of IoT. On the one hand, IoT is vulnerable and open to a wide range of cyber-attacks due to *opening deployment* and *limited resources*. On the other hand, and unlike other non-IoT objects, IoT is usually a *heterogeneous network* of devices from many vendors that lack common governance in terms of protocols and standards. The scheme that Fu et al. (2011) proposes is believed to address these problems over two stages;

1. Using anomaly mining algorithm to detect anomalies in data of *perception layer*
2. Using a distributed intrusion scheme to distinguish between anomalies and malicious intrusion at *network segment level* but not as a whole – i.e. not using a centralized intrusion detection scheme.

Assuming that each IoT object has a terminal that pushes raw data toward a centralized collection point for further analysis, Fu et al. (2011) call data at this level *data of perception layer*. Such data can be temperate, humid or in fact anything that a sensor as an IoT object can collect and report. Fu et al. (2011) believe that anomalies can be detected at this stage through analysing data patterns and reporting anomalies to the distributed intrusion scheme in stage 2.



Fu et al. (2011)

Figure 2.6: Example of anomalies in data.

Figure 2.6 illustrates a sample of temperature data as collected by a sensor. Stage 1 of Fu et al. (2011) relies on analysing such a data sample to find anomalies based on attributes suitable for intrusion identification. Fu et al. (2011) identify 5 *anomaly attributes*;

1. *Pulse*: a sudden abruption of data for a very short time.
2. *Sharp ascent*: a sudden abruption of data that stays for a long time.
3. *Sharp descent*: a sudden drop of data that stays for a long time.
4. *Constant reading*: constant reading over a period of time.
5. *Change in patter*: data over a certain period of time is similar to normal patterns. However, there could be a difference in mean and/or variance.

In order to observe these activities in data samples, Fu et al. (2011) have used an analytical algorithm named *SAX* to partition data samples based on time to windows. A window will contain data for a certain amount of time. SAX uses the 5 anomalies attributes to compare windows with each other's, extract the matching anomalies, and report to stage 2 using the following frame:

<*terminalid, anomaly_attri, property_attri, timestamp*>

The Terminalia is the ID of the terminal; the IoT object at perception layer. Anomaly_attri is the anomaly attribute as explained above, property_attri is the actual value or data which can be voltage, temperature, light, humidity, etc., and timestamp is the time at which the anomaly was detected.

Fu et al. (2011) use an unsupervised data-mining algorithm to identify the normal pattern. While Fu et al. (2011) have not disclosed what unsupervised data mining algorithms were used, it can be assumed that clustering followed by identification classification algorithms were used. The former will be clustering data based on similarity into unknown groups, and the latter will be classification of data into normal and abnormal groups. Identification of normal patterns in terminal data will make it easy to deeply use 5 anomalies attributed to further classify the anomalies in the abnormal data group.

Having anomalies identified, and reported to the intrusion detection systems, Fu et al. (2011) can now distinguish between 3 types of activities; data polluting, degradation of service, denial of service. Stage 2 of the Fu et al. (2011) scheme suggests deploying their intrusion detection scheme at data perception layer, data distribution layer (i.e. network access layer), and application layer, and ensure that there is communication between IDs on all these layers to exchange detection rules.

In order to evaluate their system, Fu et al. (2011) used *Intel Lab Project* dataset. The project in essence is a lab with sensors that collect temperature, humidity, light and voltage every 30 seconds and report to a central collection point. The dataset Fu et al. (2011) used contains 2.3 million readings. Currently there is no dataset with attacks already annotated and available publically. Therefore, Fu et al. (2011) added manually random anomalies to the reading and created their evaluation based on them. Unfortunately, there is no reported accuracy for the system.

Liu and Wu (2014) describe more reasons why traditional intrusion detection schemes cannot be applied effectively in IoT. Traditional intrusion detection schemes are complex and computationally intrusive schemes. Hence they cannot be applied directly to IoT. Furthermore, model based and similarity based anomaly detection techniques suffer from complexities in design and implementation as well as high false alarm. Therefore, Liu and Wu (2014) suggest a lightweight anomaly-mining algorithm for IoT. Liu and Wu (2014) use Euclidean distance and Jaccard coefficient to find anomalies in data. Euclidean distance and Jaccard coefficient are mathematical operations that can be applied on two data sets to measure the similarity between them.

With regard to IoT, Liu and Wu (2014) assume that each sensor produce data in multiple records. By measuring the distance between two records from the same sensor, it can be decided how similar to each other the records are. Knowing the normal state of the record, Liu and Wu (2014) put an assumption or threshold based on the Jaccard coefficient:

If 0.8 < Jaccard coefficient <=1; normal

If Jaccard coefficient < 0.8; abnormal

In other words, if the similarity between two records of data is less than 0.8, then abnormality does exist. If the similarity was found more than 0.8, then data is normal. Liu and Wu (2014) use sliding window as a way to maintain statistical information is data stream. Hey used unsupervised methods to find normal pattern in data. Unfortunately, Liu and Wu (2014) do not disclose any information with regards to which dataset they used as well as the result they obtained.

Butun et al. (2015) approach IoT anomaly detection from a different angle focusing on IoT in the cloud, which seems to be an extremely interesting approach due to the explosion of cloud computing and big data technologies. The authors analyse the current challenges of security in IoT. Firstly, the size and number of data pumped into the cloud from IoT devices is massive. By 2020 there will be an estimated 26 billion IoT devices connected to the internet (Gartner, 2013). Secondly, the heterogeneous connectivity as well as the traffic patterns of IoT devices add complexity to the task of anomaly detection in IoT. Authors provide a survey of anomaly detection in IoT and they summarise the findings as follow:

1. *Distributed Approaches:* In this approach, devices collaboratively try to solve the problem of anomaly detection. Rule-based anomaly detection schemes are widely used in this approach. Rule-based anomaly detection includes cooperative detection, statistical anomaly detection, and support vector machine. The key advantages in distributed approaches in intrusion detection are (1) fast (2) scalable.

2. *Centralized Approaches:* compared to distributed approaches, centralized approaches are simple to implement and cost IoT devices less computation power. Reputation based and heuristic ranking algorithms are common approaches used in detecting malicious nodes in wireless sensor network WSN.

3. *Hierarchical Approaches*: The Internet is a hierarchical network, hence the hierarchical approach is regarded by the author to be the best approach to use in a large network. Clustering, statistical anomaly detection, and Markov model are common approached used in detecting anomalies of IoT.

4. Stand Alone Detection Approaches: This approach focuses on making each IoT node or device capable of detecting attacks through keeping a short-term dynamic statistical history of the event.

The above approaches focus on security of IoT devices from a network point of view. Thus, they can be used to detect network based attacks directed to IoT such as fake nodes, fake MAC addresses, fake routing, DoS and DDoS.

Butun et al. (2015) also provide a novel classification of the methodologies of attack detections dividing them into 3 types:

1. *Statistical*
2. *Data-mining*
3. *Artificial intelligence AI.*

All these methods require either learning from pervious data samples or statistical analysis of data. State machines, clustering, description languages and expert systems are widely used techniques in the *data mining* type. Fuzzy, genetic algorithm, Bayesian, Markov and neural networks are widely used techniques in AI.

Table 2.1 provides a summary of the related work. The way to approach the table is through realizing that same techniques of anomaly detection in case of non-IoT network can be used for the same purpose in IoT network, with special attention paid to the unique characteristics of IoT – heterogeneous devices, heterogeneous data, limited computing resources, that dictate using lighter version of these techniques in anomaly detection.

Table 2.1: Table of related work.

Author(s)	Method/model	Accuracy	
Golman (2014)	Support Vector Machine (SVM) and C5.0 for anomaly detection.	99.78%	Non-IoT
Hajare (2016)	Intrusion detection model based on big data analysis.	Not reported	

Tanpure et al. (2016)	Hybrid intrusion detection system using two data mining techniques; K-means and Naïve Bayes.	Not reported	
Hodo et al. (2016)	Artificial neural network intrusion detection system for IoT attacks.	99.4%	IoT-based
Pajouh et a. (2016)	Two-tier Classification Model for Anomaly Detection in IoT Backbone Networks	84.82%	
Fu et al. (2011)	Unsupervised classification of normal and abnormal anomalies in IoT	Not reported	
Liu and Wu (2014)	Lightweight unsupervised anomaly detection using Euclidean distance and Jaccrad coefficient in IoT	Not reported	

1.4. Summary

In this chapter, the IoT has been defined, and the architecture of IoT systems explained. Security is a key part in each layer in IoT architecture since it ensures availability, integrity and authenticity of each object as well as the data the objects exchange in IoT networks starting from device layer all the way up to the application layer.

ML was also discussed in this chapter. It was seen how ML can be divided based on learning style into three types; supervised learning, unsupervised learning and reinforcement leaning. ML can also be divided based on learning algorithms into many types such as regression algorithms, clustering algorithms, decision tree algorithms, deep learning and artificial neural network algorithms.

In this chapter also, an overview about the work related to anomaly detection in general and anomaly detection in IoT was given. It was observed that several statistical, data mining and AI techniques are used in detecting anomalies in IoT and non-IoT. A mix of clustering and classification techniques is commonly used among the researchers to find anomalies. They two techniques are used to cluster label data – as in supervised learning, or unlabelled data – as in unsupervised learning, into groups. Clustering is followed by a classification technique to identify normal and abnormal data. Such activities take place during training phase. Afterword, testing based on rules takes place. In testing, the model for anomalies detection is tested using data other than the one used in training phase, and outcomes are evaluated.

Designing anomaly detection schemes for IoT take similar approach as in designing anomaly detection schemes for other devices. However, one has to consider the unique challenges that IoT has. For instance, intrusive detection schemes that require significantly high computing power or memory consumption do not suit IoT. Furthermore, the IoT devices are heterogeneous and generate heterogeneous data that mount to big data. Thus the approaches to detect anomalies in IoT should not be limited to detecting network attacks, instead, they should go deeper and detect anomalies in data or the payloads carried in network traffic.

Chapter 3: Methodology

This chapter describes the methodology; the way by which the model for anomalies detection in IoT using a combination of two algorithms; inverse weight clustering (IWC) and decision tree (DT) algorithms. It includes the description of data to be used in developing and testing the model, as well as the description of how IWC and DT work in clustering and classifying data. It also describes the evaluation criteria; the criteria by which the accuracy of the model in detecting anomalies in IoT can be measured.

3.1. Model Design

The purpose of the proposed anomalies detection model in IoT is to detect anomalies in data that is being exchanged by IoT devices at the application layer of IoT architected as described in the previous chapter. In order to do so, the model requires designing a process that takes data from IoT devices as an input, systematically process input data, and produce predictions of two folds; "normal" or "abnormal". Figure 3.1 provides a high level view of the model design.

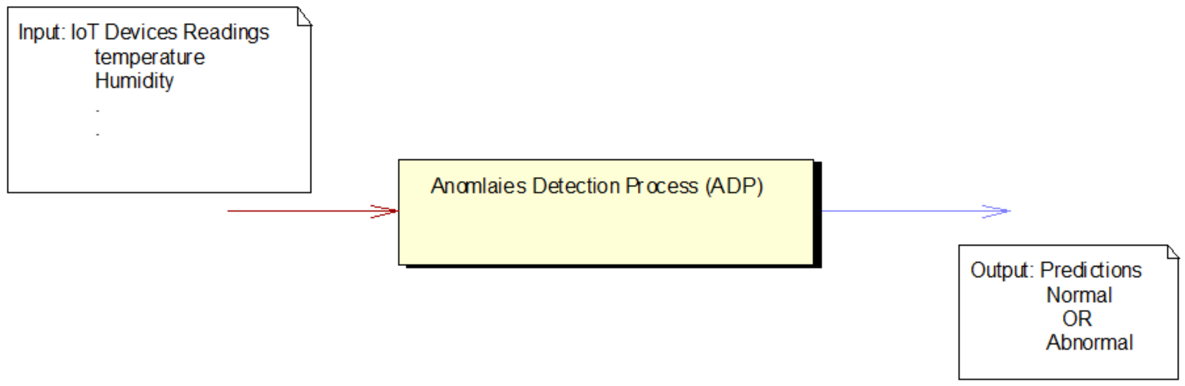

Input: IoT Devices Readings
temperature
Humidity
.
.
Anomlaies Detection Process (ADP)
Output: Predictions
Normal
OR
Abnormal

Figure 3.1: Anomalies Detection Process (ADP) - high level view.

Anomalies Detection process (ADP) is the process that is responsible for clustering, classification, and producing predictions. It consists of three stages:

1. Input data preprocessing: this stage is concerned with formatting the input data in a way that makes it easy for ADP to process it.
2. Processing: this stage is concerned with clustering input data into groups using IWC and building DT for classification.

3. Predictions: this stage is concerned with using the already built DT for predictions.

Figure 3.2 depicts ADP in more depths. The input data goes into the preprocessing stage and then used by IWC to build two clusters; one cluster combines normal data and the other one combines abnormal data. Clustering takes place based on similarities between data attributes.

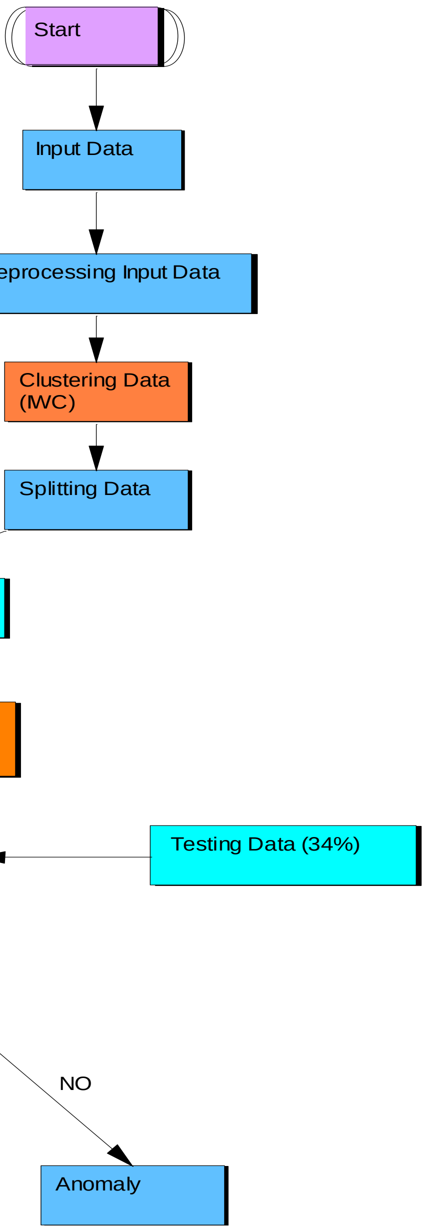


Figure 3.2:The flow chart of the proposed intrusion detection model.

After clustering, the preprocessed input data is split into two groups; one group is used for training and building the decision tree, and the other group is used for testing the accuracy of the decision tree in distinguishing between normal and abnormal data.

3.2. Input data

When it comes to input data for intrusion detection in IoT, it is very hard to obtain labeled data. Fu et al. (2011) points out this challenge, which most independent researchers in the field of intrusion detection in IoT face. On the other hand, there are plenty of non-IoT dataset available freely for researchers who are working on intrusion detection such as the one published by MoD; KDD Cup 1999 dataset. The dataset consists of millions of records collected from military network environment and dedicated primarily for network intrusion detection researches.

In order to overcome such challenges, unlabeled IoT dataset from Intel Labe was acquired. The dataset contains more than 2.3 million readings collected from 54 sensors in Intel Berkeley Research lab in 2004 (Madden, 2004). The raw dataset can be downloaded from: http://db.csail.mit.edu/labdata/labdata.html.

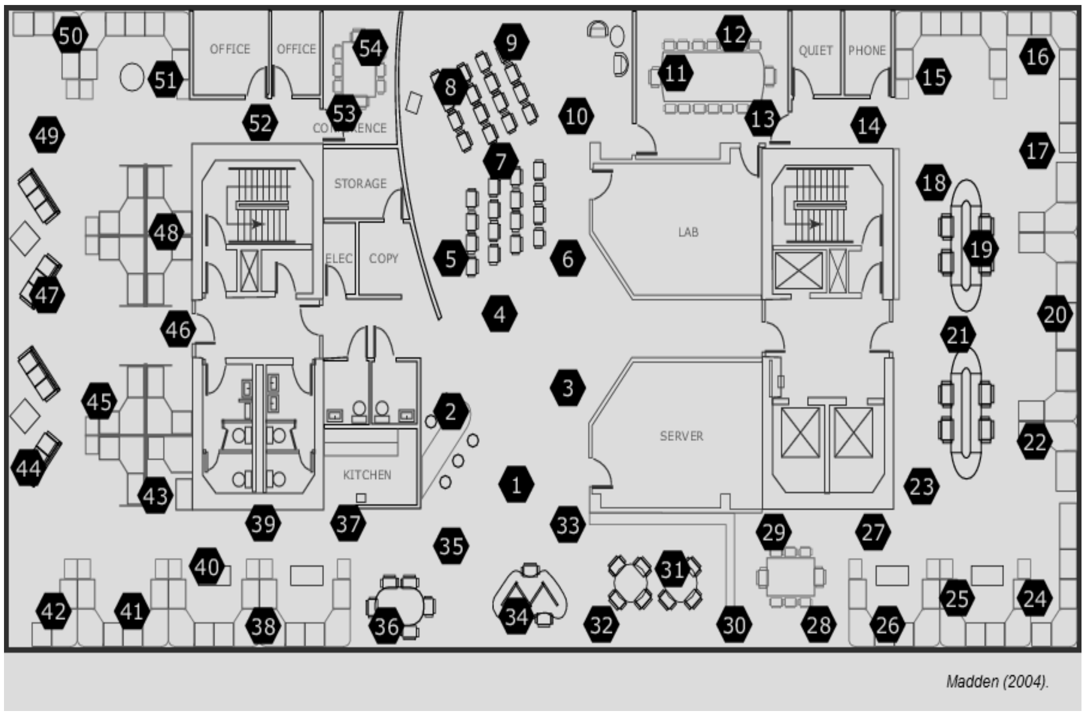


Figure 3.3: Sensors distribution at Intel Berkeley Research lab.

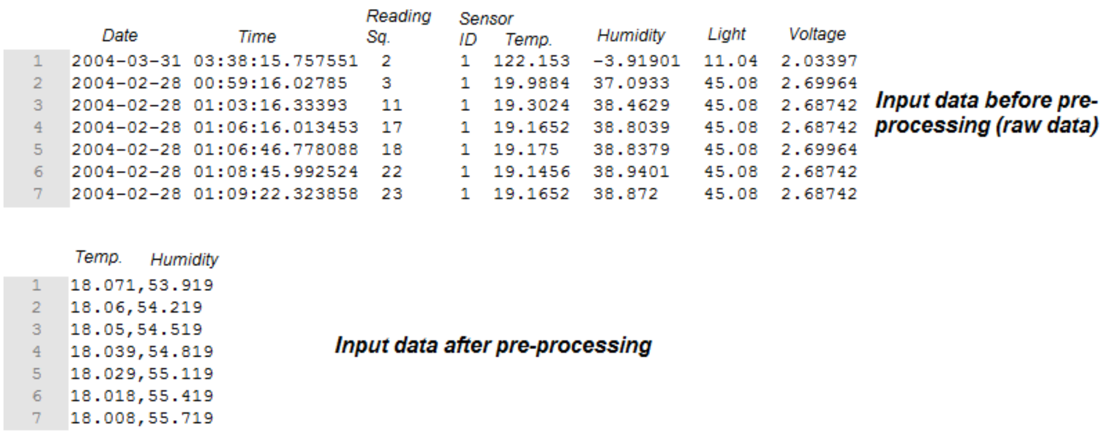
Figure 3.3 depicts the location of the 54 sensors in lab. Each one of these sensors can collect data about temperature, humidity, light, and voltage and transmit this data to the gateway every 31 seconds. Each reading is distinguished by date, time, and sender sensor ID (Madden, 2004).

While this dataset is freely available, it is not primarily dedicated to intrusion detection research hence it is not labeled datum. It, therefore, requires processing in order to use it to develop the proposed anomaly detection model in this research.

3.3 Data pre-processing

Processing more than 2.3 million readings of 54 sensors can be computationally expensive. Therefore, a random sample of readings was taken from the 2.3 million readings and placed in .csv Microsoft Excel file named “inputData.csv”. In addition to this, unnecessary attributes (i.e. columns) such as date and time of reading, sensor ID, light and voltage were data that were deleted since the proposed anomalies detection model is concerned with detecting anomalies at applications layer.

Moreover, anomalies were added to the input data manually (i.e. 454 records). Fu et al. (2011) used the same way to manipulate input data by manually adding records with anomalies in the dataset. The benefits of taking such an approach are; (1) the input data will be somehow labeled and known to the teacher, (2) the input dataset will become qualified dataset for anomalies detection and fit the purpose of the research. (3) overcoming the issue of obtaining labeled intrusion dataset for IoT at an expensive cost.


Input data before pre-processing (raw data)
Input data after pre-processing

	Date	Time	Reading Sq.	Sensor ID	Temp.	Humidity	Light	Voltage
1	2004-03-31	03:38:15.757551	2	1	122.153	-3.91901	11.04	2.03397
2	2004-02-28	00:59:16.02785	3	1	19.9884	37.0933	45.08	2.69964
3	2004-02-28	01:03:16.33393	11	1	19.3024	38.4629	45.08	2.68742
4	2004-02-28	01:06:16.013453	17	1	19.1652	38.8039	45.08	2.68742
5	2004-02-28	01:06:46.778088	18	1	19.175	38.8379	45.08	2.69964
6	2004-02-28	01:08:45.992524	22	1	19.1456	38.9401	45.08	2.68742
7	2004-02-28	01:09:22.323858	23	1	19.1652	38.872	45.08	2.68742

```
  Temp.  Humidity
1 18.071,53.919
2 18.06,54.219
3 18.05,54.519
4 18.039,54.819
5 18.029,55.119
6 18.018,55.419
7 18.008,55.719
```

Figure 3.4: Input data before and after pre-processing.

The resulting input dataset would be made of two attributes (i.e. columns); temperature and humidity, and comprising of 7526 records. The value of each attribute in each record is real as depicted in figure 3.4.

3.3. Clustering

According to Aggarwal and Reddy (2013), clustering or cluster analysis refers to splitting data into groups in such a way that elements in each group are similar to each other rather than the elements in other groups. Each group is known as a cluster.

There are several types of clustering approaches; centroid-based clustering, distribution-based clustering, density based clustering, etc. (Aggarwal and Reddy, 2013). Centroid-based clustering is commonly used in intrusion detection application since it represents each cluster of data by a central vector called “Centroid”. The centroid is not necessarily a member of the cluster data. It comes as result of applying clustering algorithm on the data.

K-means is by far the most popular clustering algorithm (Haq et al., 2015). Given the number of clusters to be produced, the algorithm can produce centroids. For example, it the input data is supposed to be clustered into two groups – in this case k equals to 2, K-means will randomly pick at two records from input data and start to fetch the other records in the same dataset and compare them with the 2 randomly selected records by computing the distance between them (Aggarwal and Reddy, 2013).

When k-means computes the distance between records, a short distance indicates that the fetched record is most likely to be similar to one of the randomly chosen records at the beginning of the clustering operation. One the algorithm finishes comparing all the records; it produces k centroids representing k clusters. At this stage, the clusters are not labeled. Therefore, the teacher has to find a way to label these clusters. Figure 3.5 depicts clustered dataset into 3 groups using k-means.

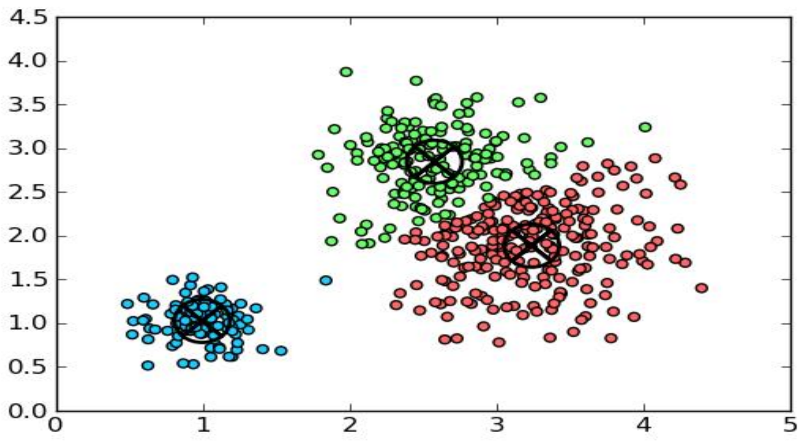


Figure 3.5: Example of K-means clustering.

Ashour and Fyfe (2007) pointed out one of disadvantages of K-means. Since the teacher is most likely do not know which records or k vectors best represent the data, and that the k-means randomly chooses these records, this means the centroid that the algorithm produces at the end of clustering might not be accurate. Therefore, they proposed the Inverse Weight Clustering (IWC) algorithm to address this critical defect in K-means. IWC in essence is built upon k-means algorithm. However, it relies on running the k-means many times until the centroids and clusters become stable. In other words, while the k-means stops once k-centroids and clusters are formulated, IWC takes the resulting centroids and rerun k-means over the same data by computing the distance between each record and the centroids.

In K-means the centroids can be found using the following formula:

$$\mathbf{m}_k = \frac{\sum_n r_{kn}\mathbf{x}_n}{\sum_{j,n} r_{jn}} \qquad (1)$$

$$\text{where e.g. } r_{kn} = \frac{\exp(-\beta d(\mathbf{x}_n, \mathbf{m}_k))}{\sum_j \exp(-\beta d(\mathbf{x}_n, \mathbf{m}_j))} \qquad (2)$$

d(a,b) is the Euclidean distance between a and b. IWC expand K-means as following:

$$J_I = \sum_{i=1}^{N} \sum_{k=1}^{K} \frac{1}{\| \mathbf{x}_i - \mathbf{m}_k \|^P} \tag{3}$$

$$\frac{\partial J_I}{\partial \mathbf{m}_k} = \sum_{i=1}^{N} P(\mathbf{x}_i - \mathbf{m}_k) \frac{1}{\| \mathbf{x}_i - \mathbf{m}_k \|^{P+2}} \tag{4}$$

$$\begin{aligned}
\frac{\partial J_I}{\partial \mathbf{m}_k} &= 0 \Longrightarrow \\
\mathbf{m}_k &= \frac{\sum_{i=1}^{N} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^{P+2}} \mathbf{x}_i}{\sum_{i=1}^{N} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^{P+2}}} \\
&= \frac{\sum_{i=1}^{N} b_{ik} \mathbf{x}_i}{\sum_{i=1}^{N} b_{ik}} \qquad (5)
\end{aligned}$$

where

$$b_{ik} = \frac{1}{\| \mathbf{x}_i - \mathbf{m}_k \|^{P+2}} \tag{6}$$

The way by which k-means can be enhanced to become less susceptible to the selection of initial centroid can be done by partially derive J_I with respect m_k as shown in equation 4. This will produce centroids that best represent the data as shown in figure 3.6.

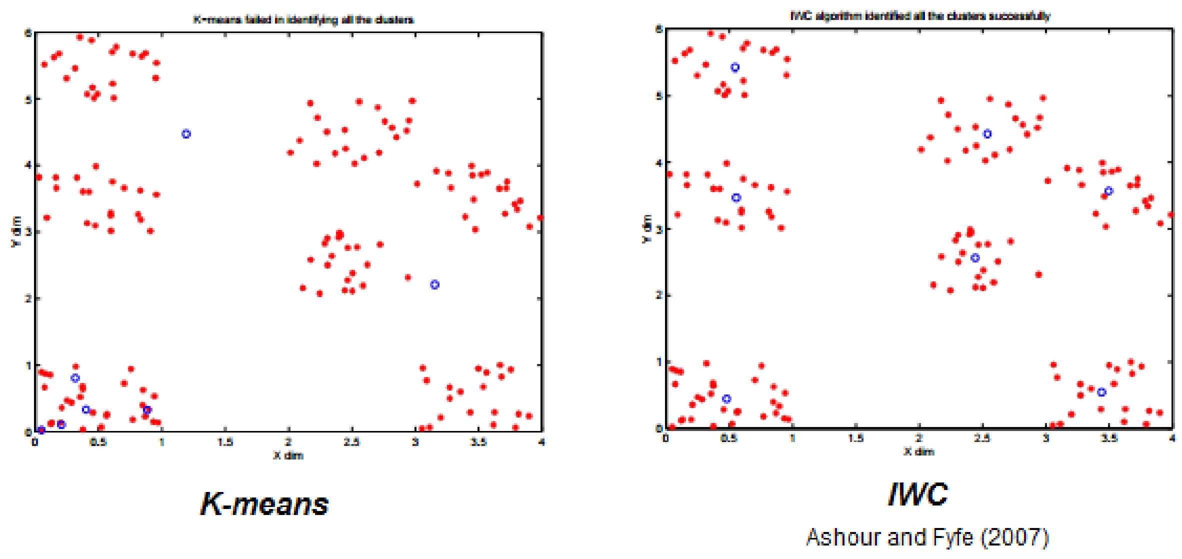


Figure 3.6: IWC vs K-means in clustering data.

The above equations for IWC as proposed by Ashour and Fyfe (2007) can scientifically improve k-means clustering. Therefore, these equations are going to be implemented and used for clustering input data in the proposed anomalies detection model for IoT in order to produce centroids that represent input data in the best way.

3.4. Classification

Decision tree (DT) algorithms are commonly used in classification (Quinlan, 2014). DT can be perceived as predication model that maps observations about an element to conclusion in a tree structure made of nodes and branches as shown in figure 3.7. Each leave in the tree represents a class. In the case of the proposed model, there are two classes; normal and abnormal. These classes are extracted as a result of data clustering using IWC. Each branch represents a conjunction that leads to class label (Quinlan, 2014).

C4.5 is an algorithm that can be used to generate the DT. It was proposed by Ross Quinlan. In order for C4.5 to be able to build DT, it has to be given a set of training data of already classified samples. Therefore, the input dataset used in clustering can split into two parts:

1. Training dataset: which contains 4990 records that are classified to normal and abnormal.
2. Testing dataset: which contains 2536 records without labels so that it can be used for testing the DT-C4.5.

Using the training dataset, C4.5 begins to build the tree. A node in the tree consists of actual value of temperature and humidity. Based on normalization theory, C4.5 will start to create other leaves in the trees that can connect all the attributes with each other’s using normal and abnormal classes to create branches using if statements. In other words, if the value of a given attribute is x, and then the class would be normal or abnormal based on information gain (Quinlan, 2014).

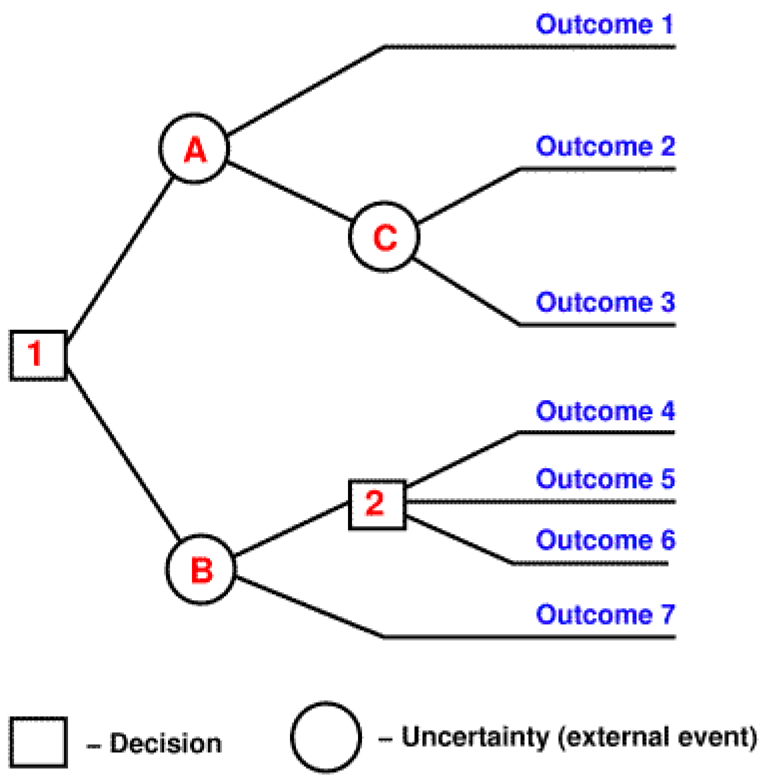
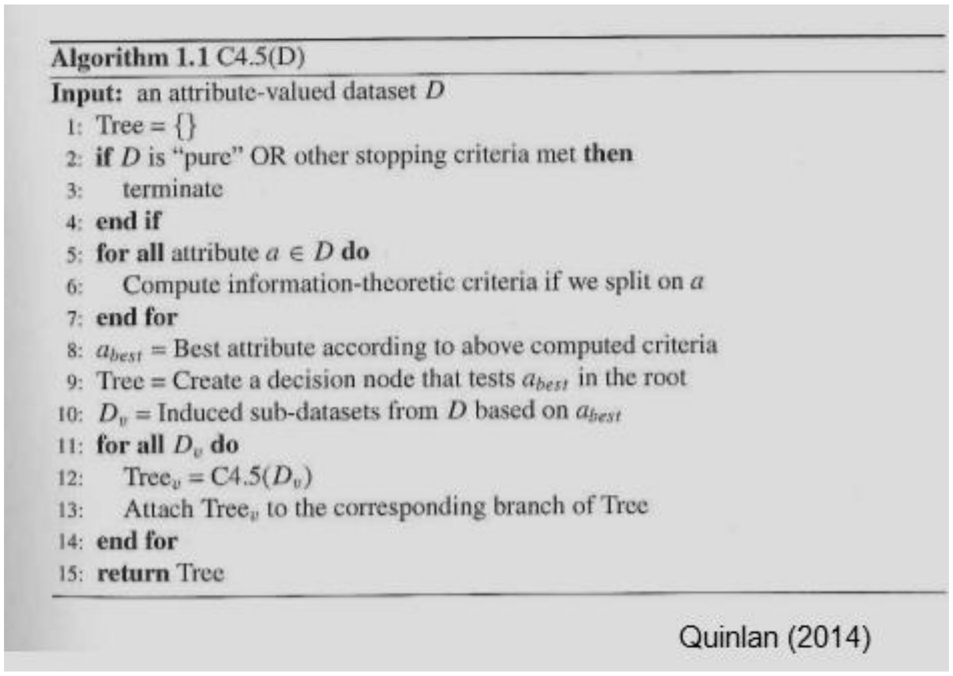


Figure 3.7: Decision Tree.

The pseudocode of C4.5 algorithms is shown in figure 3.8. The algorithm accepts training dataset of classified data. It first checks if the termination criteria are satisfied or not. If yes, the algorithm terminates. If no, the algorithm computes information-theoretic criteria for all attributes. It then chooses the best attribute using information-theoretic criteria. Then, it creates a decision node based on the best attribute which will be the first node in the tree (Quinlan, 2014).

From the first node, the algorithm starts to split the training dataset based on the first node into further nodes and processes each attribute recursively from the lowest level in the tree to the tope node keeping updating all the node until termination criteria is met (Quinlan, 2014).



```
Algorithm 1.1 C4.5(D)
Input: an attribute-valued dataset $D$
1: Tree = {}
2: if $D$ is "pure" OR other stopping criteria met then
3:     terminate
4: end if
5: for all attribute $a \in D$ do
6:     Compute information-theoretic criteria if we split on $a$
7: end for
8: $a_{best}$ = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests $a_{best}$ in the root
10: $D_v$ = Induced sub-datasets from $D$ based on $a_{best}$
11: for all $D_v$ do
12:     Tree$_v$ = C4.5($D_v$)
13:     Attach Tree$_v$ to the corresponding branch of Tree
14: end for
15: return Tree
```

Quinlan (2014)

Figure 3.8: C4.5 algorithms.

3.5. Evaluation Method

Confusion matrix is the technique that is usually used to describe and characterize the performance of the classification model in intrusion detection system (Kurniawan, Rosmansyah, and Dabarsyah, 2015). The matrix is simple, but contains confusing items; true positive (TP), true negative (TN), false positive (FP) and false negative (FN). It is 2x2 dimensional matrix and represent the relationship between the actual value and the predicted value as show in figure 3.9.

If the classifier predicts that there are issues or anomalies in data and that the data is actually abnormal, then the attempt is said to be true positive (TP). In the case of the proposed anomalies detection model, if the model predicts that a given test data or record as an anomaly and the data or record is actually being an anomaly, then this can be used as an indicator to TP rate of the proposed system.

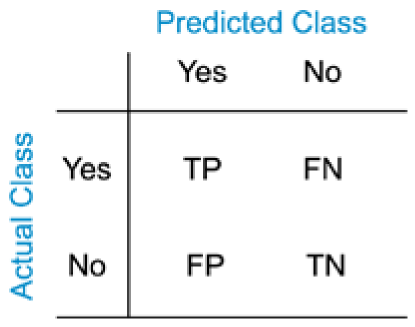


Figure 3.9: Confusion matrix.

If the classifier predicts that there is no issue in data and that that data is actually normal, then the attempt is said to be true negative (TN). In the case of the proposed anomalies detection model, if the model predicts that a given test data or record as normal and the data or record is actually being normal, then this can be used as an indicator to TN rate of the proposed system.

If the classifier predicts that there is an issue or anomaly in data while the data is actually normal and does not contain an issue or anomaly, then the attempt is said to be false positive (FP). In the case of the proposed anomalies detection model, if the model predicts that a given test data or record has issue or anomaly while it is not, then this can be used as an indicator to FP rate of the proposed system. Most of anomalies detection models hope to reduce the FP to the lowest level.

Finally, If the classifier predicts that there is no issue or anomaly in data while the data actually has an issue or anomaly, then the attempt is said to be false negative (FN). In the case of the proposed anomalies detection model, if the model predicts that a given test data or record has no issue or anomaly while it has, then this can be used as an indicator to FN rate of the proposed system. In anomaly detection models, this rate indicates the level of failure in the model capability to deliver the intended function – anomalies detection.

With this in mind, the overall accuracy – how often is the classifier correct, of the system can be computed using the following formula:

$$\frac{TP + TN}{x}$$

Where x is the total number of records given as an input to the classification model. In the case of this research x equals the number of records in testing data; 2,536.

The misclassification rate of the model – how often is the classifier wrong, can be computed using the following formula:

$$\frac{FP + FN}{x}$$

The TP rate (aka. Sensitivity or Recall) of the model can be computed using the following formula:

$$\frac{TP}{number\ of\ actual\ abnormal\ records\ in\ x}$$

The FP rate of the model can be computed using the following formula:

$$\frac{FP}{number\ of\ actual\ normal\ records\ in\ x}$$

Specificity of the model or TN rate can be computed using the following formula:

$$\frac{TN}{number\ of\ actual\ normal\ records\ in\ x}$$

Precision of the model can be computed using the following formula:

$$\frac{TP}{number\ of\ prediced\ abnormal\ records\ in\ x}$$

Finally, the prevalence of the model can be computed using the following formula:

$$\frac{actual\ number\ of\ abnormal\ records\ in\ x}{x}$$

These metrics are going to be used to evaluate the proposed anomalies detection model and compare the results with other related works.

Chapter 4: Implementation and Results

This chapter describes the implementation of the proposed model for anomalies detection in IoT. The model was implemented and tested using MATLAB. The implementation of the model consists of three stages; input data pre-processing, clustering, classification and testing. Input data pre-processing is a stage for qualifying Intel Labe IoT dataset to be used as an input to anomalies detection process (ADP). Clustering and classification are the core sub-processes in ADP and the model in general. Clustering divides input data into two groups of data that has almost similar structure and features using Invers Weight Clustering (IWC) algorithm. Classification sub-process uses C4.5 algorithms for building decision tree (DT) that is used for classifying data into normal and abnormal.

This chapter also provides the results of running the implementation of the proposed model for anomalies detection in IoT. MATLAB is used to run the test using part of unclassified input data and obtaining the results.

4.1. Implementation

The proposed model for anomalies detection in IoT consists of six files. The files are described in table 4.1 and **attached** to the appendix.

Table 4.1: The files that make the proposed model for anomalies detection in IoT.

File Name	Description
IoT_Clustering.m	This MATLAB file implements clustering sub-process in ADP. It accepts the pre-processed input data file in a form of a .csv file as an input, converts input data into matrix, calls IWK.m to cluster input data into two groups and produces centroids, and produces clustered data in a form of .csv file. A .csv are Microsoft Excel file that MATLAB can process.
IWK.m	This MATLAB file represents the implementation of IWC as proposed by Ashour and Fyfe (2007). It accepts input data in a form of matrix, derive the centroids based

	on the number of clusters K, and return the resulting centroids in a form of matrix.
IoT_Classification.m	This MATLAB file implements classification sub-process in ADP. It accepts the clustered data in a form of matrix, and splits it into two groups, training and testing groups. The training group is used to build the decision tree.
C4_5.m	This MATLAB file implements C4.5 DT algorithm as proposed by (Quinlan, 2014). It accepts a matrix that consists of training data, a matrix of cluster IDs that identifies each record or row in the training matrix, and the number of nodes to start the tree with. It uses make_tree.m file to build the DT based on information-theoretic criteria. C4_5.m returns the decision tree.
make_tree.m	This MATLAB file builds the tree.
use_tree.m	This MATLA file uses the DT that is built by C4_5.m and make_tree.m files and applies it to testing data matrix. It also compares the results of classifying input data with the cluster ID for each record in testing dataset to produce results. The results are given in the form of true positive, true negative, false positive and false negative form.

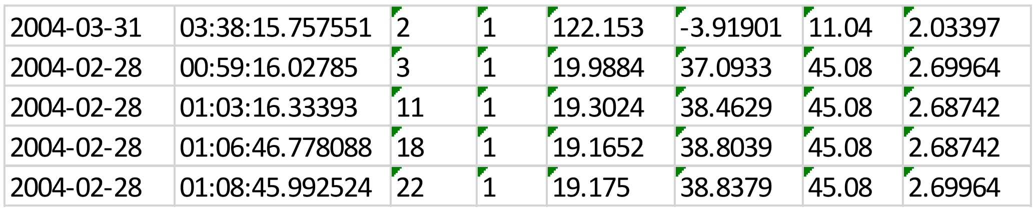
MATLAB is used to implement and test the proposed model for anomalies detection in IoT. It is relatively simple and easy to use compared to other high level programming language such as Java or C. Moreover, matrix is a basic element in MATLAB. A simple integer value is considered a matrix of one row and one column. In applying this to the types of input data used in anomalies detection filed, the data is usually presented in a form of matrix.

Additionally, MATLAB is fast in running and processing data of large size. It automatically distributes work over the processing cores within the machine without the need for providing optimization or distributed processing parameters as in the case of other programming language (Attaway, 2013).

4.1.1. Input Data-prepressing

The Intel Lab IoT dataset – described in the previous chapter, consists of 2.3 million readings of 54 sensors distributed in several locations in Intel Berkeley Research lab. The sensors collect 4 types of sensory data; temperature, humidity, light and voltage.

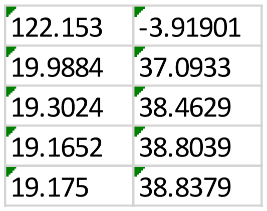
Once downloaded, the dataset comes in text format. It can be opened using Microsoft Excel. The data would look like the data shown in figure 4.1. The first 4 columns in this data represent date and time of data collection, reading and node ID respectively. These columns or attributes are not needed. Therefore, these attributes will be detected. The last 4 columns or attributes represent temperature, humidity, light and voltage respectively. The first two attributes are needed in the proposed model for anomalies detection in IoT. The other two attributes can be processed as part of future work.



2004-03-31	03:38:15.757551	2	1	122.153	-3.91901	11.04	2.03397
2004-02-28	00:59:16.02785	3	1	19.9884	37.0933	45.08	2.69964
2004-02-28	01:03:16.33393	11	1	19.3024	38.4629	45.08	2.68742
2004-02-28	01:06:46.778088	18	1	19.1652	38.8039	45.08	2.68742
2004-02-28	01:08:45.992524	22	1	19.175	38.8379	45.08	2.69964

Figure 4.1: A sample of raw data as obtained from Intel Lab.

After removing the unneeded columns and data from Intel Lab dataset, the new dataset would look like the data shown in figure 4.2, a dataset with two columns or attributes that represent temperature and humidity. Out of 2.3 million readings, 7,072 readings were chosen randomly to make input dataset. Then 454 readings with anomalies were added to the input dataset. Therefore, the input dataset would consist of 7,526 readings.



122.153	-3.91901
19.9884	37.0933
19.3024	38.4629
19.1652	38.8039
19.175	38.8379

Figure 4.2: A sample of input data after pre-processing.

4.1.2. IWC Algorithm Implementation

Invers Weight Clustering (IWC) algorithm is implanted in the IWC.m file. The method is implemented as a MATLAB function that accepts two inputs;

1. Matrix of input data for clustering
2. Number of clusters that the algorithm is supposed to cluster the input data to.

The function returns a matrix of clustered data, and the interface of the function is given using the following line of code (Line 4: IWC.m)

```
function [m] = IWK(x,K)
```

The model is implemented to reach IWC function from IoT_Clustering.m. This file is implanted to read input data file in .csv format and convert it into 7526x2 matrix using the csvread() function that is provided by MATLAB to automatically read .csv files and convert them to matrix. The following lines of code shows how input data file is read and converted to matrix (Line1 and 3: IoT_Clustering.m):

```
filename = 'inputData.csv';

M = csvread(filename);
```

After the MATLAB processes the input data Excel file to a matrix, the data will be held in the system memory and ready for clustering. For the purpose of simplicity, the input data matrix will look like the data shown in figure 4.3.

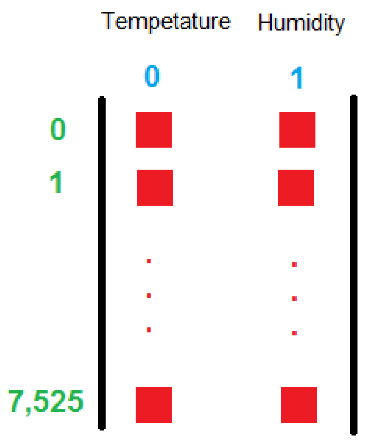


Figure 4.3: Input data matrix before clustering.

The data matrix depicted in figure 4.3 will be passed to IWC function along with a number of clusters to be created using the following line of code (Line 8: IoT_Clustering.m):

C = IWK(M,K);

Where M is the input data matrix and K equals 2, the number of clusters that IWC algorithm is instructed to create. The IWC will randomly pick up two rows from input matrix and start to compare other rows with them by computing the Euclidean distance between the rows to build the clusters and derive the centroids. IWC was implanted to run this operation 10 times before choosing the optimal centroid that best represents the two clusters of input data through normalization (Line 20-85: IWC.m). IWC returns a matrix of one attribute; the centroid and two rows, which represents the values of the two derived centroids.

This matrix is returned IoT_Clustering.m, which will use the same technique in normalizing data by measuring the distance between each record in input data matrix and the two centroids. Based on the distance, IoT_Clustering.m will attempt to recreate the input data matrix and place a cluster label next to each row. The label can be either “1” or “2”. “1” indicates that this record belongs to cluster 1 and “2” indicates that this record belongs to cluster 2. The result will be clustered data matrix that would look like the matrix depicted in figure 4.4.

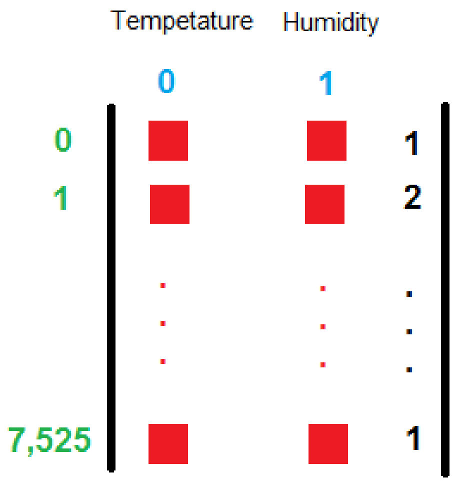


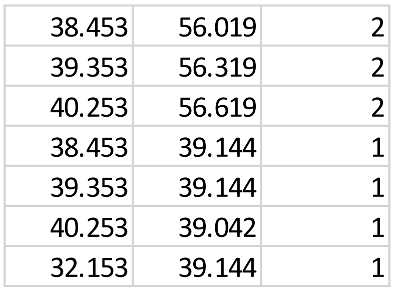
Figure 4.4: Input data matrix after clustering.

IoT_Clustering.m converts the clustered input data matrix into Excel file using the MATLAB function csvwrite(). This is done in the line following line of code (Line 27: IoT_Clustering.m):

```
csvwrite(reslutfile,R)
```

Where resultfile is a variable that contains the name of Excel file that is supposed to hold the clustered input data, and R is the name of the matrix that contains that clustered input data. The code was implemented to store data in an Excel file with the following name "clusteredData.csv". A sample of what a file would contain is depicted in figure 4.5.

The reason behind exporting cluster input data into Excel files is because it is going to be used in the next stage for building the decision tree (DT) and conducting the test.



38.453	56.019	2
39.353	56.319	2
40.253	56.619	2
38.453	39.144	1
39.353	39.144	1
40.253	39.042	1
32.153	39.144	1

Figure 4.5: A sample of input data after clustering.

4.1.3. Classifier Implementation - C4.5

C4.5 is a decision tree algorithm that is used to classify data based probabilities. IoT_Classification.m contains the implementation of the classification process, which involves the implantation of C4.5. Before building the decision tree, data is supposed to be divided into two groups; training group made up of 4,990 records and testing group made up of 2,536 records. The training group constitutes 66% of the clustered input data while testing data constitutes 34% of the clustered input data.

IoT_Classification.m reads the clustered data stored in Excel file using MATLAB csvread() function and store in a matrix as shown in the following lines of code (Lines 2-3: IoT_Classification.m):

```
Data = 'clusteredData.csv';

CData = csvread(Data);
```

The matrix that IoT_Classification.m would create is exactly similar to the one depicted in figure 4.4. However, IoT_Classification.m slices this matrix into two matrices; train data matrix and test data matrix. These two matrices do not contain the cluster ID of each row. The way that these matrices would look is depicted in figure 4.6. Moreover, they are created by the following lines of codes (Line 11 and 14: IoT_Classification.m):

```
train_data = CData(1:4990,1:2)'

test_data = CData(4991:7526,1:2)'
```

IoT_Classification.m slices clustered data input matrix into two further matrices; train target data matrix and test target data matrix. These two matrices are made of one attribute, which is the cluster ID of each row. The way that these matrices would look is depicted in figure 4.7. Moreover, they are created by the following lines of codes (Line 12 and 15: IoT_Classification.m):

```
train_targets = CData(1:4990,3)'

test_targets = CData(4991:7526,3)'
```

The """at the end of each matrix indicates that the matrix in put in reverse order as C4.5 requires input matrices to be given in reverse order while it builds the tree. Train data and target data matrices are sent to C4_5.m along with percentage of incorrectly assigned samples at a node. This is can be seen in IoT_Classification.m at line 17 of code:

```
[tree, discrete_dim] = C4_5(train_data, train_targets, 1)
```

For each row in the train data matrix, C4_5.m processes each record until it finds that number of nodes that the tree would contain. This is considered as termination criteria. Once this number is obtained, train data, train target data and maximum number of node in the tree are used to build the decision tree using make tree.m which is called from inside C4_5.m in line 21:

```
tree  = make_tree(train_data, train_targets, inc_node, discrete_dim, max(discrete_dim), 0);
```

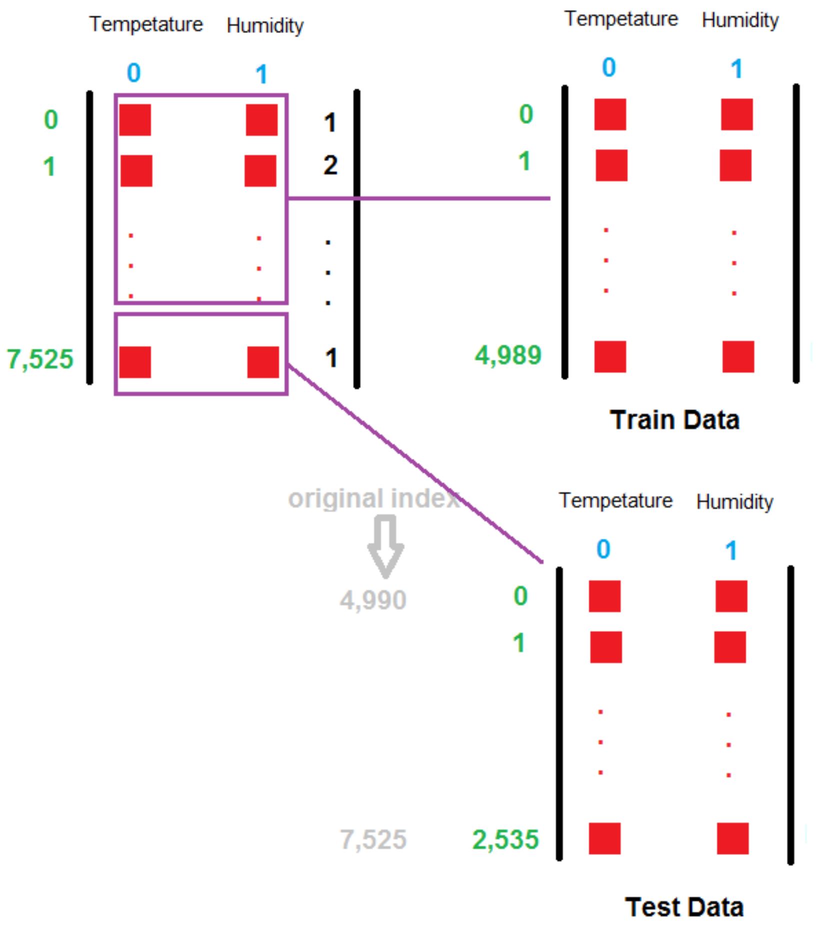


Figure 4.6: Splitting input matrix in train and test data.

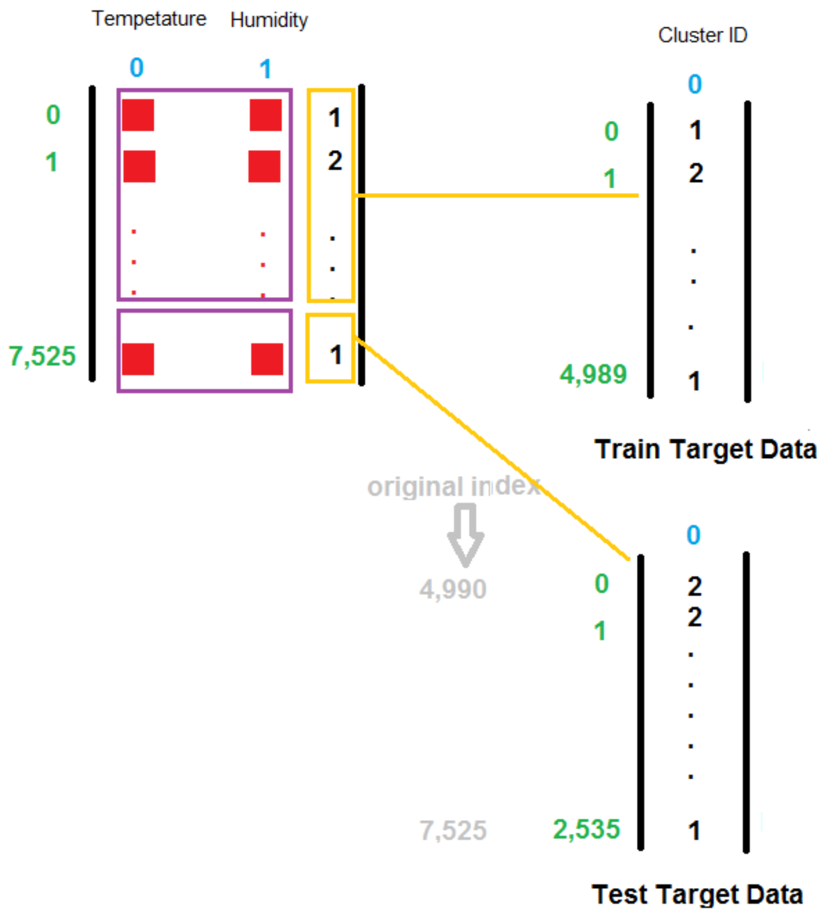


Figure 4.7: Splitting input matrix in train target and test target data.

4.2. Results

After running the model using MATLAB, results of clustering and classification of data are obtained. Figures 4.8 and 4.9 shows input data before and after clustering using IWC algorithm.

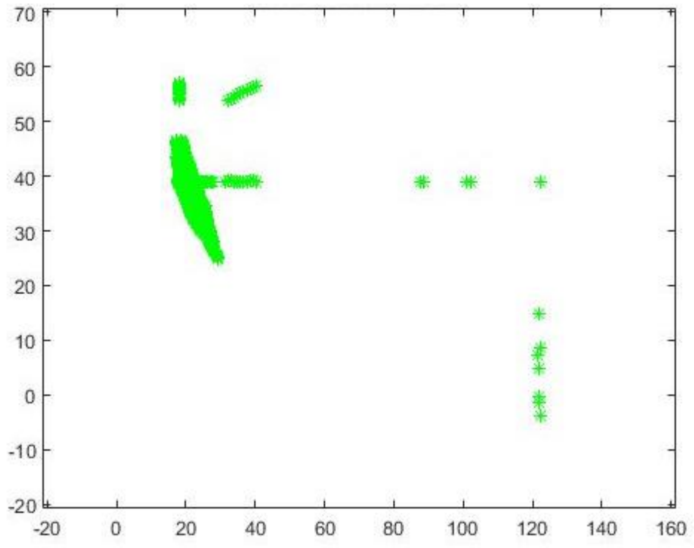


Figure 4.8: Input data before clustering.

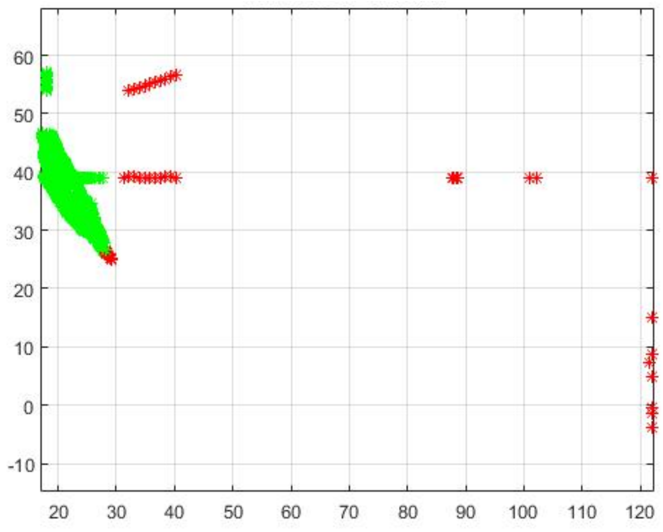


Figure 4.9: Input data after clustering.

Through figure 4.8 it can be seen that all data belongs to one class or cluster. After clustering input data using IWC algorithm, data was split into two classes, one shown in red which indicates that this data belongs to class 1, and the other data belongs to class 2 and shown in green in figure 4.9.

After running the classifier on test data which consists of 2536 rows, the model reported the evaluation parameters of confusion matrix; TP, TN, FP and FN. Table 4.2 shows the results of the proposed anomalies detection model on confusion matrix.

Table 4.2: Results of the proposed anomalies detection model on confusion matrix.

X= 2536	**Predicted anomalies**	**Predicted normality**
Actual anomalies (182)	**TP = 179**	**FN = 3**
Actual normality (2,354)	**FP = 51**	**TN = 2303**

Chapter 5: Evaluation

This chapter provides the evaluation of the proposed anomalies detection model in IoT. It contains the calculated performance metrics of the proposed model in the light of the results given in the previous chapter. The performance metrics are metrics that can be calculated using the number of correct and incorrect predictions made by the model, and include overall accuracy, misclassification rate, sensitivity, specificity, precision, and prevalence.

5.1. Analysis

The proposed model for anomalies detection in IoT using Inverse Weight Clustering (IWC) algorithm and C4.5 decision tree algorithm was tested using labeled testing dataset consisting of 2536 records (i.e. 34% of input dataset). Each record in testing dataset constitutes three values; temperature, humidity and cluster ID. The temperature and humidity are the readings that were originally obtained from the IoT sensors at Intel Lab (see chapter 3, section 3.2 and 3.3). The cluster ID is an integer value which can be either "1" or "2". These two values represent the cluster to which each reading (i.e. temperature and humidity) belongs. The cluster ID values were obtained as a result of running IWC algorithm against input dataset which consists of 7526 records.

C4.5 decision tree was trained using labeled training dataset consisting of 4990 records (i.e. 66% of input dataset). It then was run against testing dataset, but without giving it the cluster ID that shows to which cluster each record in the testing dataset belongs. In other word, C4.5 was run against testing dataset without labels. Since the number of records in the unlabeled testing dataset is 2536, the model produced 2536 predictions in a form of "1" or "2". The resulting predictions were comparted with the cluster ID of each record in the labeled testing dataset, and the results show:

- The total number of **predicted anomalies** that were **actually anomalies** was 179. Therefore, the true positive (TP) value of the model is:

$$TP = 179$$

- The total number of **predicted normalities** that were **actually normalities** was 2303. Therefore, the true negative (TN) value of the model is:

$$TN = 2303$$

- The total number of **predicted anomalies** that were **actually normalities** was 51. Therefore, the false positive (FP) value of the model is:

$$FP = 51$$

- The total number of the **predicted normalities** that were **actually anomalies** was 3. Therefore, the false negative (FN) value of the model is:

$$FN = 3$$

In order to be able to calculate the performance metrics of the proposed model such as overall accuracy, misclassification rate, sensitivity, specificity, precision, and prevalence, the following actual values are needed:

Let x refers to the total number of records in testing dataset, then

$$x = 2536$$

The actual number of normalities in the testing dataset is 2,354. Let A_{Normal} refers to actual number of normalities in the testing dataset, then

$$A_{Normal} = 2{,}354$$

The actual number of anomalies in the testing dataset is 182. Let $A_{Anomalies}$ refers to actual number of anomalies in the testing dataset, then

$$A_{Anomalies} = 182$$

Therefore, the overall accuracy of the proposed model is producing correct predictions is 0.97.

$$\frac{TP + TN}{x} = \frac{179 + 2{,}303}{2{,}536} = 0.97$$

The misclassification rate of the proposed model in producing wrong predictions is 0.021.

$$\frac{FP + FN}{x} = \frac{51 + 3}{2{,}536} = 0.021$$

The sensitivity or recall of the proposed model in detecting anomalies is 0.98.

$$\frac{TP}{A_{Anomalies}} = \frac{179}{182} = 0.983$$

The false positive rate of the proposed model is 0.022.

$$\frac{FP}{A_{Normalities}} = \frac{51}{2,354} = 0.021$$

The specificity of the proposed model in accurately identifying normal readings is 0.97.

$$\frac{TN}{A_{Normalities}} = \frac{2,303}{2,354} = 0.97$$

The precision of the proposed model is raising alerts upon findings anomalies is 0.78.

$$\frac{TP}{TP + FP} = \frac{179}{179 + 51} = 0.78$$

The prevalence of the proposed model is 0.072.

$$\frac{A_{Anomalies}}{x} = \frac{182}{2,536} = 0.072$$

5.2. Discussion

Anomalies detection models are characterized by their accuracy and false positive rate. This is not to say that the other metrics of performance are not important. However, the proposed mode for anomalies detection in IoT has shown 97% accuracy in correctly detecting normal and abnormal patterns in IoT data. This percentage of accuracy indicates that combining IWC with C4.5 can lead to producing a highly accurate anomalies detection model. Moreover, the using IWC for clustering input data and then for building C4.5 decision tree leads to obtaining accurate results more than using k-means with C4.5.

The centroids that IWC produces as a result of running k-means algorithm more than one time are much more accurate than the centroids that k-means would produce. These fine turned centroids play a key role in building the classifier at the stage of training. The more accurate the built classifier is; the less false alarms the model would produce. When the false alarm rate is high, the model might not be used since dealing with alarms that indicate anomalies while they are not is costly for systems administrators. The proposed model false positive rate is 2.1%. This value equals to the overall misclassification rate of the model, which constitutes a false negative rate.

In order to better understand the false negative rate which is one of the most concerning rates when dealing with anomalies detections model; reflection on model sensitivity or recall is important. The proposed model sensitivity is 98.3%. This means that the false negative rate of the model is 1.7% which represents the chance that anomalies will pass through the model undetected. This percentage is usually not revealed to model users. However, it is critical and depends on the level of the risk that IoT model owners would accept. While 1.7% is extremely low and there is no anomalies detection model which will be able to bring this percentage to zero, the criticality of data in IoT system determine whether this percentage can be accepted or not. If IoT devices exchange data similar to the data used in this research in manufacturing facilities, then such level of failure in detecting anomalies might be tolerable if these anomalies do not represent risk to human life. If such a level of false negative is not tolerable then the proposed model might not be used.

Another way to reveal the accuracy of the proposed model in detecting anomalies is by looking at prevalence metric. The actual percentage of anomalies in testing data was 7.2%. This value represents the prevalence. The model predicted 7% of the actual 7.2% anomalies. This means that despite the low prevalence of anomalies, the proposed model is able to locate and identify them within a large dataset and with high accuracy.

In comparison to other work from other researchers in the same field, the overall accuracy of the proposed model, which equals to 97% falls in the middle between Hodo et al (2016) and Pajouh et al (2016). Hodo et al (2016) used an artificial neural network to detect anomalies in IoT and achieved 99.4% overall accuracy, while Pajouh et al (2016) used two classifiers to detect anomalies in IoT backbone networks and achieved 84.82%. Using artificial neural networks is a computationally expensive model when compared to IWC and C4.5 combined. Therefore, Hodo et al (2016) might not be suitable for usage with host-based anomalies detection solution in IoT. Pajouh et al (2016) work is concerned with anomalies at a network level more than application level in IoT. However, this research shows that the number of classifiers that can be used in identifying anomalies might not be relevant in IoT environment as one classifier seems to be enough to obtain very highly accurate results.

Chapter 6: Conclusion

6.1. Research Overview

Internet of Things (IoT) is one of the fastest growing technologies in recent years. It is a technology by which billions of smart objects or devices known as “Things” can use several types of sensors to collect various type of data about themselves and/or the surrounding environment and share it with authorized parties to serve several purposes such as controlling and monitoring industrials facilities or improving business service or functions.

IoT appeared in 1999 when researchers at the Massachusetts Institute of Technology (MIT) were creating an Internet based network that would cover all things in the world to realize automatic identification of things through information sharing. Due to the high cost of semiconductors that IoT objects are normally made of as well as the high utilization of IPv4, the technology could not breakout.

However, with the advent of IPv6, which offers address space that allows the connecting of billions of devices to the Internet, and the decrease in semiconductors, IoT has come back. According to Gartner (2016), there are more than 3 billion devices connected to the Internet. The number will increase to 20 billion by 2020. While these statistics show how prevalent IoT is, researchers warn against the risk that is associated with these devices.

The prevalence of IoT objects everywhere, while making life easier, it also provides “bad guys” with unpreceded number of targets to attack. The loss that may result from attacking IoT objects range from low to high loss depending on the type of data that an IoT object possesses. Compromising an IoT object such as a CCTV that collects photos of a crime scene can lead to criminals escaping justice.

6.2. Problem Definition

The erythema of cyber-attacks against IoT systems has increased steadily. Since an IoT system consists of IoT objects, network, and applications, each one of these components is exposed to various types of attacks. Some attacks are physical while others logically target data while it is collected by IoT objects and/or it is being communicated. Some attacks may target default routing configurations inside these objects making them sending their data to unauthorized parties. Other attacks may constitute introducing fake IoT objects and make it send malicious data to all other objects and components in the network.

The literature review revealed that there are many ongoing researches that work on protecting IoT systems from network attacks at perception and network level of IoT systems. The perception level is the level at which an IoT object interacts with the subject environment to collect data, and push it over the network to the application level where most of the processing and mining of the collected data takes place.

Security solutions such as intrusion detection are used to minimize the risk to having any component or data in an information system being compromised. This means that despite so much research and security solutions that are being utilized every day to protect IoT systems, the chance of having an IoT object being compromised and sending abnormal data to be processed at application level still stands.

There has been a shortage of research that tries to find anomalies in IoT data at application level. Thus this research represents an attempt to fill this gap by adding another layer of protection to the already existing layers or the layers that are being developed for IoT systems.

6.3 Experimentation and Results

Machine learning is being extensively used in anomalies detection. It provides techniques by which a model for detection can be produced using training processes on data that is relevant to each subject area. This research proposes a novel approach for anomalies detection in IoT based on a combination of two robust machine learning algorithms; inverse weight clustering (IWC) and C4.5 decision tree algorithm. IWC is an enhanced version of k-means algorithm that can be used to effectively cluster data into groups based on similarities between this data. C4.5 is a decision tree algorithm that can be used to build a decision tree for classifying data.

In order to build the proposed anomalies detection model and meet the second objective in this research, an unlabeled dataset for IoT was obtained from Intel Lab. The data was preprocessed to contain only two attributes; temperature and humidity. The number of records in the dataset was also reduced to 7526 records. The dataset was clustered using IWC in to two groups; each group was identified by an ID with value of “1” or “2”.

The dataset was then split into two datasets; training dataset which contains labeled records and represents 66% of dataset, and testing data which contains unlabeled records and represents 34% of dataset. The training dataset was used by C4.5 to build the decision tree, which then applied to testing data.

The results show that the proposed system's overall accuracy in predicting anomalies and normalities in IoT is 97%, and the misclassification rate is 2.1%. The sensitivity of the system in detecting anomalies exceeds 98% and the false positive rate is 2.1%. The results were derived from the confusion matrix that was proposed in chapter 3 to meet the fourth object of this research.

6.4. Contribution and Impact

The key contribution of this research is the successful and accurate combination of IWC and C4.5 in detecting anomalies in IoT data at an application level layer. The proposed model provides great anomalies detection solution in IoT data at application level for all researchers and companies to make use of it's real work since it is a lightweight model when compared to other anomalies detection models for IoT. Other models use artificial intelligent models that are not significantly more accurate than the proposed model in this research, they are significantly complex, computationally intrusive, and do not suit deployment as a IoT host based anomalies detection solution.

The proposed model, on the other hand, is a lightweight solution that is suitable for use as a host based anomalies detection solution at the level of IoT objects to inspect the collected data before sharing it, and at the level of the application layer in IoT systems.

The proposed model for anomalies detection has some limitations. Firstly, the model is valid for detecting anomalies in sensors that read and report temperature and humidity. In an environment where IoT sensors collect and share more than temperature and humidity, the proposed model cannot be used.

Secondly, since IWC is a modified version of k-means that is supposed to cluster data more effectively than k-means by iteration, the researchers who proposed the algorithm recommended configuring the IWC to iteratively run k-means many times to obtain optimal clusters. During the implementation, IWC was implemented to run k-means ten times. This means that the obtained clusters might not be optimal, and hence this might justify the 2.1% misclassification rate in the proposed model.

Finally, the classifier used in this research to build a decision tree is not the newest version of the classifier. C4.5 was used instead of C5.0. This may also contribute to the 2.1% misclassification rate in the proposed model.

6.5 Recommendation and Future Work

I recommend using the proposed anomalies detection model in IoT systems which tolerate a misclassification rate of 2.1%, where having few anomalies remain undetected do not lead to significant loss in human lives, data and assets.

The limitations that were mentioned previously are going to be part of future work. IWC is going to be reinvestigated to determine the optimal number of running k-means within IWC and obtaining optimal clusters. Moreover, C5.0 is going to replace C4.5. Finally, the model will be augmented to cover more attributes in data and become capable of detecting anomalies in more than temperature and humidity readings.

Bibliography

Aggarwal, C. C. & Reddy, C. K. (2013) Data Clustering: Algorithms and Applications. Chapman and Hall/CRC.

Alexander, D. & Finch, A. (2014). Information Security Management Principles (2nd Eds.). BCS, The Chartered Institute for IT.

Ashour, W. & Fyfe, C. (2007) Inverse Weighted Clustering Algorithm. The University of Paisley.

Attaway, S. (2013). Matlab: A Practical Introduction to Programming and Problem Solving (3rd Eds.). Butterworth-Heinemann.

Basu, S. K. (2013). Design Methods and Analysis of Algorithms (2nd ed.). PHI Learning.

Bell, J. (2014). Machine Learning: Hands-On for Developers and Technical Professionals. John Wiley & Sons.

Brownlee, J. (2013) A Tour of Machine Learning Algorithms. Retrieved from: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.

Buczak, A. L. & Guven, E. (2015). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. In IEEE Communications Surveys & Tutorials, vol. 18, issue 2, pp. 1153-1176. Second quarter 2016. DOI: 10.1109/COMST.2015.2494502.

Butun, I., Kantarci, B., & Erol-Kantarci, M. (2015). Anomaly detection and privacy preservation in cloud-centric Internet of Things. In IEEE International conference on Communication Workshop (ICCW), London, 8-12 June 2015, pp. 2610–2615. DOI: 10.1109/ICCW.2015.7247572.

Candela, J. Q. (2016). The Most Fascinating Work Facebook Is Doing in Machine Learning. Retrieved from http://www.huffingtonpost.com/quora/the-most-fascinating-work_b_9470660.html.

Gartner (2013). Forecast: The Internet of Things, Worldwide, 2013. Retrieved from: http://www.gartner.com/document/2625419?ref=QuickSearch&sthkw=G00259115.

Golman, V. (2014). An Efficient Hybrid Intrusion Detection System based on C5.0 and SVM. In International Journal of Database Theory and Application, vol. 7, No. 2 (2014), pp. 59-70.

Gordon, A. (2015). Official (ISC)2 Guide to the CISSP CBK (4th Eds.). Auerbach Publications.

Familiar, B. (2015). Microservices, IoT and Azure: Leveraging DevOps and Microservice Architecture to deliver SaaS Solutions. Apress.

FET. (2012) Guardian Angels. Retrieved from: http://www.ga-project.eu/files/content/sites/guardians-angels-neutre/files/pdf/Guardian_Angels_Final_Report_July_2012.pdf.

Flach, P. (2012). Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press.

Fortino, G. & Trunfio, P. (Eds.). (2016). Internet of Things Based on Smart Objects: Technology, Middleware and Applications. Springer.

Fu, R., Zheng, K., Zhang, D., & Yang, Y. (2011). An Intrusion Detection Scheme Based on Anomaly Mining in Internet of Things. In IEEE International Conference on Wireless, Mobile & Multimedia Networks (ICWMMN 2011), Beijing, 27 – 30 Nov. 2011, pp. 315-320. DOI: 10.1049/cp.2011.1014.

Hajare, S. A. (2016) Detection of Network Attacks Using Big Data Analysis. In International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, issue 5, pp. 86-88.

Haq, N., Onik, A., Hridoy, A., Rafni, M., Shah, F., & Farid, D. (2015). Application of Machine Learning Approaches in Intrusion Detection System: A Survey. In International Journal of Advanced Research in Artificial Intelligence, vol. 4, issue 3, pp. 9-18.

Hodo, E. et al. (2016) Threat analysis of IoT networks using artificial neural network intrusion detection system. In International Symposium on Networks, Computers and Communications (ISNCC), pp. 1-6.

Holler, J., Tsiatsis, V., Mulligan, C., Karnouskos, S., Avensand, S. & Boyle, D. (2014). From Machine-to-Machine to the Internet of Things Introduction to a New Age of Intelligence. Elsevier.

IERC. (2014) Internet of Things. Retrieved from: http://www.internet-of-things-research.eu/about_iot.htm.

Jung, E., Cho, I., & Kang, S. M. (2014). An Agent Modeling for Overcoming the Heterogeneity in the IoT with Design Patterns. In Park, J., Adeli, H., Park, N. and Woungang, I. (Eds.) Mobile, Ubiquitous, and Intelligent Computing. Vo. 274, pp. 69-74.

Jung, H. (2015). A thing ID-based IoT internetworking framework. In International Conference on Information and Communication Technology Convergence (ICTC). DOI: 10.1109/ICTC.2015.7354567.

ISACA. (2015). Innovation Insights. Retrieved from: http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/innovation-insights.aspx.

Kurniawan, H., Rosmansyah, Y. & Dabarsyah, B. (2015). Android anomaly detection system using machine learning classification. In the International Conference on Electrical Engineering and Informatics (ICEEI).DO:10.1109/ICEEI.2015.7352512.

Madden, S. (2004) Intel Lab Data. Retrieved from: http://db.csail.mit.edu/labdata/labdata.html.

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill Education.

Morgan, L. (2015). 11 Cool Ways to Use Machine Learning. Retrieved from: http://www.informationweek.com/strategic-cio/executive-insights-and-innovation/11-cool-ways-to-use-machine-learning/d/d-id/1323375.

Nerney, C. (2012) The tiny (yet powerful) world of speckled computing. Retrieved from: http://www.itworld.com/article/2721483/consumer-tech-science/the-tiny--yet-powerful--world-of-speckled-computing.html.

Ning, H. (2013). Unit and Ubiquitous Internet of Things. CRC Press Inc.

Pajouh, H. H., Javidan, R., Khayami, R. and Ali, D. (2016) A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. In IEEE Transactions on Emerging Topics in Computing, vol. PP, Issue 99, pp. 1-11. DOI: 10.1109/TETC.2016.2633228.

Prabha, K. & Sree, S. S. (2016). A Survey on IPS Methods and Techniques. In International Journal of Computer Science Issues, Vol. 13, Issue 2, pp. 38-43. March 2016. DOI:10.20943/01201602.3843.

Shalev-Shwartz, S. & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

Sherasiya, T. and Upadhyay, H. (2016) Intrusion Detection System for Internet of Things. In IJARIIE-ISSN(O), vol. 2, issue. 3 (2016), pp. 2395-4396.

Tanpure S. S. et al. (2016) Intrusion Detection System in Data Mining using Hybrid Approach. In International Journal of Computer Applications, pp. 0975-8887.

Van Der Meulen, R. (2015). Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent from 2015. Retrieved from: http://www.gartner.com/newsroom/id/3165317.

Vermesan, O., & Friess, P. (Eds.). (2014). Internet of Things Applications - From Research and Innovation to Market Deployment. River Publishers.

Quinlan, J. R. (2014). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

Zhao, K. & Ge, L. (2016). A Survey on the Internet of Things Security. In 9th International Conference on Computational Intelligence and Security (CIS). DOI: 10.1109/CIS.2013.145.

Appendix

IoT_Clustering.m

```
filename = 'inputData.csv';
reslutfile = 'clusteredData.csv';
M = csvread(filename);

%nunmber of clusters to be created.
K = 2;
%call IWK clustering algorithm. C is a matrix of two values - centroids.
C = IWK(M,K);
%retrieve number of rows and colum of inputData.
[S,dim] = size(M);
%create 2 matrixes of ones values.
R=ones(S,dim+1);
N=ones(K,1);
%recreate inputData to add the cluster ID next to each record.
R(1:S,1:dim)=M(1:S,1:dim);

for i = 1:S
    for j = 1:K
%calcuate the distace between each recoord and centroieds in C. Minimum distance indicates the closnes
of each record to a centroid.
N(j)=norm(M(i,:)-C(j,:))^2;
    end
 [Mn,I] = min(N);
 R(i,dim+1)=I;

end
%copy the clustered data
csvwrite(reslutfile,R)
%show data before clusting and after clustering.
figure(1)
plotmatrix(R(1:S,1), R(1:S,2), '*g')
title('DATA befor clustred')
grid on
axis equal

figure(2)
for i = 1:S
if R(i,dim+1) == 1
    plot(R(i,1), R(i,2), '*r')
    hold on
else
   plot(R(i,1), R(i,2), '*g')
    hold on
end
end
title('DATA after clustred')
grid on
axis equal
clear;
```

IWC.m

```
% This code takes as input parameters, the real dataset that we need to
% cluster and the number of clusters that we need. It returns the prototypes
% values
function [m] = IWK(x,K)

   MM=1;
   NN=MM+2;
   U=1;
   UU=0;

   [dataLen,dim]=size(x);
   dataLen;
   dim;

   m = rand(K,dim)*1;
   d=zeros(dataLen,K);
   flag=zeros(dataLen,1);
   targets = zeros(K,1);
   %this is to run the K-means 10 times.
   for count = 1:10
      for k=1:K
         for n=1:dataLen
            d(n,k)=norm(x(n,:)-m(k,:));
         end
      end

      [mins,I] =min(d');
      I(1,1)
      d;
      for n=1:dataLen
         flag(n) = 1;
      end
                     %trying to find best represntative clusters.
      flag;
      for k=1:K
         num=0;
         den=0;
         for n=1:dataLen
            val1=0;
            val2=0;
            if ((d(n,k) - mins(1,n) < 0.00001) && flag(n,1) ==1)
            %    if(0)
               flag(n,1)=0;
               if mins(1,n) ~= 0
                  for L=1:K
                     val1 = val1 + d(n,L)^(-MM);
                     val2 = val2 + d(n,L)^(-MM);
                  end

              val1=val1-mins(1,n)^(-MM);
              val2=val2-mins(1,n)^(-MM);
              val1 = val1 * NN * mins(1,n)^(NN-2);
              val2 = val2 * NN * mins(1,n)^(NN-2);
              val1= -1*(NN-MM) * mins(1,n)^(NN-MM-2) - val1;
              val2= -1*(NN-MM) * mins(1,n)^(NN-MM-2) - val2;
            else

                val1 = mins(1,n)^(NN-MM-2)*(MM-NN);
            val2 = mins(1,n)^(NN-MM-2)*(MM-NN);


            end


            val1 = val1 * x(n,:);
            num = num + val1;
            den = den + val2;
          else

                val1= val1 + x(n,:) * ( U* MM * mins(1,n)^(NN-0.01)/d(n,k)^(MM+2));
                val2= val2 +          ( U* MM * mins(1,n)^(NN-0.01)/d(n,k)^(MM+2));

            num = num + val1;
            den = den + val2;
          end
        end

        m(k,:) = num/den;
        targets(k,1)=I(1,1)
      end


    end
     targets
     m
```

IoT_Classification.m

```
clear;
Data = 'DataC4_5.csv';
CData = csvread(Data);
%Ouput evaluation variables.
TP=0;
TN=0;
FP=0;
FN=0;

%take out data for training.
train_data = CData(1:4990,1:2)'
train_targets = CData(1:4990,3)'
%take out data for testing.
test_data = CData(4991:7526,1:2)'
test_targets = CData(4991:7526,3)'
%call C4.5 classification method to create the tree.
[tree, discrete_dim] = C4_5(train_data, train_targets, 1)

disp('Classify test samples using the tree')
%use the tree built using training data to examin the test data. Input: test data, number of records in test data, tree, discrete_dim indicates number of classes i.e. 2, 1 or 2.
result    = use_tree(test_data, 1:size(test_data,2), tree, discrete_dim, unique(train_targets));
%transpose the matrix
result'
%restore the result to its orginial state.
result1 = result';
test_data=test_data'
test_targets1=test_targets'
[S,dim] = size(test_targets1);
desResult = zeros(S,1);
figure(1)
for i = 1:S
    if test_targets1(i,1) == 1 && result1(i,1) == 1
        TP = TP+1;
         plot(test_data(i,1), test_data(i,2), '*g')
         hold on
    elseif test_targets1(i,1) == 2 && result1(i,1) == 2
          TN = TN+1;
         plot(test_data(i,1), test_data(i,2), 'om')
         hold on
    elseif test_targets1(i,1) == 1 && result1(i,1) == 2
          FN = FN+1;
            plot(test_data(i,1), test_data(i,2), 'xr')
         hold on
    elseif test_targets1(i,1) == 2 && result1(i,1) == 1
          FP = FP+1;
          plot(test_data(i,1), test_data(i,2), '+y')
         hold on
    end

end
```

```
title('Test data after classification')
grid on
axis equal

for i = 1:S
   if result1(i,1) == 1
     desResult(i,1) = 'A';
   else
       desResult(i,1) = 'N';
   end

end

classifiedReslut = [test_data test_targets' result' ]
dlmwrite('classifiedReslut.txt', classifiedReslut, 'delimiter', '\t');
dlmwrite('NoramalAbnormalclassifiedReslutReslut.txt', string(desResult), 'delimiter', '\t');



disp('True Negative')
TN
disp('True Positive')
TP
disp('False Negative')
FN
disp('False Positive')
FP
disp('Detection Rate')
DetectionRate = (TP) / (TP+FP)
disp('False Alarm')
FalseAlarm = (FP) / (FP+TN)
disp('Accuracy')
Accuracy = (TP+TN) / (TP+TN+FP+FN)
```

C4_5.m

```
function [tree, discrete_dim]  = C4_5(train_data, train_targets, inc_node)

[Ni, M]                 = size(train_data);
inc_node    = inc_node*M/100;
Nu          = 10;

discrete_dim = zeros(1,Ni);
for i = 1:Ni,
    Ub = unique(train_data(i,:));
    Nb = length(Ub);
    if (Nb <= Nu),

        discrete_dim(i)           = Nb;
        dist            = abs(ones(Nb ,1)*train_data(i,:) - Ub'*ones(1, size(train_data,2)));
        [m, in]         = min(dist);
        train_data(i,:)  = Ub(in);
    end
end

disp('Building tree')
tree            = make_tree(train_data, train_targets, inc_node, discrete_dim, max(discrete_dim), 0);
```

Make_tree.m

```
function tree = make_tree(patterns, targets, inc_node, discrete_dim, maxNbin, base)

[Ni, L]                                     = size(patterns);
Uc                                  = unique(targets);
tree.dim                            = 0;

tree.split_loc                      = inf;
if isempty(patterns),
   return
end

if ((inc_node > L) | (L == 1) | (length(Uc) == 1)),
   H                                = hist(targets, length(Uc));
   [m, largest]      = max(H);
   tree.Nf        = [];
   tree.split_loc  = [];
   tree.child                  = Uc(largest);
   return
end

for i = 1:length(Uc),
   Pnode(i) = length(find(targets == Uc(i))) / L;
end
Inode = -sum(Pnode.*log(Pnode)/log(2));

delta_Ib    = zeros(1, Ni);
split_loc = ones(1, Ni)*inf;
for i = 1:Ni,
   data    = patterns(i,:);
   Ud      = unique(data);
   Nbins = length(Ud);
   if (discrete_dim(i)),

      P    = zeros(length(Uc), Nbins);
      for j = 1:length(Uc),
         for k = 1:Nbins,
            indices  = find((targets == Uc(j)) & (patterns(i,:) == Ud(k)));
            P(j,k)    = length(indices);
         end
      end
      Pk         = sum(P);
      P          = P/L;
      Pk         = Pk/sum(Pk);
      info       = sum(-P.*log(eps+P)/log(2));
      delta_Ib(i) = (Inode-sum(Pk.*info))/-sum(Pk.*log(eps+Pk)/log(2));
   else

      P    = zeros(length(Uc), 2);

       [sorted_data, indices] = sort(data);
       sorted_targets = targets(indices);

       I   = zeros(1, L-1);
       for j = 1:L-1,
            P(:, 1) = hist(sorted_targets(1:j) , Uc);
            P(:, 2) = hist(sorted_targets(j+1:end) , Uc);
            Ps                  = sum(P)/L;
            P                   = P/L;
            Pk      = sum(P);
            P1      = repmat(Pk, length(Uc), 1);
            P1      = P1 + eps*(P1==0);
            info        = sum(-P.*log(eps+P./P1)/log(2));
            I(j)        = Inode - sum(info.*Ps);
         end
         [delta_Ib(i), s] = max(I);
         split_loc(i) = sorted_data(s);
      end
   end

[m, dim]    = max(delta_Ib);
dims        = 1:Ni;
tree.dim    = dim;

Nf                  = unique(patterns(dim,:));
Nbins     = length(Nf);
tree.Nf = Nf;
tree.split_loc      = split_loc(dim);

if (Nbins == 1)
   H                                    = hist(targets, length(Uc));
   [m, largest]      = max(H);
   tree.Nf        = [];
   tree.split_loc  = [];
   tree.child                    = Uc(largest);
   return
end
if (discrete_dim(dim)),

   for i = 1:Nbins,
      indices        = find(patterns(dim, :) == Nf(i));
      tree.child(i)  = make_tree(patterns(dims, indices), targets(indices), inc_node, discrete_dim(dims), 
maxNbin, base);
   end
else

   indices1                             = find(patterns(dim,:) <= split_loc(dim));
   indices2                             = find(patterns(dim,:) > split_loc(dim));
   if ~(isempty(indices1) | isempty(indices2))
      tree.child(1) = make_tree(patterns(dims, indices1), targets(indices1), inc_node, discrete_dim(dims), 
maxNbin, base+1);
      tree.child(2) = make_tree(patterns(dims, indices2), targets(indices2), inc_node, discrete_dim(dims), 
maxNbin, base+1);
   else
      H                                 = hist(targets, length(Uc));
      [m, largest]  = max(H);
      tree.child                  = Uc(largest);
      tree.dim              = 0;
   end
end
```

Use_tree.m

```
function targets = use_tree(patterns, indices, tree, discrete_dim, Uc)

targets = zeros(1, size(patterns,2));
if (tree.dim == 0)

   targets(indices) = tree.child;
   return
end
dim = tree.dim;
dims= 1:size(patterns,1);
if (discrete_dim(dim) == 0),

   in                  = indices(find(patterns(dim, indices) <= tree.split_loc));
   targets             = targets + use_tree(patterns(dims, :), in, tree.child(1), discrete_dim(dims),
Uc);
   in                  = indices(find(patterns(dim, indices) >  tree.split_loc));
   targets             = targets + use_tree(patterns(dims, :), in, tree.child(2), discrete_dim(dims),
Uc);
else

   Uf                  = unique(patterns(dim,:));
   for i = 1:length(Uf),
      if any(Uf(i) == tree.Nf)
         in       = indices(find(patterns(dim, indices) == Uf(i)));
         targets  = targets + use_tree(patterns(dims, :), in, tree.child(find(Uf(i)==tree.Nf)),
discrete_dim(dims), Uc);
      end
   end
end
```