Botnet Detection In IoT Devices using Machine Learning

By

Shakir Rouf
16101104
Nazmus Sakib Akash
16101208
Amlan Chowdhury
16101042
Sigma Jahan
16301031

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
BRAC University
December 2019

Declaration

It is hereby declared that:

1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Shakir Rouf 16101104	Nazmus Sakib Akash 16101208
Amlan Chowdhury 16101042	Sigma Jahan 16301031

Approval

The thesis/project titled “Botnet Detection In IoT Devices using Machine Learning” submitted by

1. Shakir Rouf (16101104)
2. Nazmus Sakib Akash (16101208)
3. Amlan Chowdhury (16101042)
4. Sigma Jahan (16301031)

Of Fall, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on December 24, 2019.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Mahbub Alam Majumdar
Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Internet of Things (IoT) devices are a group of interconnected devices or machines that have the ability to transfer data over a network without the influence of any external factor. The technology makes use of sensor nodes embedded into everyday computing objects, which communicate in a wireless multi-hop fashion to exchange data over a local network or the internet. With the rapid technological advancements taking place around the globe, the use of IoT devices has also increased proportionately. Although the prevalence of IoT devices in human lives has influenced the IoT manufacturers to make it cheap an accessible, but on the other hand, the system provides minimal control with no substantial security measures due to its prodigious application, which in turn makes it susceptible to botnet attacks. Botnet is a network of interconnected malware contaminated IoT devices, individually referred to as a bot. These bots are used as instruments of malicious attack on a network of IoT devices which allows the group of hackers (referred to as Botmaster) to perform distributed denial-of-service attack (DDoS), data theft and spam by flooding the network with unnecessary information. As a result, botnet detection has risen as an essential ingredient of network security. In this paper, our motive is to use various Machine Learning algorithms to detect botnet attacks and filter out the algorithm which will be most suitable and accurate to detect such attacks by comparing the derived outputs.

Keywords: *Botmaster, Botnet, DDoS (Distributed Denial of Service), IoT (Internet of Things), Multi-hop, Machine Learning Classifiers, Sensor Nodes.*

Acknowledgement

Firstly, all praise to the Almighty for whom we have been able to complete our thesis without any major interruption. Secondly, to our advisor Dr. Amitabha Chakrabarty sir for his kind support and advice in our work. He has pushed us to test our limits and bring out results in a sector we struggled to understand. And finally, to our parents who have always motivated us and inspired us to work. Without their thorough support, our thesis would not have been complete. With their prayers, support, inspiration and investment, we are about to complete out graduation from this prestigious institution of Brac University.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Internet of Things (IoT) refers to a global network of sensor nodes. It is a mesh of devices connected over a local network or the Internet so that they may exchange data. This data is generated by the sensor nodes that are embedded into the devices. The sensor nodes are constantly generating large volumes of raw data which have little or no interpret-ability. Our research reflects on taking this data and deriving meaningful insights from it. As mentioned before, IoT in general refers to a very large network, which is practically impossible to visualize from a standalone point of view. This make the system harder to monitor, while leaving loop holes in the security protocols, that can be easily bypassed and would let any given person an access to a huge amount of confidential data. Sensitive data can then be misused to perform various illegal activities. The system provides little control with no evident security measures. Our research is based upon the Mirai botnet attack that shook the whole world when it was released on the Internet as an open-source, leaving people access to an algorithm that would prove to be very dangerous for large organizations and would leave the privacy of people exposed to these hackers, known as Botmasters. In our research we have targeted to solve these problems by collecting raw data that has been divided into two parts. The benign data represent the state of a device where it is not under attack, while the attack data represents vice versa. We have looked in to various classification algorithms which were backed by multiple feature selection processes, known as dimension reduction techniques, to determine the state of a device. The dimension reduction techniques have helped us to determine which features would prove to be prolific in determining the outcome. Our research dives deep into four different feature selection processes and each of the processes contain four classifiers. Although we had access to 9 different sets of data from 9 different IoT devices, we did our research based on a security camera. The insights from the research can be further used to determine how good the algorithms that we have designed would work on the other devices.

1.1 Motivation

The Internet of Things (IoT) enhances our methods of collecting data from the real world like smart devices, factories, automobiles, agriculture, medical services and even ourselves. The resulting surge of records, from numerous IoT devices all over the world will eventually create many opportunities for improving quite a number of elements of real-world problems. However, realizing this potential also requires us to address the fact that the raw data needs to be mined properly using relative mathematical models, to give it some meaning. The knowledge generated from the system can be then used for multiple purposes. It is important to note that raw data itself will provide no substantial benefit, hence, processing it is mandatory. As the reputation of the Internet increases, so does the wide variety of miscreants who abuse the Internet for their nefarious purposes. Mirai (known as 'future' in Japanese literature) is a malware that places itself onto devices connected through a network, via downloading. The devices may include IoT units like smartphones, laptops, desktops or other devices. The malware eventually turns these devices into bots that can be remotely controlled using a Control & Command server, and makes them a part of the larger Bot Network (botnet) which can be used for performing large-scale DDoS (Distributed Denial of Service) assaults of multiple types. This malware mainly focuses on devices like domestic routers and IP cameras that are constantly connected to the Internet through some path using a Network [1]. The discovery of the Mirai botnet dates back to August, 2016 [2] and has been developed by group of white hat malware researchers known as Malware-MustDie [3]. They are known throughout the world for being part of some of the most troublesome DDoS (Distributed Denial of Service) assaults. They include the attack on the website of computer security journalist 'Brian Krebs' on the date 20^{th} September, 2016 [4], and followed up with the assault on French Internet host OVH [5]. There was also the DYN cyber-attack dating back to October, 2016 [6][7]. There was a text log between 'Anna-senpai' and 'Robert Coelho', which serves as a proof of the fact that the Mirai botnet was named after one of the Television animated series in 2011 named 'Mirai Nikki' [8]. The worst of the circumstances was yet to follow, when the Mirai source code was uploaded onto Hack Forums as an open-source [9]. From there onwards, with code being published online, multiple miscreants have made use of this with slight upgrades and alterations, and adapted them into their own personal malware-based attacks which eventually caused a great deal of harm [10][11]. Then came the DDoS assault that left a whole lot of the Internet inaccessible in U.S (United States). It was actually the work of the Mirai botnet, although officials initially thought of the attack having been being executed by an opposing nation. This Mirai botnet based on which we are performing our research, never had such grand nefarious ambitions. All they wanted to do through developing such malware in the first place was, to make little money off of Minecraft enthusiasts, but it eventually evolved and became more powerful beyond their imagination. It is a story of unintended penalties and sudden protection threats, and it says a lot about our current world. The immense rise in the number of IoT based DDoS attacks, witnessed in current years, will likely proceed until IoT device manufacturers address this security issue and take steps to prevent this. Our motivation is to serve as part of the bigger group, who intend to prevent such mishaps from taking place and work to design and deploy algorithms that will prove to be helpful in detecting and preventing it.

1.2 Thesis Contribution

1.2.1 Problem statement

In the world today, IoT device manufacturers have targeted consumer level IoT devices; the consumers care more for the low value and ease of use, while disregarding the need for security and control. The exponential rise in demand for these IoT devices has led the IoT device manufacturers to avoid necessary protection features and keep injecting large volumes of insecure gadgets such as Digital Video Recorder boxes, IP cameras and many more. And therefore, it leads to these weaknesses being epitomized through the insecure security protocols being used, default credentials and the inherent computational limitations. Hence, we are left with an exponentially expanding band of external attack sources due to the immense production of such devices and the ease with which hackers can locate them and use them for various malicious means, such as Shodan [12]. One such famous system that has incorporated itself into the ecosystem of hackers today, is the use of Botnets. The bot is known as a malware affected device that intends of compromising and taking control of these IoT devices that are constantly connected to Internet through various means. The process takes place in the following way. Firstly, the malware is placed onto the user's device which can be a smartphone, PC or laptop; it is done via downloading the malware forcefully or tricking the user into downloading it through the use of social engineering techniques like clicking on an advertisement and many more like this [13]. The victim is totally uncertain of the fact that his device has been compromised and is now part of the Bot Network. From here on out, multiple malware-infected devices creating a network of bots, is put under the command of a single entity. This entity is known as the hacker or otherwise the Bot Master. The network of bots is therefore known as the Botnet. The hacker controls the network of bots through a Control and Command server and perform these DDoS (Distributed Denial of Service) assaults. To complement the evaluation of these devices, it is necessary to have a network based-detection device accessible which can monitor the whole network and look for the indications of bot-infected gadgets. As long as these issues remain unaccounted for, the technology known as IoT can become the new playground for these hackers to perform more dangerous DDoS assaults that could mean a great deal of harm to us, and hence presents us with a wide variety of challenging task to accomplish so that we can be able to counteract it. Due to the fact that more of these DDoS attacks are trying to target consumer-level IoT devices and the problems presented earlier, coupled with a lack of technical attention towards the inherent computational vulnerabilities of the IoT technology, is what led us to deciding our topic of research.

1.2.2 Solutions

There have been some significant achievements in the field of Intrusion Detection, with the execution of deep learning approaches like: Bidirectional Long-Short Term Memory and Recurrent Neural Networking. A packet-level Machine Learning DoS detection has also been worked on, with it being able to recognize typical DoS attack traffic from consumer IoT devices.
The results we observed from our research showed us that machine learning is ap-

plicable to our problem and our problem is a learnable one. The final model was able to predict the state of an individual IoT device with an average accuracy of 98% and within the time frame of 3 to 10 seconds. With the help of four machine learning algorithms which include kNN, Naïve Bayes, Random Forest, Support Vector Machine and the dimension reduction processes such as Principal Component Analysis, Independent Component Analysis, Random Forest Regression, High Correlation Filter; we have managed to solve the problem statement that has been presented in (section 1.2.1). We have been able to accurately predict the state of an IoT device; whether it is in a benign state, or is it being attacked. All four of our Machine Learning classifiers have performed very good. The reason behind this is that, our dataset was very representative. Our sampling techniques have captured substantial amount of information in the original dataset which eventually has helped us to develop predictors that are able to predict the outcomes with high precision while being efficient at the same time with very fast prediction times. The result analysis in Chapter 6 will provide a more in-depth review of our work.
It was also evident that Random Forest was able to improve the performance of the model greatly, although at a considerable cost in terms of overfitting. This is a good reminder that proper feature engineering and collecting as much data as possible has a much larger pay-off than refurbishing the algorithm. We also observed the trade-off in run-time versus accuracy, while working with kNN classifier, it produced some of the best results but underperformed in terms of time taken for prediction in comparison to the other learning models. This is one of many considerations we have to take into account when designing machine learning models. We trained our models to be accurate to a great extent, but we do not know how it generates the predictions. So, our next step in the machine learning process is crucial, to understand how the model makes predictions. Achieving high accuracy is great, but it would also be helpful if we could figure out why the model is able to predict with such a high accuracy to better understand each model. One of the possibilities could be overfittng by seeing too many samples and relying too much on a large sample data rather than designing a model around it to generate predictions. Even so we are still left with quite a few questions. For example, what features does our model rely on when predicting the state of an IoT device? Is it possible to design a more robust model, based on the feature selection process? Would the models survive in real-world situations and not generate predictions based on overfitting? The upcoming chapters will delve into it and answer the questions, following up with a more meaningful insight on our work.

1.2.3 Methodology

Our research uses raw data collected from the UCI repository number 00442 of machine learning. We ran the data through a few pre-processing phases before diving into the feature selection and classification process. We introduced a new column 'Attack' which denotes the state of the IoT device (Security Camera) in either benign phase or attack phase. We had our hand on two different datasets that were concatenated to form a single dataframe. This dataframe was then run through a handful of dimension reduction techniques. This is because our raw dataset had 115 features, hence it would have been hard to design a model around it, and even more hard to visualize the data and how it behaves. Then we applied the previously mentioned classification techniques to complete our research .We will dive deep into each of those algorithms, and do an in-depth analysis of them in the following chapters.

Chapter 2

IoT & Security

2.1 What is IoT?

The Internet is a global wide-area network which connects all the computer system across the world, with the inclusion of devices that are part of the IoT (Internet of things) technology. It is a method for interfacing a computer to another one, using switches and servers. The web is a system of worldwide trades including: private or open businesses, scholastic corporations and government systems – associated with directed, remote and fiber-optic advances [14]. Internet Protocol (IP) is considered as the internet's fundamental component and communication backbone. The Internet of Things (IoT) portrays the system of physical items, "things", which are installed with sensors, programs and different advances to connect and trade information along with different gadgets and frameworks over the web. The Internet of Things has transformed basic machines and home frameworks into "smart" gadgets that can be observed and controlled over the Internet. IoT gadgets are intended to work for individuals at home or in industry. All things considered, the IoT gadgets can be ordered into three primary classifications: consumer, enterprise and industrial. Although, smart devices like watches and phones are part of the consumer IoT sector, the weather and traffic management system is a core component of the Industrial or Enterprise IoT chains.

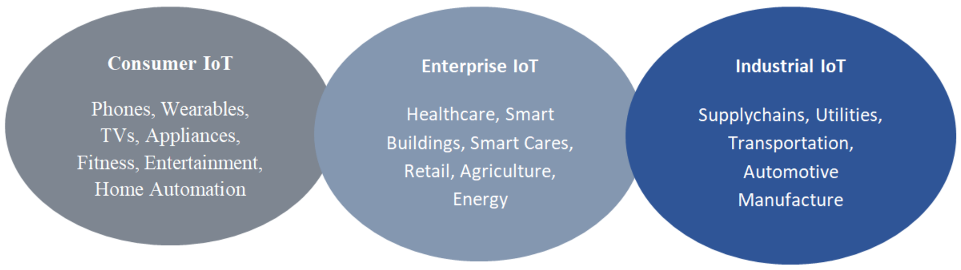

Consumer IoT
Phones, Wearables, TVs, Appliances, Fitness, Entertainment, Home Automation
Enterprise IoT
Healthcare, Smart Buildings, Smart Cares, Retail, Agriculture, Energy
Industrial IoT
Supplychains, Utilities, Transportation, Automotive Manufacture

Figure 2.1: Classification of IoT Devices

2.1.1 Importance of IoT

In the course of recent years, IoT has became one of the most significant advancements of the 21st century. Since we can couple everyday items like kitchen machines, vehicles, indoor regulators, child screens, to the web through the help of the sensor nodes embedded into them. By the use of methods for ease of registering to the cloud, large information, investigation, and portable advances, physical things can share and gather information with insignificant human mediation. In this hyper-connected world, computerized frameworks can record, screen, and change every interaction between connected devices. Hence, the physical world meets the advanced world of technology, to coordinate. IoT is changing each aspect of our lives and it isn't only simply changing our lives, it is making our lives, way simpler and more straightforward. IoT gives organizations and individuals better understanding and power. Moreover, IoT enable us to be progressively associated with the general surroundings and to accomplish significantly elevated level of work. Within excess of 7 billion associated IoT gadgets today, specialists are anticipating that this number should develop to 10 billion by 2020 and 22 billion by 2025 [15]. Although the IoT has been in presence for quite a while, an assortment of current advances in various innovations is constantly helping it to evolve.

1. Access to ease: low-control sensor innovation. Moderate and solid sensors are making IoT innovation workable for common people.
2. Availability: A large group of system conventions for the internet has made it simple to interface sensors to the cloud and gadgets for moving productive information.
3. Distributed computing stages: The expansion in the accessibility of cloud stages empowers the two organizations and customers to get to the foundation where they have to scale up without really overseeing everything.
4. AI and investigation: With signs of progress in Artificial Intelligence and investigation, alongside access to different measures of information put away in the cloud, organizations can accumulate knowledge quickly and effectively. The rise of these united advances keeps on driving the limits of IoT and the information created by IoT additionally sustains these advances.
5. Conversational computerized reasoning: Advances in neural systems have brought Natural Language Processing (NLP) to IoT gadgets, (for example, Alexa, Cortana, and Siri) and made them engaging, moderate, and reasonable for home use.

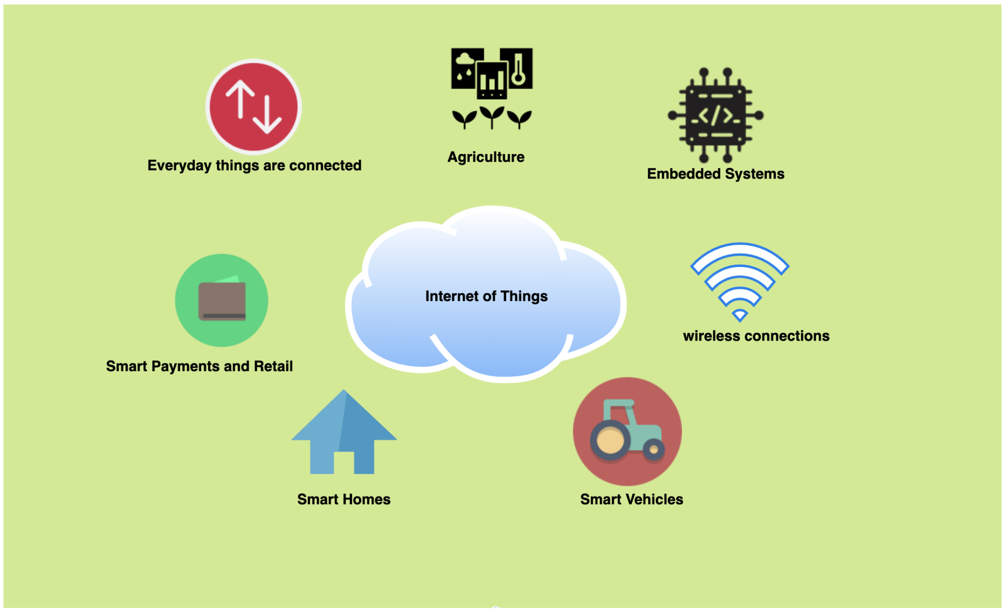


Figure 2.2: Various usages of IoT

2.1.2 How does IoT work?

The whole IoT system mainly incorporates 4 integral parts: Collecting the data generated by the sensor nodes, a User interface, Processing the data and Connectivity.

1. Collection of data from the sensor nodes: This exhibit the use of sensors to collect data. Sensors can range from a wide variety of moisture, air quality, temperature, motion, light sensors and many more. This collection of sensors aids us in collecting data from the real world. Although the data is unstructured and raw, yet, insight can be drawn from them eventually leading to more smarter choices.

2. User Interface: From here on out, the information collected by the sensors and processed is made significant to the end-user to a greater extent. It can be using an alarm to notify the client of a message, email, phone-call, warning and so on. For example, using a text or alarm-based indicator to an exorbitantly high temperature so that the user may take necessary actions regarding the issue. This way the user can have access to an interface that empowers them to proactively screen the system and act accordingly.

3. Processing the Data: When the surge of information floods the cloud, some pre-programming helps to prepare it. This can be something very simple, such as monitoring the temperature and observing if it is becoming very hot. Moreover, it can be an unpredictable event taking place. In any case, if the temperature rises above a certain threshold or someone violates your privacy at home through breaking and entering, the data collected is processed and presented to the user so that he/she may take action.

4. Connectivity: It is the core aspect of the whole technology; this is what mediates the flow of information from one system to another. The information generated by the sensor nodes are then forwarded to the cloud. There are

many ways these gadgets can be connected to the cloud, through using cellular, WiFi, Bluetooth, satellite or being interfaced directly to the Internet through the use of an Ethernet cable. Each and every alternative has a certain trade-off that takes place between the capacity to transfer data, security, range of coverage etc. The best option that should be selected comes down to keeping the specific application purpose in mind. Although they all serve the same purpose, which is getting the information generated, to the cloud.

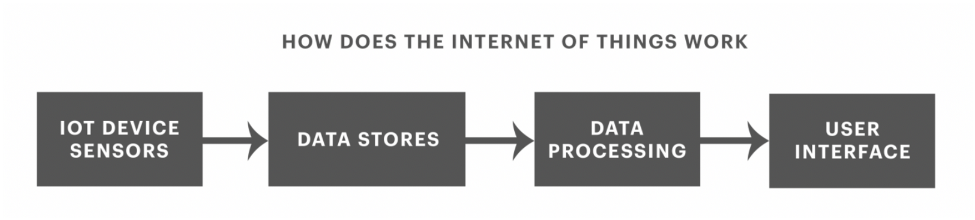


Figure 2.3: How IoT works

2.2 The Challenges of IoT

Multiple key IoT issue territories are analyzed to investigate the most crushing difficulties and questions .The main challenges of IoT include security, legal, regulatory, and rights privacy.

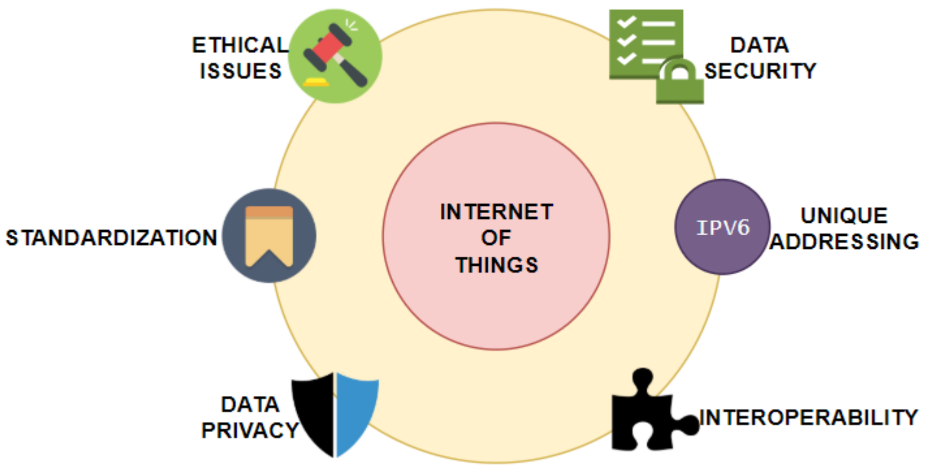


Figure 2.4: Obstacles of IoT

2.2.1 Security

Even though in the world today, monitoring security threats is not a new, yet the extensive use if IoT devices introduce some new and more challenging security tasks. Both the users and manufacturers of the IoT technology need to be aware of the fact that these personal information in their devices are not secure enough due to internal vulnerabilities of the system. The main reason behind this is that, IoT provides a substantial amount of the personal data in extreme details without the user's active participation, and the technology has become an integral part of our daily lives. Due the network being saturated enough by millions of devices, the task of monitoring the system is a challenging one indeed. The idea of IoT devices being constantly connected to the Internet in some way, eventually deteriorates the security and the strength of the Internet as a whole. As far as the standard of the Internet security is concerned, designers and clients are not paying enough attention to this issue while manufacturing or purchasing these IoT devices. Likewise, a synergistic effort is expected to create a good enough influence to solve this issue, and would definitely improve the technology altogether. Specialists have just shown that it is possible to make remote hacks on pacemakers and vehicles, and in October 2016, the Mirai botnet had compromised the servers on the east shore of the United States. It was even made possible to penetrate systems through making or turning devices such as remote switches and cameras into Bots controlled by a Botmaster [16]. On account of the Mirai botnet, it was evident that the use of default passwords and security protocols made it easier for bypassing the protection. Although, much more solid passwords, validation/approval procedures, and cryptography would actually make a dent in the issue regarding security.

2.2.2 Privacy

In the IoT (Internet of things) technology devices are connected with a complex and saturated collection of hardware and software, hence it becomes hard to monitor and manipulate the system as sensitive information or personal data can leak through anytime through some kind of security bypassing by a hacker. All the devices are constantly sharing substantial amount of personal details about the user in some sort of association. The details can range from name to birthplace and many more intricate information [17]. The Internet of Things (IoT) can produce huge quantities of data. This record has to be transmitted, processed in some way, and then doubtlessly stored somewhere, hopefully securely. (Pollmann, 2017) says "Much of this data is private data, and some can be pretty sensitive. This brings statistical privacy questions to the forefront" [18]. Although, there are safety and privacy concerns with IoT, it serves to make our everyday lives easier through its multi-functional capability and ability to gather information from outside and present them in a way that is interpretable. It eventually helps us to perform our everyday duties; remotely and automatically, and more importantly, it is a game-changer for industries.

2.3 Distributed Denial-of-Service (DDoS)

The DDoS (Distributed Denial of Service) assault is at the core of one of the dominant weapons on the Internet known to performing network-based attacks. The attack performs by flooding huge amounts of traffic onto a server or system by means of IoT devices, eventually crashing it and making it obsolete. Network connections that make up the Internet are consisted of, multiple layers of the OSI (Open Systems Interconnection) model. Many of the noteworthy layers are known as: The Network Layer, Transport Layer and Application Layer. There are multiple DDoS attacks that are based around these specific layers. The following categories exhibit their meanings:

1. Volume based attacks: This attack makes use of multiple IoT devices that have been contaminated to transfer large quantities of packets making association demands, eventually bringing down the bandwidth of the network.

2. Protocol attacks: These are more focused attacks that look for vulnerabilities in a server or system's resources eventually utilizing them to perform an assault.

3. Application attacks: They are known as the most famous forms of DDoS assaults, and perform their tasks by focusing on specific Internet applications.

Distributed Denial-of-Service attacks mainly target websites and online services. The task is to overpower the server or system by more amount of traffic than it is capable of handling. Their sole purpose is to make the site or administration obsolete through various means. The traffic is mainly compromised of packets and demands for associations [19]. This might be joined with a coercion danger of an all the more decimating assault except if the organization reimburses a cryptographic recovery system.

2.3.1 Botnet

In some essential ways, DDoS (Distributed Denial of Service) are performed by using a mesh of remotely controlled PCs or other IoT devices; these are compromised devices that have been hacked and now work as bots. The botnet, otherwise known as the Bot Network, is a case of using a devastating technology for nefarious purposes. Botnets are at the core of the numerous hazards that the Internet presents. It consists of a network of gadgets (IoT devices) that may be wirelessly connected to the Internet, which include smart phones, servers, smart devices and other IoT gadgets. These devices are comprised of a typical sort of malware that turns it into a bot and it eventually becomes a part of the larger Bot Network. Users are usually unknown to the fact that their device may be a part of the botnet. These malware infected devices are in turn remotely controlled by hackers, who are known as BotMasters; they control these contaminated gadgets through the use of a Control and Command Server, and utilize the bots to perform illegal activities, and the notorious task remains unidentified to the client. Bots are generally used to send spam mails, participate in sending over large volumes of data (Volume Based Attacks) and create malicious traffic that surpasses the threshold of a large organizational network for DDoS (Distributed Denial of Service) attacks. As mentioned, they are utilized to flood traffic by sending over huge amounts of packets, focused on systems, servers and websites than they are capable of handling.

These typical botnets vulnerates the transmission capacity of the targeted individual by sending over more association requests than a server or network is capable of managing. Hackers use botnets for multiple purposes, which may range from spam mails to downloading malwares onto a particular device. Progressively, the huge number of IoT gadgets that are overflowing the markets and extends the reach of the Internet of things system, are being targeted and transformed into a part of the Bot Network used to perform DDoS assaults. The disregard of the technology for security measures with minimal control, eventually leaves the devices defenseless against hackers who the misuse these devices to create a larger

Bot Network. The infamous DYN assault that took place in 2016 was performed through utilizing the Mirai malware, which made a Bot Network consisting of IP Cameras, Smart Televisions, Printers and Smart Phones. The Mirai Botnet made up of IoT devices might be more hazardous than it was originally intended. After the Mirai source code was released to the world as an open-source, multiple DDoS assaults have taken place, by making slight tweaks to the original code. Cyber-criminals used the code and altered it to develop multiple versions for use in future DDoS attacks [20]. What Mirai does is, it forces the devices to download a malware, eventually taking control over it and making it report to a Control and Command Server, turning them into bots that can be utilized to perform DDoS assaults on large organizations. The following figure 2.5, shows how a general bot attack is performed.

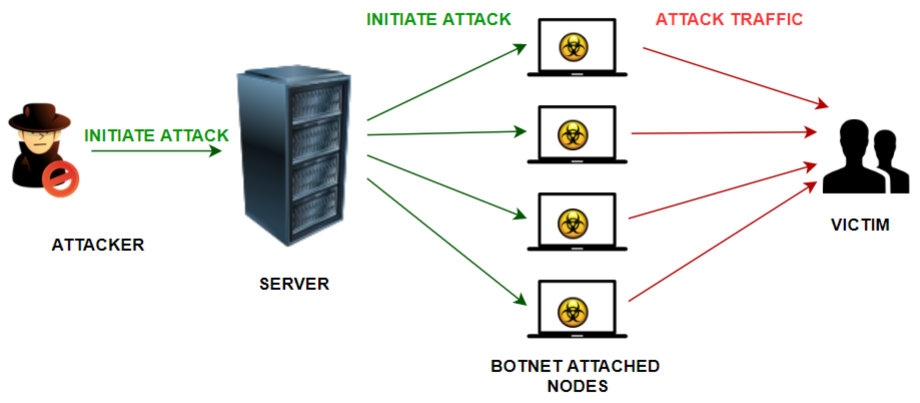


Figure 2.5: The Botnet Attack process

2.4 Machine Learning

Machine learning refers to a sort of information examination, that utilize mathematical models that learn from the information provided to it in terms of samples. It is a sort of Artificial Intelligence(AI) that furnishes frameworks with the capacity to learn without being expressly customized. The information is being utilized to make predictions, not codes. Information is dynamic so the system enables the framework to learn and develop models, and the more information that is analyzed, the higher the accuracy, in terms of prediction. Although, it may come at the cost of overfitting the training data. Machine learning was first characterized in 1959 by Arthur Samuel, a pioneer in the field of man-made consciousness and AI. Samuel characterized machine learning as a field of practice that enables personal computers to learn without being unequivocally modified [21]. The ability of Machine Learning is to develop models that are capable of making their own judgment based on decisions and classifications, rather than requiring any kind of human intervention and manual rules. There are seven steps to the whole process:

1. Collecting the data.

2. Pre-processing the data.

3. Making use of model to represent the data.
4. Training the dataset on that model.
5. Evaluating the model based on classification accuracy.
6. Performing hyper-parameter tuning.
7. Making predictions.

STEP 01
Data collection
STEP 02
Data preparation
STEP 03
Choosing a model
STEP 04
Training
STEP 05
Evaluation
STEP 06
Parameter tuning
STEP 07
Prediction

Figure 2.6: Steps of Machine Learning

2.4.1 Supervised Learning

Supervised learning is the most well-known sub branch of machine learning today. The name ”Supervised” comes from the preparation that this sort of calculation resembles, in having an instructor regulate the entire procedure. When preparing a regulated learning calculation, the preparation information will comprise of data sources combined with the right yields. During preparing, the calculation will look for designs in the information that relates to the ideal yields. In the wake of preparing, a supervised learning calculation will take in new inconspicuous sources of information and will figure out how the outcomes should be predicted based on the information that has been provided to it earlier. The target of a regulated learning model is to foresee the right mark for recently introduced information. Regulated learning can be part of two subcategories: Classification and Regression. Our model is a supervised learning model.

2.4.2 Unsupervised Learning

Unsupervised learning is where no labels are given to the learning calculation, leaving it all alone to discover structure in its information. The problem of unsupervised learning is in trying to find a hidden structure in unlabeled data. The objective in such learning issues might be to find gatherings of comparative models inside the information, where it is called clustering or to decide how the information is conveyed in the space, known as density estimation [22]. To put forward in simpler terms, for an n-sampled space $x(1)$ to $x(n)$, true class labels are not provided for each sample, hence known as learning without supervision.

2.5 Botnet Detection

Botnet detection can be categorized into the Honeynet-based and Intrusion Detection System. In IDS there is a subsection consist of behavior-based detection and hybrid. Behavior-based detection can be parted into Anomaly-based and Signature-based. Later, from anomaly-based, there are four different methods which are DNS based detection, Data-mining, Host-based, Network-based [23]. Active monitoring and passive monitoring, these two are forms a Network-based detection system. Figure 2.7 exhibits a tree that shows the multiple approaches to Botnet Detection. The goal of a detection model is to monitor the network traffic activity and look for indications to the presence of a bot-infected device. The system generally works by overseeing a huge collection of devices, which are generating data through the use of sensor nodes. Each instance of these real-time data are further processed through a predictor which classifies the state of the device. Although, real-time monitoring of these many devices is a challenging task, yet it is not an impossible one. It will require a huge amount of computational power. There are many other approaches to detect botnet like deep learning, kernel recursive, least squares, neural networking and finally machine learning.

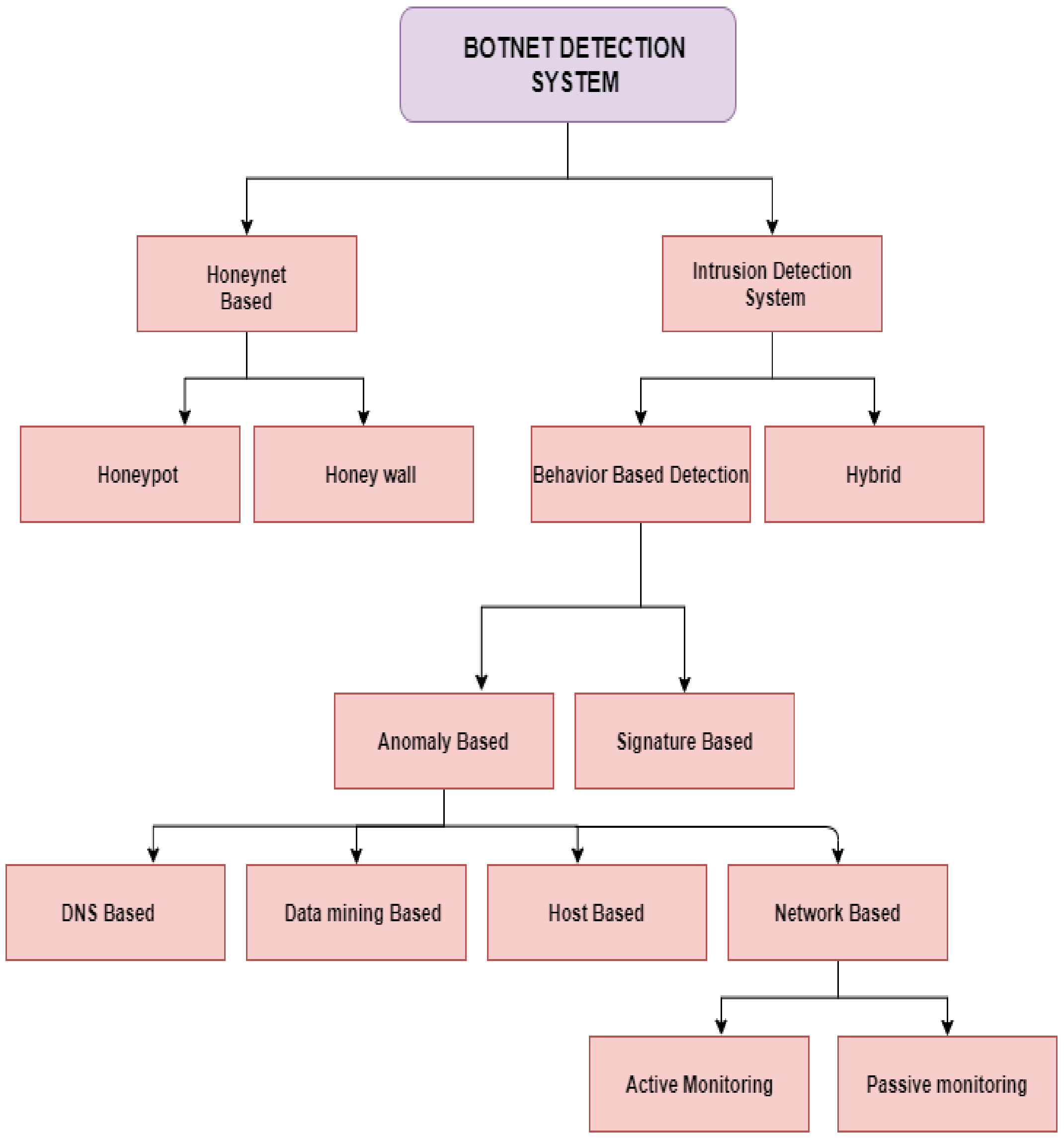


Figure 2.7: Classification of Botnet Detection

Chapter 3

Literature Reviews

IoT (Internet of Things) requires smart handling and dependable transmission inside the system. Other than all the benefits and positive qualities that the remote medium has brought to IoT, there are many drawbacks in the security and privacy issue of the system. Since, it provides a huge amount of personal in extreme detail without the user's active participation. There are different types of traffic oddities and attacks. The most widely recognized security dangers are gatecrashers, which is mainly known to the world as infection, malware or anomaly. This accompanies the danger of individual information being communicated to the world and cause more digital assaults like Denial of Service (DoS), Remote to Local (R2L), Probe, botnet and much more. Ordinary flow of traffic has pattern, due to the fact that these assaults are not regular. In any case, when another element will take an attempt to twist or alter the information, the example will change and this is the place the intrusion recognition procedure becomes possibly the most important factor with regards to the web and specifically with regards to the Internet of Things.

R. Doshi et al [24] proposed that a packet-level machine learning DoS detection can precisely recognize typical DoS attack traffic from consumer IoT devices. They utilized a constrained list of features to limit computational overhead, which is vital for real-time classification and middlebox deployment. They have tested diverse machine learning classifiers on a dataset of normal and DoS attack traffic, collected from an experimental consumer IoT attack network. However, to avoid complications of running the actual Mirai Botnet, they simulated the three most common classes of DoS attacks a Mirai-infected device will run - those being "TCP SYN Flood", "UDP Flood", "HTTP Get Flood" and only amassed 300 seconds of Attack data which makes their dataset relatively small.

Mcdermott et al [25] demonstrated the execution of deep learning in this sector with the Bidirectional Long Short-Term Memory Recurrent Neural Networking conjunction with Word embedding for botnet detection. However, the bidirectional approach adds overhead to each epoch and builds preparing time and the "ACK" attack vector metrics were shown to be less favourable.

Another proposal by Mai L. et al [26] was categorizing flows into similar groups which would decrease the complexity of training data. To screen and recognize abnormalities or any suspicious conduct, Interruption Identification Framework (IDS) and Interruption Anticipation Framework (IPS) are being utilized. As different conditions and most recent advances are inclined to be malignantly assaulted, machine learning calculations can identify, break down and group gatecrashers precisely and rapidly [27]. Yet, the research falls short in feature selection technique as they get each feature one by one until reaching the best performance for clustering similar flows. This is time consuming in regular cases as their accuracy is hindered by a prediction time of 51.07 seconds.

Roughan et al [47] uses the classification of traffic for the purpose of identifying four major classes of services: interactive, bulk data transfer, streaming, and transactional. They study the effectiveness of using packet size and flow duration characteristics, and simple classification schemes were observed to produce very accurate traffic flow classification.

In addition, a paper presented by Ahmed, shows that they have developed a sequential, real-time anomaly detection algorithm using Kernel Recursive Least Square method that incrementally constructs and keeps a dictionary of enter vectors that defines the region of normal behavior [48]. The dictionary adapts over time to tackle modifications in the shape of ordinary traffic, with new elements being added out of date individuals deleted as the normality vicinity expands or migrates.

Although multiple approaches have already been accounted for such as Neural Networking, Deep Learning and a few Machine learning approaches. Simply due to fact that our problem is a classification of compromised IoT devices, is the reason for targeting classifiers like (k-NN, Naïve Bayes, Random Forest, Support Vector Machine).

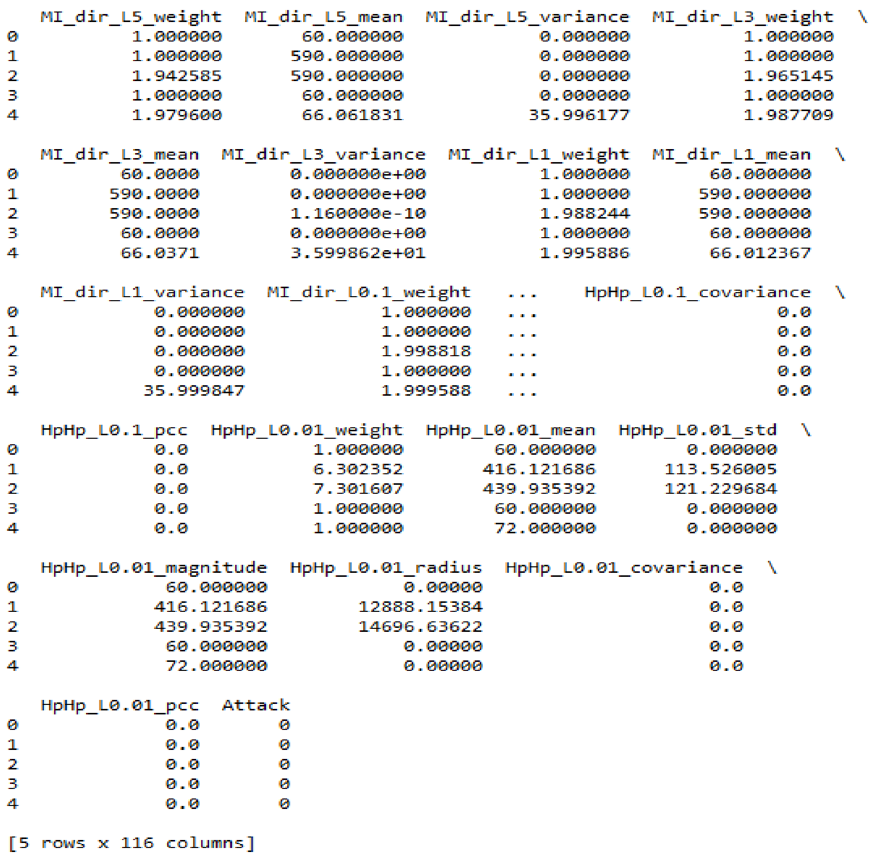
Chapter 4

Collecting and Processing the Dataset

A properly labelled dataset for Botnet detection in IoT (Internet of Things) is very rare to find. Hence for our research we are using a dataset for Mirai Botnet dataset from the UCI repository for Machine Learning. The device we are focusing on is a security camera, labelled as: Provision_PT_737E_Security_Camera. The creator of the dataset is Yair Meidan, and dates back to March, 2018. We have our hands on two different datasets, for the benign dataset labelled as benign.csv where the data represents a phase where the aforementioned IoT device is attack free. And the attack data labelled as "udp.csv", which represents the phase where the device is being attacked. In total there are 115 features for each dataset, which have been further processed using a few dimension reduction techniques. As for the number of entries or records in each dataset, it consisted of 62154 entries for the benign dataset, and 156248 entries for the attack dataset. The UCI Machine learning repository number 00442 consisted of 9 different device datasets, with around 7,062,606 number of instances, we preferred to choose the security camera dataset for our research since it suited our purpose.

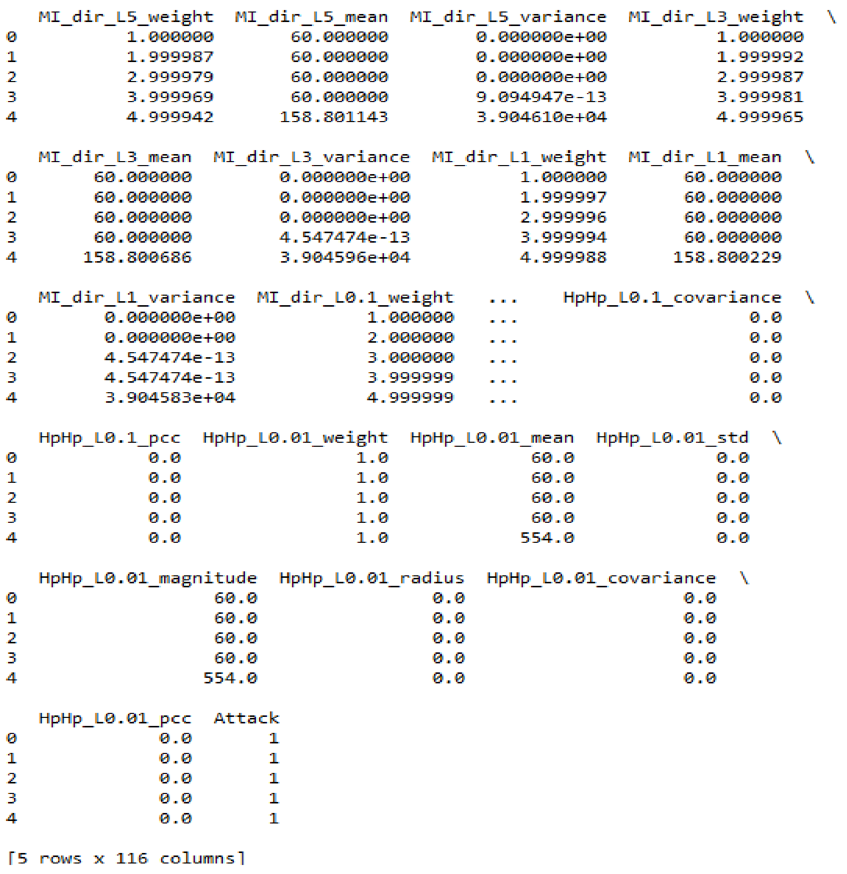
4.1 Transport Layer Data Pre-processing

Figure 3.1 and figure 3.2, shows the Affected and the Benign datasets, for the device Provision_PT_737E_Security_Camera, collected from UCI repository number 00442 for machine learning. Both the Affected and Benign datasets were introduced with a new column called, 'Attack', which denotes the state of the device. While the benign dataset's 'Attack' column was filled with '0s', the Affected dataset's 'Attack' column was filled with '1s'. Both datasets were further processed, which we will be discussing ahead.



```
   MI_dir_L5_weight  MI_dir_L5_mean  MI_dir_L5_variance  MI_dir_L3_weight  \
0          1.000000       60.000000            0.000000          1.000000
1          1.000000      590.000000            0.000000          1.000000
2          1.942585      590.000000            0.000000          1.965145
3          1.000000       60.000000            0.000000          1.000000
4          1.979600       66.061831           35.996177          1.987709

   MI_dir_L3_mean  MI_dir_L3_variance  MI_dir_L1_weight  MI_dir_L1_mean  \
0         60.0000        0.000000e+00          1.000000       60.000000
1        590.0000        0.000000e+00          1.000000      590.000000
2        590.0000        1.160000e-10          1.988244      590.000000
3         60.0000        0.000000e+00          1.000000       60.000000
4         66.0371        3.599862e+01          1.995886       66.012367

   MI_dir_L1_variance  MI_dir_L0.1_weight  ...  HpHp_L0.1_covariance  \
0            0.000000            1.000000  ...                   0.0
1            0.000000            1.000000  ...                   0.0
2            0.000000            1.998818  ...                   0.0
3            0.000000            1.000000  ...                   0.0
4           35.999847            1.999588  ...                   0.0

   HpHp_L0.1_pcc  HpHp_L0.01_weight  HpHp_L0.01_mean  HpHp_L0.01_std  \
0            0.0           1.000000        60.000000        0.000000
1            0.0           6.302352       416.121686      113.526005
2            0.0           7.301607       439.935392      121.229684
3            0.0           1.000000        60.000000        0.000000
4            0.0           1.000000        72.000000        0.000000

   HpHp_L0.01_magnitude  HpHp_L0.01_radius  HpHp_L0.01_covariance  \
0             60.000000            0.00000                    0.0
1            416.121686        12888.15384                    0.0
2            439.935392        14696.63622                    0.0
3             60.000000            0.00000                    0.0
4             72.000000            0.00000                    0.0

   HpHp_L0.01_pcc  Attack
0             0.0       0
1             0.0       0
2             0.0       0
3             0.0       0
4             0.0       0

[5 rows x 116 columns]
```

Figure 4.1: The Benign Dataset header for Transport Layer



```
   MI_dir_L5_weight  MI_dir_L5_mean  MI_dir_L5_variance  MI_dir_L3_weight  \
0          1.000000       60.000000        0.000000e+00          1.000000
1          1.999987       60.000000        0.000000e+00          1.999992
2          2.999979       60.000000        0.000000e+00          2.999987
3          3.999969       60.000000        9.094947e-13          3.999981
4          4.999942      158.801143        3.904610e+04          4.999965

   MI_dir_L3_mean  MI_dir_L3_variance  MI_dir_L1_weight  MI_dir_L1_mean  \
0       60.000000        0.000000e+00          1.000000       60.000000
1       60.000000        0.000000e+00          1.999997       60.000000
2       60.000000        0.000000e+00          2.999996       60.000000
3       60.000000        4.547474e-13          3.999994       60.000000
4      158.800686        3.904596e+04          4.999988      158.800229

   MI_dir_L1_variance  MI_dir_L0.1_weight  ...  HpHp_L0.1_covariance  \
0        0.000000e+00            1.000000  ...                   0.0
1        0.000000e+00            2.000000  ...                   0.0
2        4.547474e-13            3.000000  ...                   0.0
3        4.547474e-13            3.999999  ...                   0.0
4        3.904583e+04            4.999999  ...                   0.0

   HpHp_L0.1_pcc  HpHp_L0.01_weight  HpHp_L0.01_mean  HpHp_L0.01_std  \
0            0.0                1.0             60.0             0.0
1            0.0                1.0             60.0             0.0
2            0.0                1.0             60.0             0.0
3            0.0                1.0             60.0             0.0
4            0.0                1.0            554.0             0.0

   HpHp_L0.01_magnitude  HpHp_L0.01_radius  HpHp_L0.01_covariance  \
0                  60.0                0.0                    0.0
1                  60.0                0.0                    0.0
2                  60.0                0.0                    0.0
3                  60.0                0.0                    0.0
4                 554.0                0.0                    0.0

   HpHp_L0.01_pcc  Attack
0             0.0       1
1             0.0       1
2             0.0       1
3             0.0       1
4             0.0       1

[5 rows x 116 columns]
```

Figure 4.2: The Affected Dataset header for Transport Layer

The following information best describes each of the feature headers: [28]

1. Stream aggregation:
    (a) H: Stats summarizing the recent traffic from this packet's host (IP).
    (b) HH: Stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host.
    (c) HpHp: Stats summarizing the recent traffic going from this packet's host + port (IP) to the packet's destination host + port. Example: (192.168.4.3:1242 to 192.168.4.13:80).
    (d) HH_jit: Stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host.
2. Time-frame (The decay factor Lambda used in the damped window):
    (a) How much recent history of the stream is captured in these statistics.
    (b) L5, L3, L1, etc.
3. The statistics extracted from the packet's stream:
    (a) Weight: The weight of the stream (can be viewed as the number of items observed in recent history)
    (b) Mean: The mean of the stream.
    (c) Std: The standard deviation of the stream.
    (d) Radius: The root squared sum of the two stream's variances.
    (e) Magnitude: The root squared sum of the two stream's mean.
    (f) Cov: An approximated co-variance between the two streams.
    (g) Pcc: An approximated co variance between the two streams.

The dataset characteristic is multivariate and sequential. There are about 115 attributes. The data type is similar for all the attributes, which is float64, and the dataset contains no NaN values.

First and foremost, the udp.csv and benign.csv datasets are read into different dataframes due to the different characteristic of the dataset's, which is the attack phase. The datasets are then introduced with another attribute, labelled as 'Attack'. The 'Attack' column for the benign dataset is filled with 0's which represents that there is no attack. And the 'Attack' column for the attack dataset is filled with 1's, which represent attack. Both dataframes are then concatenated into a single dataframe with which we will be performing further processing. The single dataframe is named df_udp and the features are then normalized because it contains attributes of different scales. We normalized the dataframe so that it eliminates the units of measurement for the data, and helps us to better compare the data from different attributes. We are rescaling the data using standard scaler so that it centers around 0.

This is known as feature scaling and the formula is as follows:

$$x_{new} = \frac{x - \mu}{\sigma} \tag{4.1}$$

where, σ= Standard Deviation, μ=Mean

Dimension reduction techniques are then performed on the dataframe to allow better interpretability and eliminate attributes that are not needed in the prediction. Our target is to reduce the dimensions using four different dimension reduction approaches, to extract different datasets and then compare and analyze the results to see which technique better suits our research.

Four different dimension reduction techniques are then applied onto the dataframe:

- PCA (Principal Component Analysis)
- ICA (Independent Component Analysis)
- High Correlation Filter
- Random Forest Regression

4.2 Dimension Reduction

4.2.1 What is Dimension Reduction

In the world today, huge amount of data is generated on a daily basis. And Iot devices based on their specific use, are known to generate immense amounts of data. These sensor nodes embedded into everyday objects (IoT devices) collect and store millions of real-time data instances every day, such as weather, health, agricultural, traffic information, and in our case the Transport layer information. As the data collection and generation increases on a daily basis, it becomes a challenging task to derive inference from it or even visualizing it to see how each feature behaves. Since our goal is to work with classification problems, there are way too many factors present in the dataset based on which the final classification is to be done. The more the number of features the harder it becomes to interpret the data. This is where the dimension reduction techniques serve their purpose. It processes and reduces the number of random variables under consideration to obtain a set of principal variables. Fig 3.3 shows how a 3-dimensional dataset is reduced to a 2-dimensional dataset; the resulting 2-dimensional dataset is the used to interpret the original 3-dimensional model.

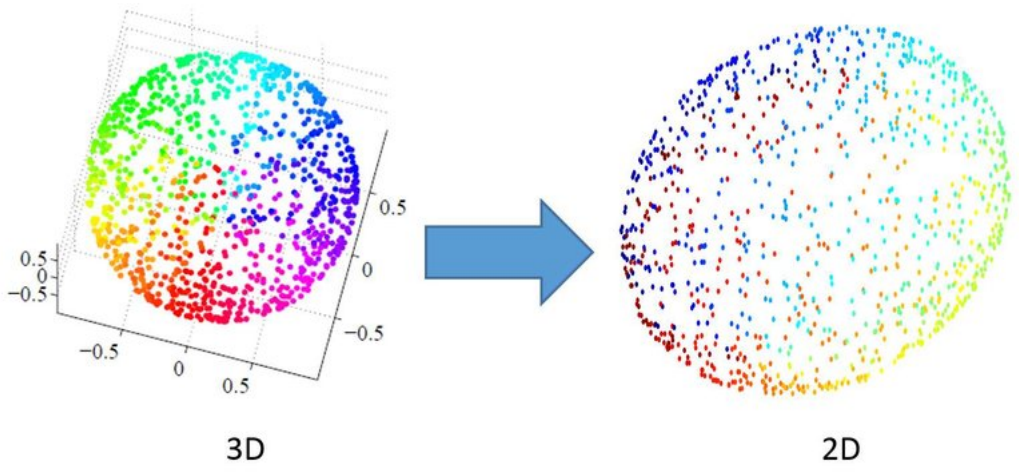


Figure 4.3: Dimension Reduction Example [29]

4.2.2 PCA (Principal Component Analysis)

One of the most well renowned dimension reduction tool used today is known as PCA (Principal Component analysis), it is used as a tool to make a high dimensional (multi attributed) dataset smaller, for better knowledge extraction while making visualization and more meaningful. PCA looks for uncorrelated variables in a high dimensional dataset, and uses a mathematical model to transform it to a smaller dataset with uncorrelated attributes or variables. The tool uses the dataset to find principal components, the first principal component captures the highest variability in the dataset and the second principal component captures the second highest variability, and so on. As shown in (figure 4.4) we are able to explain around 85% variance in the dataset using five components.

A principal component is a linear arrangement of the original predictors in a dataset. The first principal component accounts for as much variability in the data as possible, and the remaining variability is addressed by each of the successor components. Two different components are considered to be uncorrelated if their direction is orthogonal [30].

For first Principal Component $\emptyset^1$, with normalized set of predictors β^1,β^2,β^3,.....,β^x

$$\emptyset^1 = \alpha^{11}\beta^1 \ + \ \alpha^{21}\beta^2 \ + \ \alpha^{31}\beta^3 \ + \ ..... \ + \alpha^{x1}\beta^x \tag{4.2}$$

Where α^{x1} is the loading vector compromised of loading's (α^1,α^2...) for the first principal component.

The second principal component is as follows:

$$\emptyset^2 = \alpha^{12}\beta^1 \ + \ \alpha^{22}\beta^2 \ + \ \alpha^{32}\beta^3 \ + \ ..... \ + \alpha^{x2}\beta^x \tag{4.3}$$

The sequence follows on based on the number of preferred components.

After splitting the original concatenated dataset into test and train sets, we perform PCA with n_components=5, so the dimension of the test and train dataset reduces down to 5 from 115. The figure below shows the Component-wise and Cumulative Explained Variance. While, the blue line represents component-wise explained variance, the orange line represents the cumulative explained variance.

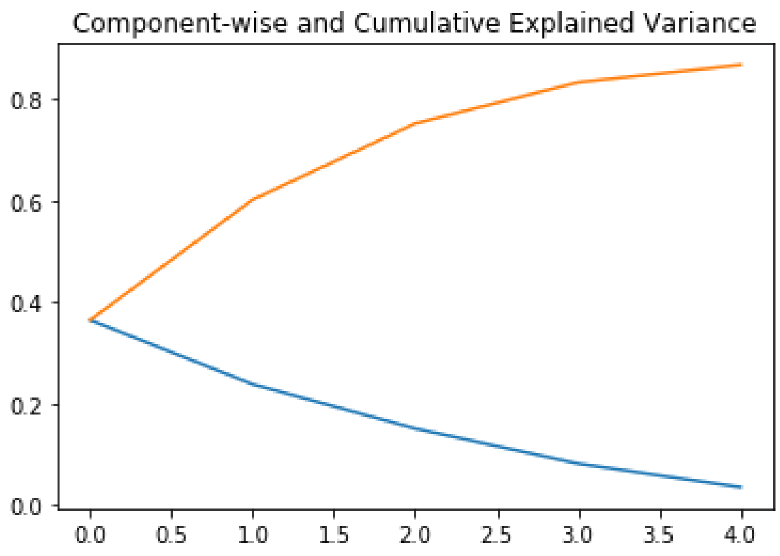
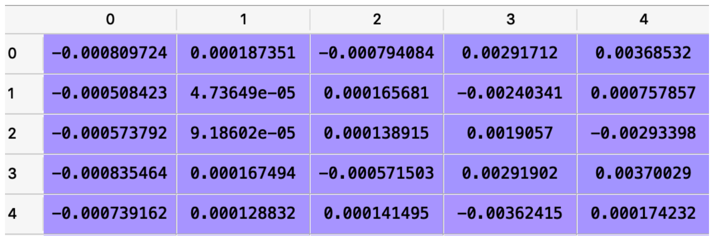


Figure 4.4: Component wise and Cumulative Explained Variance for PCA

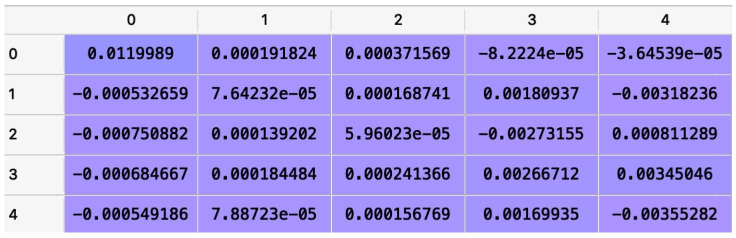
4.2.3 ICA (Independent component Analysis)

Another dimension reduction technique we are using is ICA (Independent Component Analysis). It is a computational approach to separate a multivariate dataset into its hidden subcomponents. The difference between ICA and PCA (Principal Component Analysis) is that, while PCA searches for uncorrelated variables, ICA examines for independent variables. While PCA shows that two variables are uncorrelated, which means there is no linear relationship between them, ICA shows that a variable may be independent given that they are not dependent any other variable. The original dataset is split into training and test set and is then normalized. We are performing ICA to derive two datasets of 5 independent component from the test and training dataset consisting of 115 attributes. The results are as followed:



	0	1	2	3	4
0	−0.000809724	0.000187351	−0.000794084	0.00291712	0.00368532
1	−0.000508423	4.73649e−05	0.000165681	−0.00240341	0.000757857
2	−0.000573792	9.18602e−05	0.000138915	0.0019057	−0.00293398
3	−0.000835464	0.000167494	−0.000571503	0.00291902	0.00370029
4	−0.000739162	0.000128832	0.000141495	−0.00362415	0.000174232

Figure 4.5: Training set header after ICA

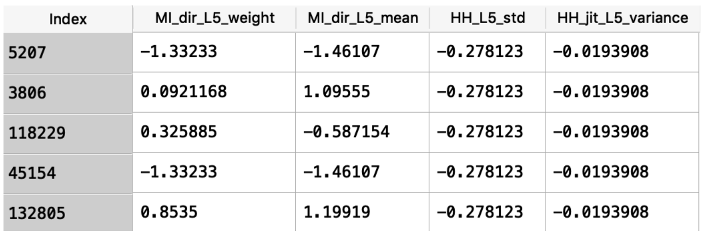


	0	1	2	3	4
0	0.0119989	0.000191824	0.000371569	-8.2224e-05	-3.64539e-05
1	-0.000532659	7.64232e-05	0.000168741	0.00180937	-0.00318236
2	-0.000750882	0.000139202	5.96023e-05	-0.00273155	0.000811289
3	-0.000684667	0.000184484	0.000241366	0.00266712	0.00345046
4	-0.000549186	7.88723e-05	0.000156769	0.00169935	-0.00355282

Figure 4.6: Test set header after ICA

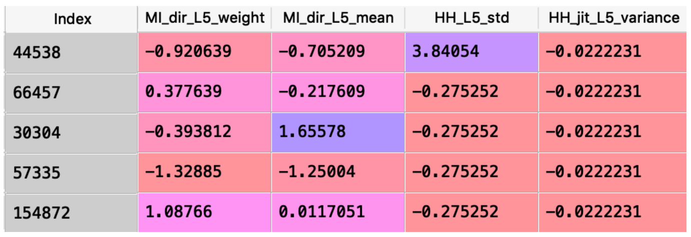
4.2.4 High Correlation Filter

The 3rd dimension reduction technique in use is the high correlation filter. High correlation between two components suggests that they have similar trends between them and are likely to carry similar information, hence increases data redundancy. Since we can determine the correlation between variables that are numerical in nature, we can then maintain a threshold for the correlation coefficient, and if it crosses that value, we can then drop one of the variables. We are maintaining a threshold point for the correlation coefficient at 0.5. The number of correlated features we found after performing high correlation was 111, so after dropping those features it formed datasets of 4 features from the test and training sets. Figures 3.2.4 and 3.2.5 show the resulting train and test sets after performing High Correlation filter.



Index	MI_dir_L5_weight	MI_dir_L5_mean	HH_L5_std	HH_jit_L5_variance
5207	-1.33233	-1.46107	-0.278123	-0.0193908
3806	0.0921168	1.09555	-0.278123	-0.0193908
118229	0.325885	-0.587154	-0.278123	-0.0193908
45154	-1.33233	-1.46107	-0.278123	-0.0193908
132805	0.8535	1.19919	-0.278123	-0.0193908

Figure 4.7: Training set header after High Correlation Filter



Index	MI_dir_L5_weight	MI_dir_L5_mean	HH_L5_std	HH_jit_L5_variance
44538	-0.920639	-0.705209	3.84054	-0.0222231
66457	0.377639	-0.217609	-0.275252	-0.0222231
30304	-0.393812	1.65578	-0.275252	-0.0222231
57335	-1.32885	-1.25004	-0.275252	-0.0222231
154872	1.08766	0.0117051	-0.275252	-0.0222231

Figure 4.8: Test set header after High Correlation Filter

4.2.5 Random Forest Regression

Lastly, we have looked in to Random Forest Regression for dimension reduction. It is known as one of the most widely used and preferred technique for feature selection. Random forest is a part of ensemble method that can perform both regression and classification by using multiple decision trees and a method which is known as Bootstrap Aggregation. It relies on training each decision tree on multiple unalike samples where sampling is done through replacement, it helps us to better understand the bias and the variance. While it is a very powerful, supervised technique for feature selection, yet it is very simple to perform. It comes with in-built feature importance and by setting a threshold, we can perform dimension reduction. In our case we are using a threshold of 0.25. Figure 3.2.6 represents the feature importance after performing Random Forest Regression.

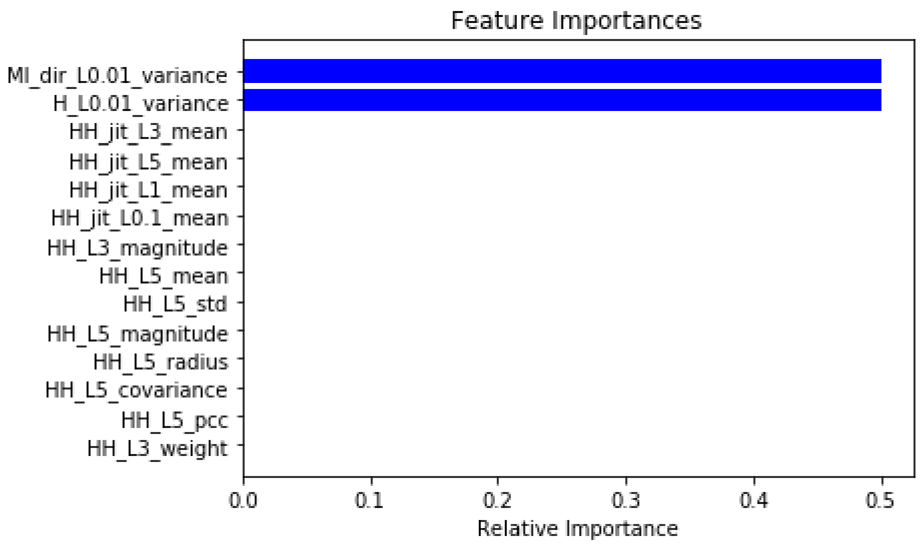
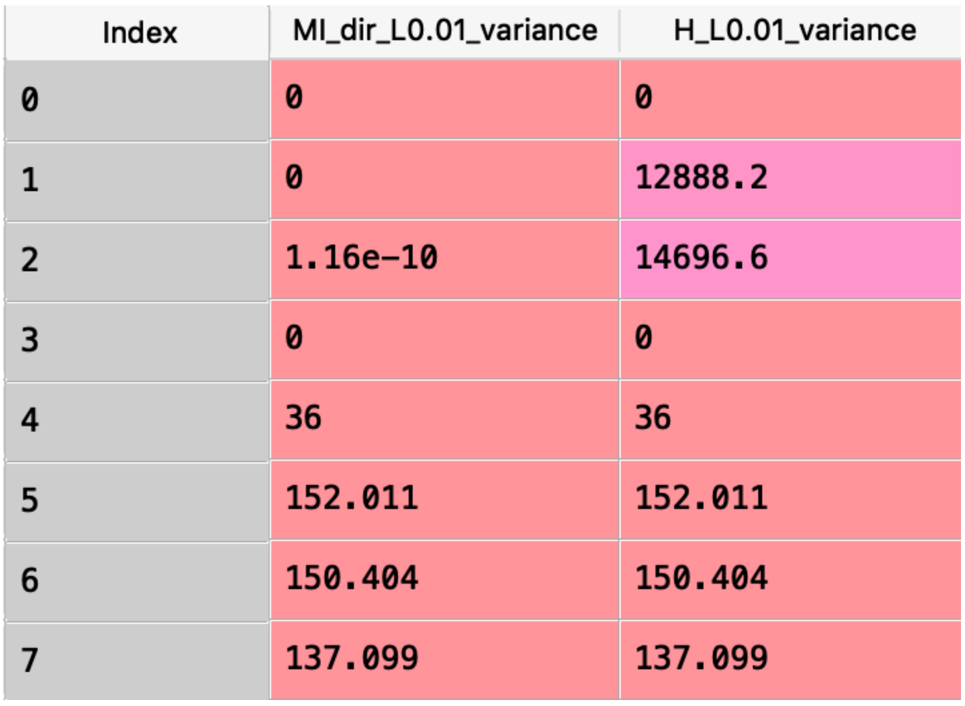


Figure 4.9: Feature Importance from Random Forest Regression

At a threshold of 0.25 which is the preferred threshold for feature selection under Random Forest Regression, a 2-dimensional dataset forms consisting of the attributes: MI_dir_L0.01_variance, H_L0.01_variance. For Random Forest Regression the test-train splitting is performed after the feature selection process. Figure 3.2.7 represents the state of the original concatenated dataset after Random Forest Regression is performed.



Index	MI_dir_L0.01_variance	H_L0.01_variance
0	0	0
1	0	12888.2
2	1.16e-10	14696.6
3	0	0
4	36	36
5	152.011	152.011
6	150.404	150.404
7	137.099	137.099

Figure 4.10: Dataset header from Random Forest Regression

Chapter 5

Research Methodology

Although there has already been some significant investigation with regards to this field, with multiple deep learning, neural networking and Machine learning approaches prevailing. We have chosen the following Machine Learning approaches like: (kNN, Random Forest, SVM, Naïve Bayes) to execute our research. From previous papers it has been evident that some significant amount of work has been done with approaches like Decision Trees, Logistic Regression and Random Forest. We have further explored a few more classifiers with some recurrent approaches, and executed them following up from a handful of feature selection processes. Feature engineering is one of the core aspects to the success in Machine Learning. Proper feature engineering would eventually produce better results, and create better predictors. In this chapter we are going to review, each of our four classifier algorithms. This chapter contains the justification for using each of the four algorithms, based on the fact that our data is linear or non-linear. This chapter dives deep into each of those algorithm and describes their purpose in our research.

5.1 K-Nearest Neighbour

kNN is a basic algorithm that stores every single accessible case and classifies new cases dependent on a likeness measure. It can solve problems related to regression, prediction, and classification of data, statistical estimation, pattern recognition, and intrusion detection [44]. It is effective in real-life problems because of its non-parametric nature which means it does not make any radical prognosis about the distribution of data. As a result, the model structure is ordained from the data. It is useful for non-linear data. It is commonly preferred for ease of illustrating output and the fast calculation. It falls under the supervised machine learning category which relies on labeled sample data. This algorithm takes labeled data as input. After learning a function from the given input, it can then propagate an output when a new unlabeled data is given. The algorithm stores training samples in an array of data points which refers to each element of this array conveys as a tuple (x,y). After determining distance from distance functions like Euclidean, Manhattan, Minkowski for each element in the array, a set of S is made of K smalled distance obtained.

$$Euclidean, \sqrt{\sum_{i=1}^{k} (x(i) - y(i))^2} \tag{5.1}$$

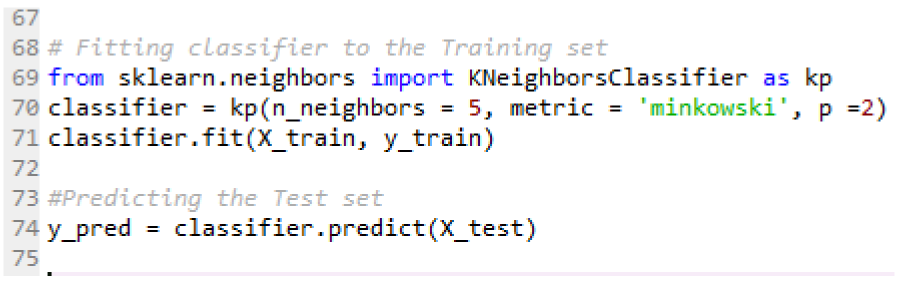
$$Manhattan, \sum_{i=1}^{k} |x(i) - y(i)| \tag{5.2}$$

$$Minkowski, (\sum_{i=1}^{k} (|x(i) - y(i)|)^{q})^{\frac{1}{q}} \tag{5.3}$$

Every one of these separations relates to a previously grouped information point. Then it returns to the majority label among S. K nearest algorithm. It stores the whole preparing dataset which it utilizes as its portrayal. It doesn't gain proficiency with any model. It makes forecasts in the nick of time by ascertaining the likeness between an information test and each preparation occurrence.

5.1.1 Implementation of kNN on Transport Layer Dataset

After pre-processing dataset, we implemented k-Nearest Neighbour on transport layer.Transport layers work straightforwardly inside the layers above to convey and get information without mistakes. The most crucial part of implementing kNN algorithm is to find out the value of k. It plays a significant role on the result and the level of accuracy. In our case, we have worked with 5 neighbours and we took the default value of k while implementing the algorithm on pre-processed dataset as we got highest accuracy doing so. We received an accuracy of 99.97% with that default k's value which is very satisfactory. Furthermore we used Minkowski distance metric. Minkowski distance is a metric in Normal vector space and S normal vector has some properties like zero vector, scalar factor and triangle inequality which keeps the standard prompted metric-homogeneous and interpretation invariant [31]. Minkowski distance is a generalization of the Euclidean and Manhattan distances which takes two points of p. The Minkowski distance between two variables X and Y. The case where p = 1 is equivalent to Manhattan distance and the case where p = 2 is equivalent to Euclidean distance.The code mentioned below exhibits our implementation of kNN.



```

# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier as kp
classifier = kp(n_neighbors = 5, metric = 'minkowski', p =2)
classifier.fit(X_train, y_train)

#Predicting the Test set
y_pred = classifier.predict(X_test)
.
```

Figure 5.1: The Code for kNN classification

5.2 Naïve Bayes

Naive Bayes is a probabilistic machine learning algorithm depends on the Bayes Theorem, utilized in a wide assortment of order errands. Regular applications of Naïve Bayes incorporate separating spam, arranging records, estimation expectation, etc. The name ‘Naïve’ is utilized because it accepts the highlights that go into the model is free of one another. That is changing the estimation of one element, which doesn’t legitimately impact or change the estimation of any of the different highlights utilized in the calculation. This algorithm works with Bayes theorem which can be illustrated as:

$$P(A|B) = P(B|A).P(B)/P(A) \tag{5.4}$$

Here, P (A—B) means posterior probability, P (B—A) indicates the likelihood of the event, P(B) represents class prior probability and P(A) refers to predictor prior probability. A is is a dependent feature vector (of size n) where:

$$A = (a1, a2, a3.\dots.an) \tag{5.5}$$

The diverse NB classifiers vary fundamentally by the suppositions which are made in regards to the dissemination of the conditional probability P(Ai|B) [32]. There is a huge scope to Naïve Bayes. Since it is a probabilistic model, the calculation can be coded up effectively and the predictions can be made in real-time very quickly. Moreover, it is effectively adaptable, fast learner and is traditionally used for the calculation of decisions in real-world applications, that are required to react to a client’s solicitations quickly.

5.2.1 Implementation of Naïve Bayes on Transport Layer

There are three kinds of Naive Bayes models underneath the scikit-learn library for implementation: Gaussian, Multinomial, Bernoulli.

Gaussian is used in classification and it assumes that facets observe a normal distribution. When the predictors take up a continuous fee and are not discrete, we expect that these values are sampled from a Gaussian distribution. Multinomial is used for discrete counts. Multinomial NB implements the naive Bayes algorithm for multinomial allotted facts and is one of the two traditional naive Bayes editions used in text classification. Bernoulli implements the naive Bayes education and classification algorithms for information that is distributed in accordance with multivariate Bernoulli distributions. There might also be more than one features but everyone is assumed to be a binary-valued (Bernoulli, Boolean) variable. Therefore, this class requires samples to be represented as binary-valued function vectors; it surpassed any different variety of data, a Bernoulli NB instance may binarize it’s enter (depending on the binarize parameter). The binomial mannequin is beneficial if function vectors are binary [33].

Here, after data processing and studying the type of our dataset we got that all the features are following the normal distribution, hence we implemented Gaussian. Gaussian Naïve Bayes can work better with a normal distribution. When dealing with the Gaussian Naïve Bayes classifier, the consequence model will have a high-performance with excessive coaching pace with the skills to predict the chance of the feature that belongs to the Zk class. The purpose of the Naïve Bayes classifier to compute the $\hat{i}$th statement would be with the aid of computing the following probability [34].

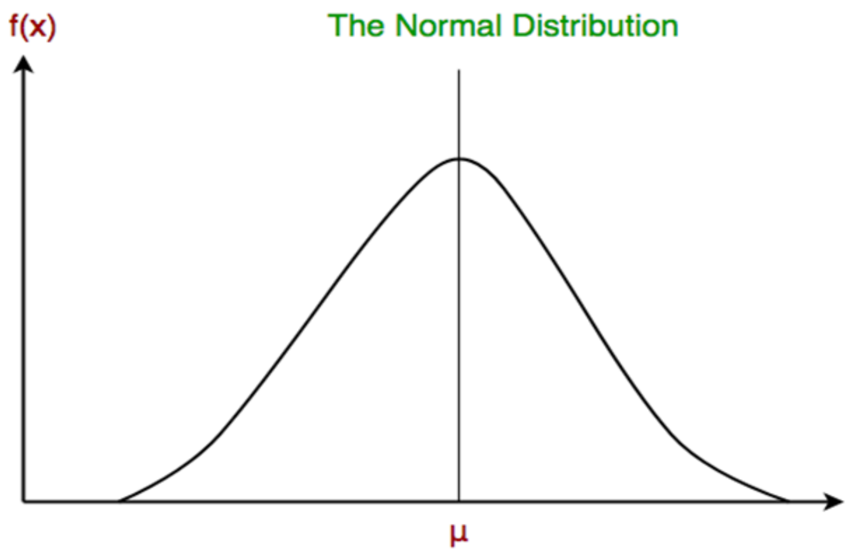
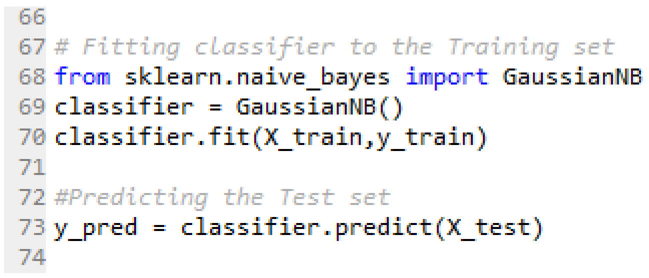


Figure 5.2: Normal distribution for Gaussian Naïve Bayes [35]

Probability of $(Zk|X(i)$ .The $X(i)$ can be illustrated as the vector of the number of features with the number of components as $[X(i,0), X(i,1), X(i,2), X(i,3)]$ [36]. The following figure exhibits our implementation of Naïve Bayes Classifier.



```

# Fitting classifier to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train,y_train)

#Predicting the Test set
y_pred = classifier.predict(X_test)

```

Figure 5.3: Code for Naïve Bayes Classifier

5.3 Random Forest

The Random Forest classifier is termed as an ensemble algorithm as it encapsulates more than one algorithm which might be of same or of different kind. It comprises of a large number of individual decision trees that operate together at the same time. Each individual tree produces a class prediction and the class with the most votes is chosen as the model's prediction by random forest. Each tree derives its input from the sampled data of the dataset and from the features available; a subset of features is selected for each node [45]. The trees do not involve any pruning. The advantages of random forest are its capability of efficiently working on large databases and still maintaining a fast computational speed; however it is prone to overfitting.

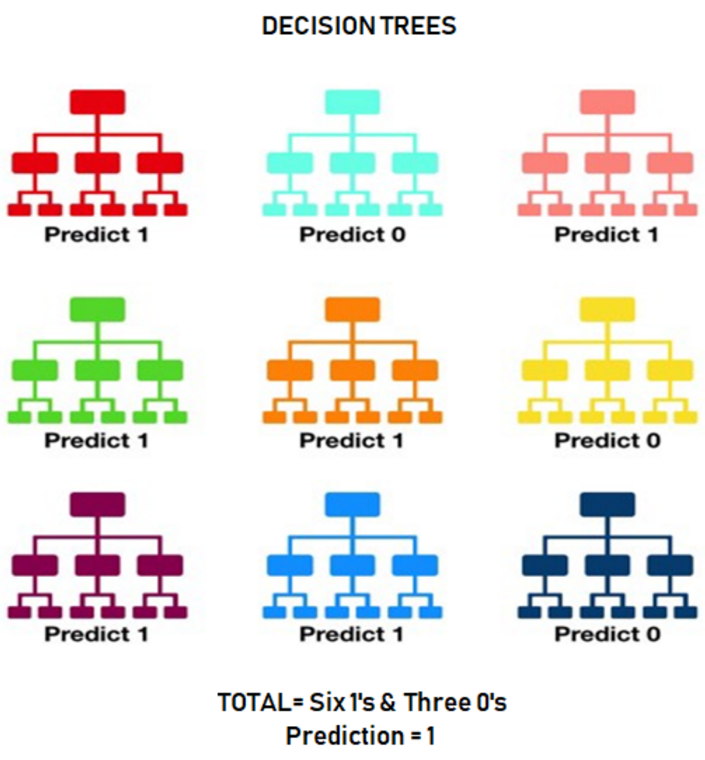
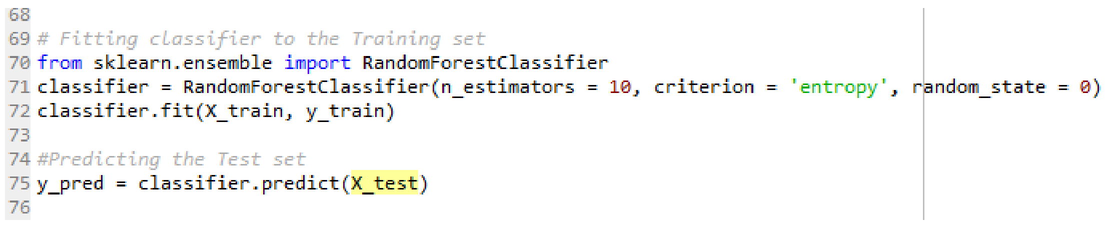


Figure 5.4: Visualisation of a Random Forrest algorithm [37]

5.3.1 Implementation of Random Forest on Transport Layer

Firstly, the classifier works on the dataset by creating multiple different decision trees, then it trains each individual decision trees on multiple divergent samples where the sampling is done through replacements. The process in turn aids us in having a better interpretation for the bias and variance. Random Forest Classification comes with an in built function known as feature importance. This is what makes this classification technique simple yet very powerful. The feature importance method is then applied onto the original concatenated dataset before normalization and the test-train split is performed. This exhibits the feature importance of each individual attribute in terms of its relative importance based on the dataset. Since we already have the relative importance of the attributes we can then make use of a threshold which will help us to drop any attributes that fall below this margin. For our research, we are setting the threshold at 0.25. The figure below shows how we implemented Random Forest Classification as our predictor.



```

# Fitting classifier to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

#Predicting the Test set
y_pred = classifier.predict(X_test)

```

Figure 5.5: The Code for Random Forest classification

5.4 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm which classifies data into several groups. Data is then classified into different classes depending on their individual groups. Then the data is plotted on a n-dimensional plane where n is the number of features provided as input. These classes are then differentiated from each other using a hyper-plane. The aim is to find a plane consisting of the maximum margin where the margin is defined as the distance between data points of both classes and the hyper-plane [46]. The data points closest to the margin are called support vectors. The hyper-plane which has the highest distance from the closest data points of each class is considered as the optimal hyper-plane. Considering the dimension to belong to a data set of 2(two) input features, the flowing steps and equations elaborate the process of constructing a hyper-plane.
The optimal hyper plane has the equation:

$$w * x - b = 0 \tag{5.6}$$

In above mentioned equation, w is the normal vector to the hyper-plane, while x can vary for each instance and b is the distance between the nearest point to the hyper-plane and the hyper-plane itself.

$$w * x - b >= 1 \tag{5.7}$$

$$w * x - b <= 1 \tag{5.8}$$

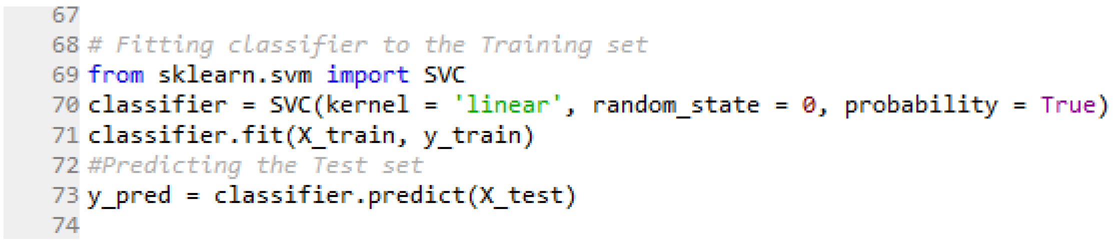
If the resultant value of the equation is positive, then the hyper-plane belongs to the positive class and so any value greater than 1 will showcase a data point belonging to the positive class. Similarly, the value being negative indicates the hyper-line is of the negative class. So any value less than -1 will prove to be data from the negative class.
Any variable that effects the degree of maintenance of constraints is called a Regularization Parameter [38]. Consider the following equation:

$$min \left\| w \right\|^2 + C * \sum_{i}^{N} (\varepsilon i) \tag{5.9}$$

The equation determines the type of margin derived from the dataset. A steep value of C will result in a narrow margin and a more distinct separation. On the contrary, if the C showcases an infinite value, the margin will be labelled as a hard one. The evident impact of the range of values of C in this equation makes it the Regularisation Parameter of this instance.

5.4.1 Implementation of Support Vector Machine on Transport Layer

For our research, the training set is fitted using SVM. Our training set consists of 152881 instances after the transport layer dataset was split into train and test sets following the ratio 7:3. Due to the size of the training set being immense, the SVM algorithm will not work properly if the data is very continuous, which leads us to the conclusion that even with a narrow margin, it would still make mistakes. The specific kernel that we have chosen was tested based on the fact that it supported our cause the best. For our training set we have used a linear kernel to draw the hyper plane that would separate the two classes which are the abnormal/attack phase and the normal/benign phase. The following figure exhibits how we implemented the Support Vector Machine Classification.



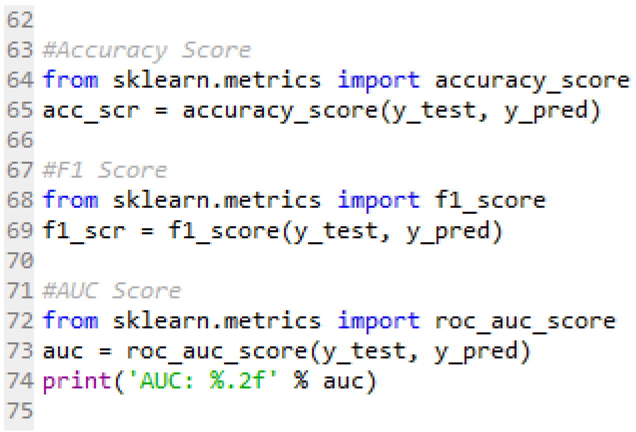
```

# Fitting classifier to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0, probability = True)
classifier.fit(X_train, y_train)
#Predicting the Test set
y_pred = classifier.predict(X_test)

```

Figure 5.6: The Code for SVM classification

Chapter 6

Result Analysis

In this Chapter we will be review our results. We will compare each different Dimension reduction technique and the classification techniques, based on these parameters: F1 score, Accuracy, AUC (Area Under the ROC Curve) Score. These accuracy parameter entries are given in terms of percentage. We will also be comparing each individual algorithm based on their prediction time, which is the time it takes to generate a prediction given the test data-set, under the four different dimension reduction techniques. Finally, we will end this chapter with a comparison of the algorithms based on the confusion matrices they have presented us with based on their feature selection process. The experiment is carried using Spyder (Python 3.6) in Mac OS environment on 8GB RAM and 2.3 GHz Intel Core i5. Moreover, for the transport data, we introduced two labels, '0' and '1' and 0 refers to Normal/Non-attack and 1 refers to Abnormal/Attack. It is also noteworthy that the entire data-set was split 7:3 where 70% of the data was used to train the classifiers and 30% of the data was used to test the trained classifiers.



```python

#Accuracy Score
from sklearn.metrics import accuracy_score
acc_scr = accuracy_score(y_test, y_pred)

#F1 Score
from sklearn.metrics import f1_score
f1_scr = f1_score(y_test, y_pred)

#AUC Score
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print('AUC: %.2f' % auc)

```

Figure 6.1: The Code Segment for determining each metric

6.1 Comparison using F1 score

The F1 score is an accuracy parameter that is based on the sensitivity and precision of a confusion matrix [39]. Sensitivity shows us how much of the actual positives our model has captured out of all the actual positives, which is the sum of true positives and false negatives. While precision is the calculation that determines how much of the predicted positive is actually positive. Combining both of these,

Classification		F1 Score (percentage)			
		Dimension Reduction Techniques			
		PCA	ICA	High Correlation	Random Forest Regression
1	KNN	99.98	99.98	99.97	100
2	Naïve Bayes	99.56	94.25	99.82	99.86
3	RFC	100	99.99	99.99	99.99
4	SVC	100	99.57	99.97	100

Table 6.1: F1 Scores

we get the function that determines the F1 Score.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{6.1}$$

$$Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{6.2}$$

$$F1score = 2 \times \frac{Precision * Sensitivity}{Precision + Sensitivity} \tag{6.3}$$

The aforementioned Table 5.1, shows a comparison of each individual algorithm under their respective dimension reduction technique, based on their generated F1 Scores given in terms of percentage. If we observe the data present in the table we can see in a comparison between the classifiers, Random Classifier performed the best. And Random Forest Regression prevailed over all other dimension reduction techniques.

6.2 Comparison using Accuracy

It is metric that is used to compare between classification models. It is the most widely used technique to measure the precision of a classifier algorithm among all. It determines the fraction of correct predictions made by a model to the total number of predictions it made [40]. The following equation shows how it is calculated.

$$Accuracy = \frac{True\ Positive + True\ Negatives}{Total\ predictions} \tag{6.4}$$

Classification		Accuracy (percentage)			
		Dimension Reduction Techniques			
		PCA	ICA	High Correlation	Random Forest Regression
1	KNN	99.98	99.98	99.96	100
2	Naïve Bayes	99.38	91.27	99.74	99.79
3	RFC	100	99.99	99.98	99.99
4	SVC	100	99.38	99.96	100

Table 6.2: Accuracy

Table 5.2, shows the accuracy that each classifier has presented us with. Similar to the F1 score results, the Random Forest Classifier has once again generated the best accuracy, and Random Forest Regression has proved to be the best dimension reduction technique.

6.3 Comparison using AUC (Area under ROC curve)

This is another performance measurement metric for classifiers. ROC (Receiver Operating Characteristics) curve is a probability curve that distinguishes between the true positive rate and false positive rate [41]. And AUC determines the fraction of area that falls underneath the ROC curve. The following table 5.3, does a brief comparison of the AUC produced by each classification model and their respective feature selection process.

Classification		AUC (percentage)			
		Dimension Reduction Techniques			
		PCA	ICA	High Correlation	Random Forest Regression
1	KNN	99.97	99.98	99.96	100
2	Naïve Bayes	98.93	84.70	99.72	99.86
3	RFC	99.99	99.99	99.98	99.99
4	SVC	99.99	98.91	99.94	100

Table 6.3: AUC Scores

The ensemble algorithms Random Forest Regression and Random Forest Classifier, performs the best among all. Random Forest model has shown great promise by producing the best possible results in all three accuracy measuring categories, among all other techniques. All the models have generated a nearly perfect ROC curve, with the only anomaly being ICA(Independent Component Analysis) followed up by Naïve Bayes Classification. It occurs due to Naïve Bayes's tendency to learn fast, and it's assumption of the features being independent. The following graph shows the ROC curve for the aforementioned Classifier.

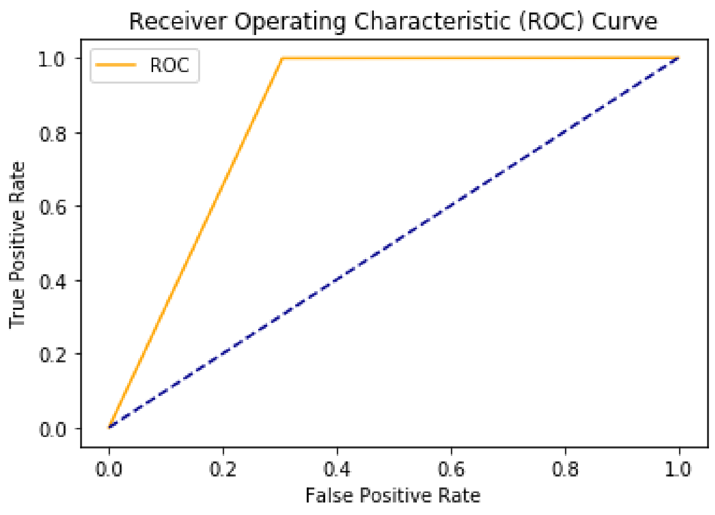


Figure 6.2: The ROC curve for Naive Bayes Classification under ICA

The following graph shows the ROC curve for Naïve Bayes under PCA.

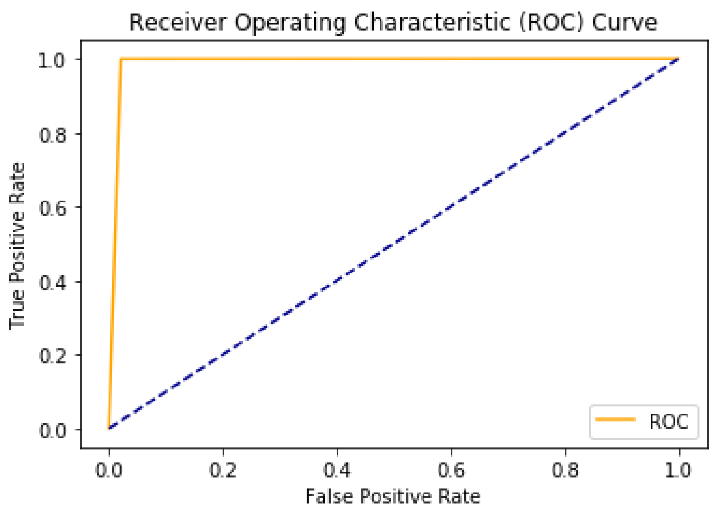


Figure 6.3: The ROC curve for Naive Bayes Classification under PCA

The rest generated a near perfect ROC curve. The following figure shows the ROC that was generated by kNN, Random Forest and SVM.

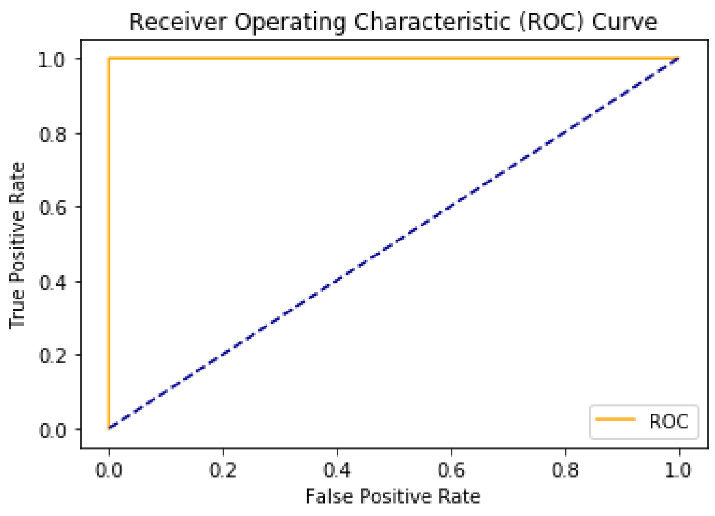


Figure 6.4: The ROC curve for Random Forest, kNN, SVM

6.4 Comparison Based on Prediction time

For measuring, or timing the prediction phase of each classifier we used the timeit function. We assumed that, when put in a real-life situation, the speed at which our model predicts an outcome will prove to be most beneficial. Table 5.4 exhibits the results generated by each model in terms of seconds.

Classification		Time (Seconds)			
		Dimension Reduction Techniques			
		PCA	ICA	High Correlation	Random Forest
1	KNN	1.9265220	1.8393085	3.5923713	1.3710406
2	Naïve Bayes	0.0254196	0.0205383	0.0122119	0.0074057
3	RFC	0.0306846	0.0290179	0.0317149	0.0291516
4	SVC	0.0324104	30.0314590	0.0915311	0.0446332

Table 6.4: Prediction Time

While Naïve Bayes Classifier has proved to be the fastest predictor among all other classifiers, it suffers greatly in terms of accuracy, which might prove to be harmful in the long run. Although Random Forest Classifier comes in second in terms of speed of prediction, it produces results with the best possible accuracy. So, even though it takes a few milli-seconds longer than Naïve Bayes Classifier to predict an outcome, it the best learner among all the other classifiers.

6.5 Comparison of the Confusion Matrix

While the accuracy measuring metrics provide a general overview, the confusion matrix shows the actual results of the prediction. It divides the results into four different categories, True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). True Positive and True Negative shows us the fraction of test data that have been correctly predicted. While False Positive and False Negative shows the segment of test data, that has been predicted erroneously by the classifier. The following figure shows how each category is represented. And the code segment in Figure 5.3 that determine the confusion matrix.

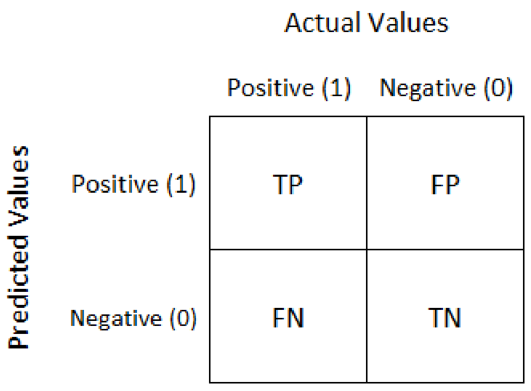
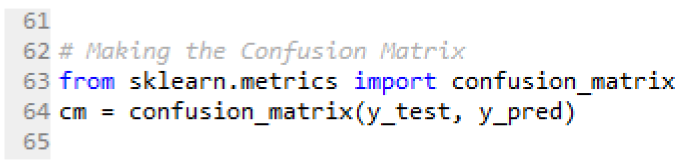


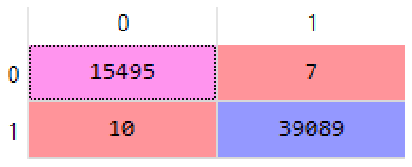
Figure 6.5: The Confusion Matrix



```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

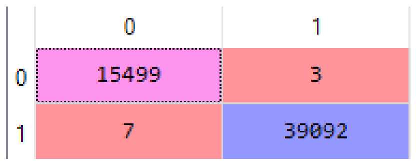
Figure 6.6: The Code Segment for Confusion Matrix

6.5.1 Analysis of k-Nearest Neighbour



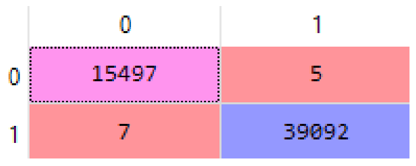
	0	1
0	15495	7
1	10	39089

Figure 6.7: High Correlation



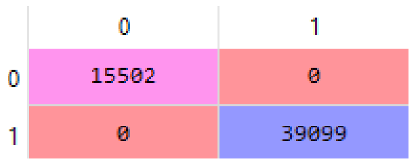
	0	1
0	15499	3
1	7	39092

Figure 6.8: ICA



	0	1
0	15497	5
1	7	39092

Figure 6.9: PCA

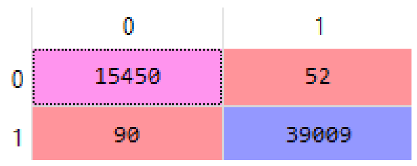


	0	1
0	15502	0
1	0	39099

Figure 6.10: Random Forest

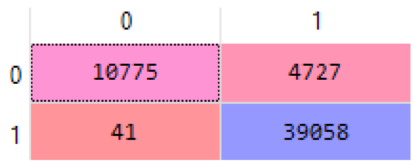
From the Figures (5.4, 5.5, 5.6, 5.7), we can see that for dimension reduction with Random Forest, k-NN provides with perfect results, having 0 false positives or 0 false negatives. It also consistently provides a good AUC score and makes fewer errors compared to the other algorithms. In case of High Correlation (Figure 542), it produces the most number of false negative instances which only amount to 10. For both PCA and ICA (Figures 5.5, 5.6), it has 7 instances where it predicts wrongly that there was no attack when actually there was one. Overall, this provides with the least amount of errors; however, k-NN algorithm also takes the most amount of time among all the other algorithms regardless of the dimension reduction technique used. k-NN algorithm works very well in situations where the dataset is well-separable into their respective classes. Since the data from the Transport Layer is continuous, k-NN provides with the best results because k-NN groups them well together. Due to k-NN being a lazy learner and doing the classification work mostly at the testing phase, k-NN takes more time compared to other algorithms.

6.5.2 Analysis of Naïve Bayes



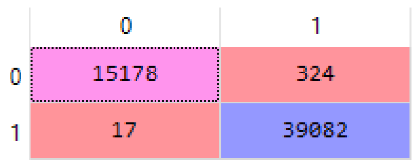
	0	1
0	15450	52
1	90	39009

Figure 6.11: High Correlation



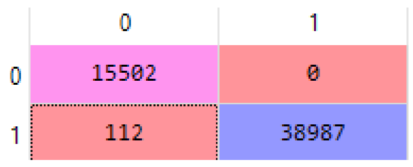
	0	1
0	10775	4727
1	41	39058

Figure 6.12: ICA



	0	1
0	15178	324
1	17	39082

Figure 6.13: PCA

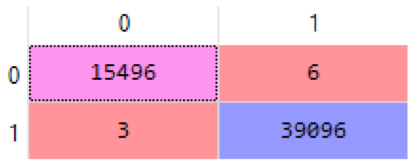


	0	1
0	15502	0
1	112	38987

Figure 6.14: Random Forest

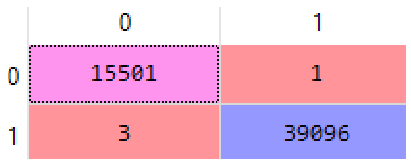
The Figures (5.8, 5.9, 5.10, 5.11) showcase the confusion matrix for Naïve Bayes Classifier under four different dimension reduction techniques. From Figure 5.10 we can see that under PCA (Principal Component Analysis) we have generated 17 false negatives and 324 false positives while the rest of the data instances have been predicted correctly. The false positives indicate that the device is not be under attack while the algorithm has predicted that it has undergone an attack and although the prediction may be wrong, we will not suffer much from it. Moreover, the low number of false negatives indicates that the algorithm will make fewer mistakes in case of an attack scenario.
Random Forest gives us much better result in terms of generated false positives since it has 0 instances of false positives (as shown in Figure 5.11). However, the number of false negatives are much higher and it is at 112 instances. Hence it does not outperform ICA dimension reduction results in that account.
From Figure 5.8, we see that High Correlation gives us moderately accurate results if compared between the figures with 90 instances of false negatives and 52 instances of false positives. In comparison, ICA provides with the least reliable data where it provides with 41 instances of false negatives and 4727 instances of false positives (shown in Figure 5.9).
Naïve Bayes' eagerness to learn is evident in the time taken to execute, and hence it is optimal for making real-time classifications; although it also provided us with more erroneous results than the other classifiers that were considered.

6.5.3 Analysis of Random Forest Classification



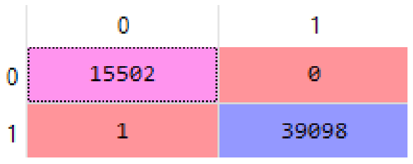
	0	1
0	15496	6
1	3	39096

Figure 6.15: High Correlation



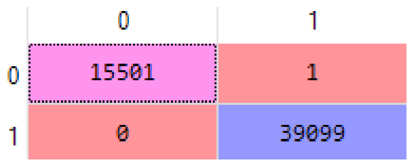
	0	1
0	15501	1
1	3	39096

Figure 6.16: ICA



	0	1
0	15502	0
1	1	39098

Figure 6.17: PCA

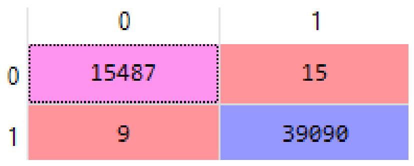


	0	1
0	15501	1
1	0	39099

Figure 6.18: Random Forest

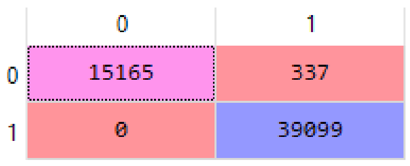
The aforementioned Figures (5.12, 5.13, 5.14, 5.15) exhibit the confusion matrix for Random Forest Classification under four different dimension reduction techniques. From figure 5.12 we can see that under High Correlation we have generated 3 false negatives and 6 false positives, the rest of the data instances have been predicted correctly. The false positives define that the device may not be under attack while the algorithm has predicted that it has undergone an attack, although the prediction may be wrong but we may not suffer much from it. Although 3 instances of the test set have been predicted as false negatives, where the system is under attack but the algorithm has predicted that it is not, this where we would probably suffer more since, false negatives affect us more than false positives.
ICA (Independent Component Analysis) gives us much better result in terms of generated false positives, although the number of false negatives remain the same. Hence it performs better than High Correlation yet suffers the same problem in terms of false negatives.
Both, PCA (Principal Component Analysis) and Random Forest Regression have generated much more accurate results, although the system would suffer a little under PCA rather than Random Forest Regression due to the 1 false negative prediction.
Collectively the Random Forest Classification has presented us with above par result based on the confusion matrix, in comparison to the other classification techniques. However, it is prone to over-fitting which indicates that it may perform erroneously on new data due to different type of noise being present there.

6.5.4 Analysis of Support Vector Machine



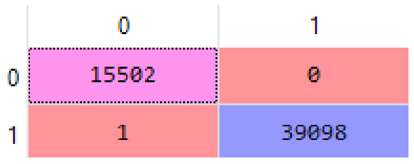
	0	1
0	15487	15
1	9	39090

Figure 6.19: High Correlation



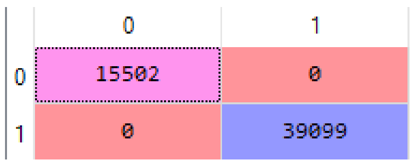
	0	1
0	15165	337
1	0	39099

Figure 6.20: ICA



	0	1
0	15502	0
1	1	39098

Figure 6.21: PCA



	0	1
0	15502	0
1	0	39099

Figure 6.22: Random Forest

In case of SVM (Support Vector Machine), we have generated the aforementioned confusion matrices. We can clearly see that for our research Support Vector Machine performs and produces the best result under Random Forest Regression, and accurately predicts each and every instance of the test set, as represented by figure 5.19. It also performs quite good under PCA (Principal Component Analysis) figure 5.18, while generating only 1 false negative result which may seem as a drawback, yet can be overlooked. Although, the system would suffer largely under ICA (Independent Component analysis), generating 337 false positives as shown in figure 5.17, which would cause a great deal of to the system by predicting that the system is under attack, whereas it is safe. Apart from that it predicts the rest of the instances of the test set accurately, with no false negatives which are more harmful than false positives. Furthermore, it goes on to show us that the hyperplane that is drawn in SVM cannot properly separate the two classes, since the sigmoid hyperplane used to separate the data according to their classes, fails at separating the attack and benign data. Figure 5.16 exhibits SVM through using High Correlation Filter. Although the accuracy of the algorithm is quite high, it produces 9 false negatives which may cause harm to the system. In a collective manner SVM performs mediocre in terms of overall accuracy, although if we decide on using Random Forest Regression as our preferred dimension reduction technique, it will prove to be advantageous, providing 100% accurate results for the test set in consideration.

6.6 Collectively analyzing all the results

Our research proposes four classification algorithms; these being Random Forest Classification, k-NN, Support Vector Classification and Naïve Bayes Classification. Furthermore, we have used four separate dimension reduction techniques alongside the algorithms to give us 16 sets of results in the detection of these attacks. The datasets are taken from a security camera afflicted with the Mirai botnet and data is taken in both the benign and attack state; so, it shows an actual interpretation of a DDoS attack and the difference in the data of the UDP layer before and during the attack. From our set of results, we judged the algorithms based on their accuracy score, AUC score, F1 score and the time needed for the algorithm to execute in the prediction block of code. These results indicate a clear image about which algorithm is optimal for detecting real-time IoT related botnet attacks. The four prediction parameters were used since they highlighted different characteristics of a good classifier. Firstly, overall accuracy represents how many times our model got the classification right while comparing between the predictions and test class. However, overall accuracy is based on one specific cut-point and comparing based on this parameter results in the comparison with respect to one threshold value. AUC score adds more depth to the classifier in the sense that it represents the area under the ROC (Receiver Operating Characteristics) curve and the ROC curve is according to the sensitivity and specificity after it tries all the possible cut-points. So, in a sense, AUC score will eliminate the possibility of the classifier getting lucky in one cut-point. Lastly, the F1 score will provide information about the balance between precision and recall of the classifier. Precision refers to the number of positive identifications that was actually correct while Recall answers the question about the proportion of actual positives that were labeled correctly by the classifier. These three parameters along with the time taken to execute the prediction block will give us information that is not misleading or situational while keeping real-time intrusion detection in mind.

From our results, we can see that among the dimension reduction techniques, using Random Forest Regression to find the most important features of the dataset and removing all the other columns from consideration provide with the highest level of accuracy throughout all the parameters. However, using RFR for dimension reduction can induce a problem of over-fitting – something we should avoid to not incur misleading results.

Principal Component Analysis is another dimension reduction technique we have used and it tackles the problem that is ever-present in RFR based dimension reduction. PCA does not remove any data from the dataset; rather it does a linear transformation with respect to a real inner product space in order to preserve the inner product and turns a set of observations consisting of possibly correlated variables into a set of principal components which are linearly uncorrelated variables. We have selected the number of Principal Components to be 5 and among this the first variable has the highest possible variance and the later components have the highest variance while they fulfill the constraint of being orthogonal with respect to the previous component. Dimension reduction done with PCA provided us with a reduced dataset of 5 columns from the initial 115 columns without changing the essence of the dataset.

Independent Component Analysis (ICA) was done on the dataset as our third attempt at dimension reduction to make sure that the underlying factors are found; something that PCA fails at doing. Even though ICA is superficially related to PCA, it is a more powerful tool as it assumes that the data is the mixture of latent linear variables which are non-gaussian and mutually independent. It enhances the reduction in case PCA missed out on any hidden factors.

As for the last dimension reduction technique, we chose High Correlation

Filter. This technique finds out columns that have a similar trend which in turn result in very similar information. Training our dataset on such a dataset with similar data will lead to overfitting and hence High Correlation Filter is used. Hence, if two columns have similar data, they are reduced to one and the dimension of the dataset is reduced. Using these four techniques, our dataset was reduced and then we applied the aforementioned machine learning classifiers on these separately to generate sixteen sets of output.

While running the classifiers on the dataset reduced using Random Forest Regression, we have found that accuracy, F1 score and AUC score is consistently high. For k-NN and SVC, the results were 100% accurate as well. However, this can be due to dropping the variables that have the least feature importance and not taking into consideration about underlying factors. Furthermore, the prediction time in case of all the classifiers in this reduced dataset is relatively low due to the training time reduced with only two important features being available. k-NN still provides us with the longest time needed at 1.37 seconds because of its laziness as a learner. Naïve Bayes being an eager classifier provides with a 99.79% accuracy while also taking the least amount of time taken with 0.007 seconds. This is optimal for a real-time IDS and is actually the least time taken in our set of 16 possible outputs. Lastly, RFC provides with 99.99% accuracy in 0.29 seconds - which is close to perfect. However, it is inadvisable to run RFC on a dataset already reduced using RFR.

Alternatively, training and testing the classifiers on a dataset reduced by High Correlation Filter also provides with a consistently high level of accuracy and AUC score. Here, RFC is the best algorithm with an accuracy of 99.98%. This is due to High Correlation Filter reducing the correlation between the features of the dataset – something that RFC can take an advantage of since low correlation between models are the key in implementing RFC. Moreover, it takes 0.031 seconds in the prediction block and is optimal for real-time detection. Again, k-NN and SVC follow RFC closely in the comparison of accuracy with their accuracy being 99.96% while k-NN possesses a greater AUC score compared to SVC. However, the accuracy scores being this high allows us to ignore these marginal differences and lets us focus on the computational time needed to predict. Here, k-NN takes the most amount of time at 3.59 seconds and thus, even with its high accuracy, becomes unviable for detecting botnet attacks in real-time. SVC takes only 0.091 seconds to provide scores close to that same accuracy of k-NN; so, it is arguably better for our scenario. Naïve Bayes classifier again takes the least amount of time, finishing its execution of the prediction block just in 0.012 seconds but cannot provide the level of accuracy as the other three algorithms; even though it assumes the variables to not be correlated.

Running the classifiers on our PCA-reduced dataset leads to the highest accuracy throughout all four algorithms without any questions of overfitting or changing the essence of the data. Here, both RFC and SVC provide with 100% accuracy and F1 scores while RFC edges out SVC in terms of time taken by a margin of just 0.002 seconds. So arguably, RFC is the better classifier while running on a dataset reduced by PCA. k-NN performs excellently as well but again, it takes the most amount of time at 1.92 seconds and is inadvisable for real-time detection. Naïve Bayes performs the fastest but also comparatively the poorest among the four classifiers.

As our last case, running the classifiers on ICA-reduced datasets gave us mixed results. Firstly, Naïve Bayes is not a good approach in ICA-reduced datasets due to their conflicting philosophy. Naïve Bayes only garnered an accuracy of 91.27% which seems okay but is below par if compared to the other results. Furthermore, it provided with an accuracy of 84% and had quite a lot of false negative instances which are fatal. So, the low prediction time cannot compensate for Naïve Bayes in this particular case. The opposite of this scenario is SVC. SVC provides us with a 99.38% accuracy but takes a huge time to run across the prediction block. Taking 30 seconds is not viable at all in an IDS; so, SVC cannot be counted upon in ICA-reduced datasets. k-NN gives us a high accuracy but again takes the second longest to compute. Lastly, the best classifier for ICA-reduced datasets is RFC from our findings. RFC takes only 0.029 seconds to execute the prediction block with a 99% accuracy.

Chapter 7

Conclusion

IoT devices or Machine-To-Machine communication is rapidly on the rise with the number of devices that exchanged information among themselves being 15.41 million in 2015 and increasing to 26.66 million in 2019. Moreover, it is projected to become in the range of 75 million by 2025 [42]. With the recent growth and widespread use of IoT devices, the vulnerabilities of the system can be exploited easily. And the example of such cases is the evolution of the Mirai Botnet and other variants. Recently, the largest scale Botnet attack was ensued on Imperva, an online streaming application with 4, 00,000 IoT devices used to propagate a DDoS attack [43]. These cases of IoT related botnet attacks are directly related to the lack of security infrastructure in the devices due to cost-related reasons. Hence, the detection of IoT botnet attacks is essential with the rise of technology. Among the multiple Intrusion Detection Systems present such as Signature-Based, Anomaly-Based and Specification-Based, we have opted for the Anomaly-based detection which is effective in the detection of unknown vulnerabilities. In the problem statement in (section 1.2.1), we have stated how consumer IoT devices can become a part of a Bot Network and serve malicious purposes. We have been able to achieve some significant success in predicting the state of an IoT device, which can be either 'Attack' or 'Benign'. With the classifiers like kNN and Random forest producing the best results in terms of accuracy. Although, Naïve Bayes has not been the most decisive methods out of all four, it has a fast learning tendency and produces more false predictions evident in the confusion matrix. Even though Random Forest prevails in all four categories, it is susceptible to overfitting. With kNN coming in second in terms of the results generated, it has a better accuracy, but it suffers significantly in prediction time, or the run time of the algorithm. The scope with our research could provide two solutions to the problem, either we could opt for a fast learner like Naïve Bayes and perform more better in real-time application for Botnet detection in the transport layer, or we could go for a better accuracy provided by kNN and Random Forest. Even though it is unknown how Random Forest would perform in real world, due to overfitting. To conclude our research by saying that different dimension reduction techniques provide different scenarios for the classifiers to run. Comparatively, PCA should be the dimension reduction technique chosen due to its nature and also because it is less prone to overfitting the classifier later on. As for the classifier that we should pick, it should be Random Forest Classifier as it outperforms most other models in all of the dimension reduced datasets. k-NN and SVC both provide high accuracy as well but their run-time is not an option for real-time IDS. Naïve Bayes can be alternatively chosen if accuracy is sacrificed for getting a faster result. Even though we have been able to set the difference between the two states of an IoT device, with predictions based on transport layer data. We are still to simulate our model on more real world devices. We have only worked with a single IoT device in focus, and how it will perform on other devices is still unknown to us. We are only generating predictions, but no possible measures have been taken on preventing an attack, even though our work will provide a lot of scope in the future, it is still in the preliminary phase of IDS, and we still need to come up with methods to prevent an attack from taking place. This would eventually relive us from the pressure regarding the problem of Botnets attacking large organization networks. We have only scratched the surface of a field which still has a lot of work to do.

7.1 Future Direction

Our future plan with this model is to enhance it more and work towards developing an Intrusion Prevention System in tandem with botnet attack detection. Although, real-time detection and immediate response to botnet attacks can be a challenging task, due to the fact that the Mirai source code can be altered in multiple ways, to bypass the security protocols. Still there is a lot of scope to this research. If we are able to design a system that can perform Botnet attack detection and prevention simultaneously, it would immensely improve the security sector of IoT .The rise of botnet related attacks compels us to carry on this research and hopefully, better systems can be developed to tackle this issue.

Bibliography

[1] Biggs, J. (2016, October 10). *Hackers release source code for a powerful DDoS app called Mirai.* Retrieved from https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/.

[2] *Mirai.* (2016, December 28). Retrieved December 10, 2019, from https://www.cyber.nj.gov/threat-profiles/botnet-variants/mirai-botnet.

[3] *MMD-0056-2016 - Linux/Mirai, how an old ELF malcode is recycled.* (2016, September 1). Retrieved December 10, 2019, from https://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html.

[4] *Krebs on Security.* (2016, September 16). Retrieved December 10, 2019, from https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[5] Chung, J., & DeBeck, C. (2019, July 18). *I Can't Believe Mirais: Tracking the Infamous IoT Malware.* Retrieved December 10, 2019, from https://securityintelligence.com/posts/i-cant-believe-mirais-tracking-the-infamous-iot-malware-2/.

[6] Hackett, R. (2016, October 3). *Why a Hacker Dumped Code Behind Colossal Website-Trampling Botnet.* Retrieved December 10, 2019, from https://fortune.com/2016/10/03/botnet-code-ddos-hacker/.

[7] Newman, L. H. (2017, June 3). *What We Know About Friday's Massive East Coast Internet Outage.* Retrieved December 10, 2019, from https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/.

[8] *Krebs on Security.* (2017, January 17). Retrieved December 10, 2019, from https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/.

[9] Warren, T. (2012, July 11). *Microsoft advises users to disable Windows Gadgets following security vulnerability fears.* Retrieved December 10, 2019, from https://www.theverge.com/2012/7/11/3151143/windows-gadgets-security-vulnerability.

[10] Kan, M. (2016, October 18). *Hackers create more IoT botnets with Mirai source code.* Retrieved December 10, 2019, from https://www.itworld.com/article/3132570/hackers-create-more-iot-botnets-with-mirai-source-code.html.

[11] *IoTroop Botnet: The Full Investigation.* (2019, February 6). Retrieved December 10, 2019, from https://research.checkpoint.com/2017/iotroop-botnet-full-investigation/.

[12] Mcdermott, Christopher & Majdani, Farzan & Petrovski, Andrei. (2018). *Botnet Detection in the Internet of Things using Deep Learning Approaches.* 10.1109/IJCNN.2018.8489489.

[13] Wurzinger, P. & Bilge, L. & Holz, T. (2009). *Automatically Generating Models for Botnet Detection.* Retrieved December 10, 2019.

[14] *Internet.* (n.d.). Retrieved December 10, 2019, from http://www.businessdictionary.com/definition/internet.html.

[15] *What is the Internet of Things (IoT)?.* (n.d.). Retrieved December 10, 2019, from https://www.oracle.com/internet-of-things/what-is-iot.html.

[16] Rouse, M., & Shea, S. (2016, February 19). *What is IoT devices (internet of things devices)?.* Retrieved December 10, 2019, from https://internetofthingsagenda.techtarget.com/definition/IoT-device.

[17] Dey, A. (2019, May 10). *Internet Of Things (IoT) security, privacy, applications & trends.* Retrieved December 10, 2019, from https://medium.com/@arindey/internet-of-things-iot-security-privacy-applications-trends-3708953c6200.

[18] *IoT and Data Privacy.* (2017, September 27). Retrieved December 10, 2019, from https://innovationatwork.ieee.org/iot-data-privacy/.

[19] Weisman, S. (n.d.). *What is a distributed denial of service attack (DDoS) and what can you do about them?.* Retrieved December 10, 2019, from https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by- norton.html.

[20] Weisman, S. (n.d.). *What is a distributed denial of service attack (DDoS) and what can you do about them?* Retrieved December 10, 2019, from https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by- norton.html.

[21] Beal, V. (n.d.). *machine learning.* Retrieved December 10, 2019, from https://www.webopedia.com/TERM/M/machine-learning.html.

[22] Mishra, S. (2017, May 21). *Unsupervised Learning and Data Clustering.* Retrieved December 10, 2019, from https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a.

[23] Rawat, R. & Pilli, E. & Joshi, R. (2018). *Survey of Peer-to-Peer Botnets and Detection Frameworks.* International Journal of Network Security. 20. 547-557. 10.6633/IJNS.201805.20(3).18). Retrieved December 10, 2019, from https://www.researchgate.net/figure/Botnet-detection-taxonomy_fig1_326901249

[24] R. Doshi, N. Apthorpe, and N. Feamster. (2018) *Machine Learning DDoS Detection for Consumer Internet of Things Devices.* Retrieved December 10, 2019.

[25] Mcdermott, C. & Majdani, F. & V Petrovski, A. (2018). *Botnet Detection in the Internet of Things using Deep Learning Approaches.* 10.1109/IJCNN.2018.8489489. Retrieved December 10, 2019.

[26] Mai, L. & Park, M. (2016). *A comparison of clustering algorithms for botnet detection based on network flow.* (2016) Eighth International Conference on Ubiquitous and Future Networks.(ICUFN). doi:10.1109/icufn.2016.7537117. Retrieved December 10, 2019.

[27] Timcenko, V. & Gajin, S.(2018, March) *Machine learning based network anomaly detection for iot environments.* Retrieved December 10, 2019.

[28] Meidan, Y. (2018). *detection_of_IoT_botnet_attacks_N_BaIoT.* Retrieved December 10, 2019, from https://archive.ics.uci.edu/ml/machine-learning-databases/00442/.

[29] Rosero, P. & Diaz, P. & Salazar Castro, J. & Peña, D. & Anaya Isaza, A. & Alvarado, J. & Therón, R. & Peluffo, D. (2017). *Interactive Data Visualization Using Dimensionality Reduction and Similarity-Based Representations.* Lecture Notes in Computer Science. 10125. 334-342. 10.1007/978-3-319-52277-7_41. Retrieved December 10.

[30] *Practical Guide to Principal Component Analysis (PCA) in R & Python.* (2019, September 3). Retrieved December 10, from https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/.

[31] Sharma, N. (2019, January 15). *Importance of Distance Metrics in Machine Learning Modelling.* Retrieved December 10, 2019, from https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d.

[32] Kumar, N. (2019, January 14). *Naive Bayes Classifiers.* Retrieved December 10, 2019, from https://www.geeksforgeeks.org/naive-bayes-classifiers/.

[33] *Naive Bayes.* (n.d.). Retrieved December 10, 2019, from https://scikit-learn.org/stable/modules/naive_bayes.html.

[34] Pulipaka, G. P. (2017, January 19). *Applying Gaussian Naïve Bayes Classifier in Python: Part One.* Retrieved December 10, 2019, from https://medium.com/@gp_pulipaka/applying-gaussian-naïve-bayes-classifier-in-python-part-one-9f82aa8d9ec4.

[35] Kumar, N. (2019, January 14). *Naive Bayes Classifiers.* Retrieved December 10, 2019, from https://www.geeksforgeeks.org/naive-bayes-classifiers/.

[36] Pulipaka, G. P. (2017, January 19). *Applying Gaussian Naïve Bayes Classifier in Python: Part One.* Retrieved December 10,

2019, from https://medium.com/@gp_pulipaka/applying-gaussian-naïve-bayes-classifier-in-python-part-one-9f82aa8d9ec4.

[37] Yiu, T. (2019, August 14). *Understanding Random Forest.* Retrieved December 10, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[38] Arko,A. & Khan, S. (2019, August 9) *Anomaly Detectiom In IoT Using Machine Learning Algorithms*, (18-19). Retrieved September 10

[39] Shung, K. P. (2018, June 8). *Accuracy, Precision, Recall or F1?.* Retrieved December 10, from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[40] *Classification: Accuracy — Machine Learning Crash Course.* (n.d.). Retrieved December 10, from https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[41] Narkhede, S. (2019, May 26). *Understanding AUC - ROC Curve.* Retrieved December 10, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[42] Statista Research Department. (2019, November 14). *IoT: number of connected devices worldwide 2012-2025.* Retrieved December 10, 2019, from https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[43] Asokan, A., & Ross, R. (2019, July 26). *Massive Botnet Attack Used More Than 400,000 IoT Devices.* Retrieved December 10, 2019, from https://www.bankinfosecurity.com/massive-botnet-attack-used-more-than-400000-iot-devices-a-12841.

[44] Sayad, S. (n.d.). *K Nearest Neighbors – Classification.* Retrieved December 21, 2019, from https://www.saedsayad.com/k_nearest_neighbors.htm.

[45] Yiu, T. (2019, August 14). Understanding Random Forest. Retrieved December 21, 2019, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[46] T. Evgeniou and M. Pontil, "Workshop on support vector machines: Theory and applications", Support Vector Machines: Theory and Applications, p. 1, 2001.

[47] M. Roughan, S. Sen, 0. Spatscheck, and N. Duffield. (2004) *"Class-of-Service Mapping for QoS: A Statistical Signature-Based Approach to IP Traffic Classification"*. Proc. 4th ACM SIGCOMM Conf on Internet Measurement, Taormina, Sicily.

[48] Ahmed, T., Coates, M., & Lakhina, A. (2007). *Multivariate Online Anomaly Detection Using Kernel Recursive Least Squares.* Retrieved December 10, 2019, from https://ieeexplore.ieee.org/abstract/document/4215661.